ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Σχεδιασμός Αρχιτεκτονικών και απεικόνιση εφαρμογών σε Επαναδιαμορφούμενες πλατφόρμες με εργαλεία λογισμικού

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Χαράλαμπος Ν. Σιδηρόπουλος

Αθήνα, Μάιος 2016

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

Σχεδιασμός Αρχιτεκτονικών και απεικόνιση εφαρμογών σε Επαναδιαμορφούμενες πλατφόρμες με εργαλεία λογισμικού

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Χαράλαμπος Ν. Σιδηρόπουλος

Συμβουλευτική Επιτροπή : Δημήτριος Σούντρης
Κιαμάλ Πεκμεστζή
Γεώργιο Οικονομάκο

Εγκρίθηκε από την επταμελή εξεταστική επιτροπή την 20 Μαϊου 2016

.....	.....	.....
Δημήτριος Σούντρης Επικ. Καθηγητής ΕΜΠ (Επιβλέπων)	Κιαμάλ Πεκμεστζή Καθηγητής ΕΜΠ	Γιώργος Οικονομάκος Επικ. Καθηγητής ΕΜΠ
.....	.....	.....
Νεκτάριος Κοζύρης Καθηγητής ΕΜΠ	Γεώργιος Στασινόπουλος Καθηγητής ΕΜΠ	Διονύσιος Πνευματικάτος Καθηγητής Πολυτεχνείου Κρήτης
	.....	
	Francky Catthoor Prof. Katholieke Universiteit Leuven, Belgium	

Αθήνα, Μάιος 2016

.....

Χαράλαμπος Ν. Σιδηρόπουλος

Διδάκτωρ Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Πρόλογος

Θα περιέγραφα το διδακτορικό ως ένα προσωπικό ταξίδι. Αυτό ήταν το δικό μου βίωμα και κοιτώντας το ταξίδι πια με την ματιά του ανθρώπου που έφτασε στον προορισμό του, έτσι πρέπει να είναι. Αυτό που μένει και έχει σημασία, όσο παράξενο και αν ακούγεται σαν πρόλογος Διδακτορικής Διατριβής, δεν είναι η τεχνική κατάρτιση, ο τίτλος του Δόκτορα ή οι επιστημονικές καινοτομίες. Οι εμπειρίες και οι αποφάσεις, απαλλαγμένες πια από την πραγματικότητα η οποία τις δημιούργησε, μιας και είναι στο παρελθόν, γίνονται κομμάτι του εαυτού. Και σε μία μακρά και συχνά επίπονη διαδικασία όπως η εκπόνηση μιας διδακτορικής διατριβής είναι αρκετά έντονες και σημαντικές για να παραμείνουν για πάντα μαζί μου.

Σε αυτό το ταξίδι συνάντησα, γνώρισα και συμπορεύτηκα με καινούριους ανθρώπους, οι οποίοι υπήρξαν η πυξίδα και ο εξάντας μου. Πρώτος ο επιβλέπων καθηγητής μου Δημήτριος Σούντρης. Τον ευχαριστώ για την εμπιστοσύνη που μου έχει δείξει και για την καθοδήγησή του. Τον χαρακτηρίζει μια σπάνια σοφία την οποία ομολογώ ότι πολλές φορές αμφισβήτησα ιδίως στην αρχή του διδακτορικού. Ευχαριστώ τον Δρ. Κωνσταντίνο Σιώζιο για την συμβολή του, και για τις εμπειρίες που μου προσέφερε η συνεργασία μας. Ευχαριστώ θερμά και όλους τους συνεργάτες του εργαστηρίου, πρώην και νυν, τον Αλέξη, τον Ηρακλή, τον Σωτήρη, τον Διονύση, τον Κώστα, τον Νίκο, τον Γιώργο Ζ., τον Γιώργο Λ. και όλους τους υπόλοιπους συνεργάτες. Ιδιαίτερα θα ήθελα να ευχαριστήσω και αυτούς που συνεργάστηκα πιο στενά τον Γιάννη Κούτρα και τον Δημήτριο Ροδόπουλο. Τέλος ευχαριστώ θερμά τον καθ. Κιαμάλ Πεκμεστζή.

Εκτός από τους συνεργάτες μου υπήρξαν και άνθρωποι σε αυτό το ταξίδι οι οποίοι αποτέλεσαν την άγκυρα και τους φωτεινούς φάρους, ιδίως στις δύσκολες στιγμές του ταξιδιού. Ευχαριστώ μέσα από την καρδιά μου την αγαπημένη και παντοτινή φίλη μου Μάρω Μπάκα. Τον Βασίλη Γόνη τον οποίο θαυμάζω για την απίστευτη καλλιέργειά του, τον ευχαριστώ για τις ατέλειωτες συζητήσεις για την ζωή.

Ένας από τους σημαντικότερους ανθρώπους, και συνοδοιπόρος καθ όλη την διάρκεια του Διδακτορικού είναι η σύντροφός της ζωής μου. Νέλλη σε ευχαριστώ από την μία για την ανοχή και την αγάπη σου -μιας και ξέρω ότι καμιά φορά είμαι δύσκολος άνθρωπος- και από την άλλη σε ευχαριστώ για την τόση χαρά και ευτυχία που μου έχεις δώσει και μου δίνεις ακόμα.

Τέλος ευχαριστώ την οικογένειά μου και τους φίλους μου που με στήριξαν όλα αυτά τα χρόνια και βοήθησαν με τον τρόπο τους σε αυτό το ταξίδι.

Περίληψη Διατριβής

Τα τελευταία χρόνια, οι επαναδιαμορφούμενες αρχιτεκτονικές και πιο συγκεκριμένα τα Field Programmable Gate Arrays (FPGAs) έχουν γίνει βιώσιμες εναλλακτικές λύσεις στην θέση των Application Specific Integrated Circuits (ASICs). Το χαρακτηριστικό της τεχνολογίας των FPGAs είναι ότι υποστηρίζουν υλοποίηση εφαρμογών μέσω της κατάλληλης (επανα)διαμόρφωσης της λειτουργικότητας των πόρων υλικού. Αυτό επιτρέπει στα FPGAs να παρέχουν μεγαλύτερη ευελιξία, να βοηθούν στην ταχεία κατασκευή πρωτοτύπων για προϊόντα και να μειώνουν σημαντικά τα non-recurring engineering (NRE) κόστη, σε σύγκριση με τις ASIC Συσκευές.

Τα χαρακτηριστικά και οι δυνατότητες των αρχιτεκτονικών αυτών έχουν αλλάξει και έχουν βελτιωθεί σημαντικά τις τελευταίες δύο δεκαετίες. Από συστοιχίες Look-Up tables (LUT), έχουμε φτάσει σε ετερογενείς συσκευές που ενσωματώνουν μια σειρά από στοιχεία υλικού (π.χ., LUTs με διαφορετικά μεγέθη, μικροεπεξεργαστές, DSP και RAM μπλοκ κλπ.). Η λογική δομή ενός FPGA έχει αλλάξει σταδιακά από μια ομοιογενή και τακτική αρχιτεκτονική σε μια ετερογενή System on Chip (SoC) συσκευή. Η πολυπλοκότητα των σημερινών εφαρμογών εισάγει συνήθως περιορισμούς στην αρχιτεκτονική οργάνωση των FPGA. Ακόμη και αν η ζήτηση για επιπλέον πόρους λογικής ικανοποιείται με πλατφόρμες που αποτελούνται από πιο πολύπλοκα λογικά μπλοκ, ή CLBs, (π.χ. με περισσότερα LUTs), το πρόβλημα αυτό εξακολουθεί να υφίσταται με τις πιο απαιτητικές σε θέμα επικοινωνίας εφαρμογές (π.χ. τηλεπικοινωνίες, κρυπτογράφηση και την επεξεργασία εικόνας, βίντεο), δεδομένου ότι η απόδοσή τους εξαρτάται συνήθως από τη διαθεσιμότητα σε I/O bandwidth.

Η παρούσα διδακτορική διατριβή διερευνεί τις προκλήσεις και προτείνει νέες λύσεις στο πεδίο της απεικόνισης (mapping) μιας εφαρμογής σε Field Programmable Gate Arrays. Ο στόχος είναι να σκιαγραφηθούν και να αναλυθούν, τα εμπόδια που περιορίζουν την αποδοτικότητα της διαδικασίας απεικόνισης και να προταθούν νέες λύσεις με στόχο την αύξησή της. Προς αυτόν τον στόχο αναπτύχθηκε μια καινοτόμα μεθοδολογία η οποία επιτρέπει την ταχεία διερεύνηση σε επίπεδο αρχιτεκτονικής διαφορετικών οργανώσεων και ιεραρχιών μνήμης, σε ετερογενή FPGAs. Παράλληλα με την μεθοδολογία αναπτύχθηκε και ένα λογισμικό πλαίσιο που υποστηρίζει την απεικόνιση μιας εφαρμογής πάνω στις προαναφερθείσες αρχιτεκτονικές. Το προτεινόμενο πλαίσιο επιτρέπει την διερεύνηση ιεραρχιών οποιουδήποτε τύπου αρχιτεκτονικού μπλοκ, όχι μόνο μνημών. Πάνω στο θέμα των αρχιτεκτονικών, για την άμβλυνση του προβλήματος του I/O

bandwidth που εμφανίζεται σε πιο πολύπλοκες εφαρμογές και για την αύξηση των επιδόσεων γενικά προτάθηκε ένα νέο τριδιάστατο αρχιτεκτονικό πρότυπο FPGA. Η τριδιάστατη αυτή αρχιτεκτονική αποτελείται από ετερογενή στρώματα, σε αντίθεση με προηγούμενες προσεγγίσεις όπου κάθε στρώμα είναι αντίγραφο του προηγουμένου. Το case study που χρησιμοποιείται αποτελείται από τρία στρώματα, σε καθένα εκ των οποίων τοποθετείται ξεχωριστά η λογική, η μνήμη, και τα I/O μπλοκ. Η επιλογή τριών στρωμάτων με τα συγκεκριμένα αρχιτεκτονικά στοιχεία δεν περιορίζει την γενικότητα της προτεινόμενης λύσης. Επιπρόσθετα αναπτύχθηκε το κατάλληλο λογισμικό πλαίσιο που υποστηρίζει την διερεύνηση τέτοιων αρχιτεκτονικών και την απεικόνιση εφαρμογών πάνω σε τέτοιες επαναδιαμορφούμενες αρχιτεκτονικές.

Εκτός από τις γνωστές προκλήσεις στο φυσικό επίπεδο που οφείλονται στην συρρίκνωση των τρανζίστορ, η αυξημένη πολυπλοκότητα των εφαρμογών αλλά και της αρχιτεκτονικής των FPGAs, καθιστά την αποτελεσματικότητα και την αποδοτικότητα των CAD εργαλείων που χρησιμοποιούνται ακόμη πιο κρίσιμες. Οι τεχνικές που επιταχύνουν τους βασικούς αλγόριθμους CAD μπορούν να επιφέρουν σημαντικές αλλαγές στο χρόνο σχεδιασμού ενός προϊόντος, ενώ πολλοί σχεδιαστές μπορεί να είναι πρόθυμοι να δεχτούν μικρή υποβάθμιση στην ποιότητα της λύσης με αντάλλαγμα ένα βελτιωμένο χρόνο εκτέλεσης των εργαλείων CAD. Προκειμένου να ενταχθούν αποτελεσματικά σε αυτό το νέο τοπίο, τα FPGAs πρέπει να υποστηρίζουν ταχεία ανάπτυξη και απεικόνιση εφαρμογών. Η βιομηχανία έχει κάνει βήματα για την ταχύτερη ανάπτυξη εφαρμογών, εξερευνώντας ποικίλες λύσεις, όπως High Level Synthesis (HLS). Τα FPGAs έχουν διερευνηθεί ως μια βιώσιμη πλατφόρμα για διάφορες εφαρμογές High Performance Computing (HPC) και ενσωματωμένων συστημάτων κυρίως λόγω του εγγενούς παραλληλισμού και της δυνατότητας επαναπρογραμματισμού που μπορεί να εφαρμοστεί είτε στο σχεδιασμό ή το χρόνο εκτέλεσης.

Για την αντιμετώπιση αυτών των περιορισμών σε αυτή την διδακτορική διατριβή εισάγεται μια νέα μεθοδολογία που έχει ως στόχο την ταχεία απεικόνιση εφαρμογών σε FPGAs. Ο στόχος αυτής της προσέγγισης είναι να μειωθεί σημαντικά ο χρόνος εκτέλεσης χωρίς ταυτόχρονα να υποβαθμιστούν σημαντικά οι επιδόσεις της εφαρμογής. Για τον ίδιο σκοπό, αναπτύχθηκε μια μεθοδολογία cloud και το αντίστοιχο λογισμικό πλαίσιο προκειμένου να καταστεί δυνατή η αποτελεσματική απεικόνιση πολλαπλών εφαρμογών κατά το χρόνο εκτέλεσης σε ένα ή περισσότερα FPGAs. Η προτεινόμενη λύση άρει τα προαναφερθέντα προβλήματα προσφέροντας γρήγορους χρόνους εκτέλεσης και επιτρέποντας να κλιμακωθεί η διαδικασία της απεικόνισης σε πολλούς πυρήνες. Προκειμένου να αξιοποιηθούν τα FPGAs σε ένα δυναμικό περιβάλλον προτάθηκε μια νέα μεθοδολογία και τα απαραίτητα εργαλεία που επιτρέπουν την αποδοτική απεικόνιση πολλαπλών εφαρμογών σε ετερογενή FPGAs. Με τη χρήση δυναμικών εικονικών

πυρήνων, προσαρμοσμένων κατανεμητών μνήμης και βελτιστοποιήσεις στην διαχείριση μνήμης, ξεπεράστηκαν οι περιορισμοί που επιβάλλονται από τα CAD εργαλεία και αποδείχτηκε θεωρητικά ότι η απεικόνιση εφαρμογών σε FPGAs μπορεί να γίνεται κατά τον χρόνο εκτέλεσης ακόμα και σε ενσωματωμένα συστήματα.

Λέξεις- Κλειδιά {FPGAs, Επαναδιαμορφούμενες Αρχιτεκτονικές, CAD Εργαλεία, Αλγόριθμοι, Τρισδιάστατες αρχιτέκτονες}

Thesis Abstract

In recent years, reconfigurable architectures and more specifically Field Programmable Gate Arrays (FPGAs) have become efficient alternatives to Application Specific Integrated Circuits (ASICs). The key to FPGAs' popularity is their feature to support application implementation by appropriately (re-)configuring the functionality of hardware resources. This allows FPGAs to provide higher flexibility, rapid product prototyping and significantly reduced non-recurring engineering (NRE) costs, as compared to ASIC (Application-Specific Integrated Circuit) devices.

The features and capabilities of these reconfigurable architectures have changed and improved significantly over the last two decades. From simple arrays of Look-Up tables (LUT), we have now reached heterogeneous devices incorporating a series of hardware components (e.g. LUTs with different sizes, microprocessors, DSP and RAM blocks etc.). The logical structure of an FPGA has changed gradually from a homogeneous and regular architecture towards a heterogeneous System on Chip (SoC) device. Furthermore the complexity of current applications usually introduces restrictions on the architectural organization of the FPGA. Even if the demand for additional logical resources is met with platforms consisting of more complex logic blocks, or CLBs, (e.g. with more LUTs), the problem persists with communication intensive applications (e.g. telecommunications, encryption and image processing, video) since their performance is typically dependent on the availability of I/O bandwidth.

This Doctoral thesis investigates the challenges and presents novel solutions on the field of application mapping onto Field Programmable Reconfigurable Arrays. The goal is to analyze and profile the obstacles and the constrains that limit the efficiency of application mapping and propose novel solutions to alleviate them. For this reason a novel software-supported methodology is introduced for enabling rapid architecture-level exploration for heterogeneous FPGAs that consist of different memory organizations and/ or hierarchies. Additionally a new tool framework was developed that enables application mapping onto these heterogeneous FPGAs. The proposed framework enables among others simultaneous handling of heterogeneous blocks with different hierarchies. In this thesis, in order to alleviate the communication bottleneck of complex applications and towards higher performance a novel architecture template was proposed for designing heterogeneous 3-D FPGAs with layers that contain blocks of different type. Added to the known challenges in the physical layer created by transistor shrinking, is the increased complexity of the target circuits and FPGAs, which makes the effectiveness and efficiency of the employed CAD tools become even more important. Techniques that accelerate core CAD algorithms can bring about important changes in product design times for these applications, whereas many designers may be willing

to trade-off some quality of the solution for an improved run-time of the CAD tools. In order to integrate efficiently on this new landscape, FPGAs need to support fast application development and implementation. Industry has taken steps towards faster application development, exploring diverse solutions. FPGAs have been explored as a viable platform for various high performance applications and various embedded computing domains mainly due to the inherent parallelism and reprogrammability feature which can be applied either at design or run time.

In order to address these limitations in this thesis, a novel methodology is introduced that targets to perform fast application's implementation onto FPGAs. The goal of this approach is to reduce considerably the run-time overhead with the minimum possible performance degradation. Towards the same goals a cloud-inspired methodology and the supporting framework are proposed in order to enable the efficient mapping of multiple designs at runtime onto a single or multiple FPGAs. The introduced solution alleviates the previously mentioned constrains by offering fast execution times and enabling portability and scalability. In order to dynamically utilize FPGAs on run time we propose a novel methodology and the supporting toolflow that enable efficient mapping of multiple applications onto heterogeneous FPGAs. With the use of dynamic virtual kernels, memory optimizations and custom memory allocators, we alleviate the constrains imposed by CAD tools, and provide a proof of concept that application mapping onto FPGAs can be done on run time even on embedded systems.

Keywords {FPGAs, Reconfigurable Architectures, CAD Tools, Algorithms, 3D Architectures}

Why a four year old child could understand this.
Run out and get me a four year old child,
I can't make head or tail out of it.

Marx

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

Κεφάλαιο 1

Εισαγωγή

1.1 Field Programmable Gate Arrays

1.1.1 Αρχιτεκτονική Δομή

Ένα field-programmable gate array (FPGA), αποτελείται από μπλοκ λογικής ενσωματωμένα σε μία γενική δομή δρομολόγησης. Τα λογικά μπλοκ στην απλούστερη μορφή τους περιέχουν υπολογιστικά στοιχεία για την εκτέλεση απλών συναρτήσεων συνδυαστικής λογικής, καθώς και flip-flops για την υλοποίηση ακολουθιακής λογικής. Τα σύγχρονα λογικά μπλοκ επεκτάθηκαν ώστε να συμπεριλαμβάνουν πρόσθετα στοιχεία υψηλής πολυπλοκότητας. Οι λογικές μονάδες συνήθως δρουν ως απλές μνήμες και ως εκ τούτου, οποιαδήποτε Boolean συνδυαστική συνάρτηση με έξι ως τέσσερις εισόδους μπορεί να υλοποιηθεί σε κάθε λογικό μπλοκ. Η γενική δομή δρομολόγησης συνήθως επιτρέπει πλήρως συνδεδεμένη καλωδίωση μέσω Switchboxes (SBs) και Connection boxes (CBs), που σημαίνει ότι στην έξοδο ενός λογικού μπλοκ μπορεί να φτάσει οποιαδήποτε είσοδος άλλου μπλοκ έτσι ώστε όλα τα λογικά στοιχεία να συνδεθούν με τον επιθυμητό τρόπο.

Αυτή η γενικότητα και ευελιξία επιτρέπει την υλοποίηση πολύπλοκων κυκλωμάτων πάνω σε FPGAs. Οι τρέχουσες συσκευές που προσφέρονται από μεγάλους εμπορικούς κατασκευαστές όπως Xilinx [8] και Altera [9] μπορούν να υλοποιήσουν κυκλώματα της τάξης των εκατομμύρια βασικών πυλών, που τρέχουν σε ταχύτητες εκατοντάδων Megahertz. Με σκοπό να αυξηθούν τόσο η απόδοση όσο και η χωρητικότητα τέτοιων αρχιτεκτονικών, πρόσθετα στοιχεία υλικού είναι συχνά ενσωματωμένα, όπως μνήμες, Digital Signal Processors (DSP), λογική fast-carry για αριθμητικές και λογικές λειτουργίες και πλήρεις μικροεπεξεργαστές. Με αυτά τα προκαθορισμένα, μπλοκ υλικού, τα οποία είναι κατασκευασμένα στο πυρίτιο, τα FPGAs είναι ικανά να υλοποιήσουν ολοκληρωμένα συστήματα και να λειτουργήσουν ως System on Chip συσκευές (SoC).

Σε ένα FPGA όλα τα λογικά μπλοκ και στοιχεία δρομολόγησης ελέγχονται από σημεία προγραμματισμού, τα οποία μπορεί να βασίζονται τεχνολογίες antifuse, Flash, ή SRAM. Η κύρια επιλογή για υλοποίηση επαναδιαμορφούμενων αρχιτεκτονικών σήμερα είναι τα SRAM-based FPGAs. Σε αυτόν τον τύπο συσκευών FPGA, κάθε δρομολόγηση καλωδίων και κάθε στοιχείο λογικής ελέγχεται από ένα απλό bit μνήμης. Ο προγραμματισμός όλων αυτών των bits μνήμης ονομάζεται bitstream. Όταν το bitstream είναι φορτωμένο στη συσκευή, ο συνδυασμός των στοιχείων λογικής και η διασύνδεσή τους (δρομολόγηση) θα υλοποιήσουν την επιθυμητή λειτουργία του κυκλώματος. Η διαμόρφωση μπορεί να πραγματοποιηθεί γρήγορα, δεν είναι μόνιμη και γενικά διαρκεί όσο το FPGA είναι ανοιχτό, επιτρέποντας παραμετροποίηση και από την πλευρά του χρήστη, ή ακόμα και στο τελικό προϊόν.

1.1.2 CAD Εργαλεία

Από την στιγμή που η διαμόρφωση ενός FPGA περιλαμβάνει την αποθήκευση τιμών σε θέσεις μνήμης, μπορεί να θεωρηθεί ως μεταγλώττιση και στη συνέχεια, φόρτωμα ενός προγράμματος σε έναν υπολογιστή. Η δημιουργία ενός FPGA-based κυκλώματος είναι μια απλή διαδικασία δημιουργίας ενός bitstream για να φορτωθεί στη συσκευή. Αν και υπάρχουν εργαλεία για να το κάνουμε αυτό από γλώσσες λογισμικού υψηλού επιπέδου, σχηματικές παραστάσεις και άλλες μορφές, λόγω των περιορισμών των μεθοδολογιών αυτών, οι σχεδιαστές FPGA συνήθως ξεκινούν με μια υλοποίηση γραμμένη σε γλώσσα περιγραφής υλικού (HDL) όπως Verilog ή VHDL. Αυτή η περιγραφή έχει βελτιστοποιηθεί για να ταιριάξει στην διαθέσιμη λογική του FPGA μέσω μιας σειράς από βήματα. Τα βήματα αυτά εκτελούνται από Computer-aided design (CAD) εργαλεία που προσφέρονται σχεδόν αποκλειστικά από τον κατασκευαστή της κάθε συσκευής FPGA.

Το πρώτο βήμα είναι η σύνθεση η οποία μετατρέπει υψηλού επιπέδου λογικής στοιχεία και behavioral κώδικα σε λογικές πύλες. Αυτό το βήμα είναι στενά συζευγμένο με το επόμενο βήμα το οποίο λέγεται technology mapping, στην οποίο οι πύλες ομαδοποιούνται ιδανικά για να ταιριάζουν με τους πόρους λογικής του FPGA. Οι στόχοι του technology mapping συνήθως είναι να μειωθεί το εμβαδόν του κυκλώματος, η αύξηση των επιδόσεων, η μείωση της κατανάλωσης ενέργειας ή οποιοσδήποτε συνδυασμός των παραπάνω.

Το επόμενο βήμα είναι η τοποθέτηση των λογικών μπλοκ που δημιουργούνται από τα προηγούμενα στάδια σε φυσικά λογικά μπλοκ του FPGA. Ο στόχος εδώ είναι να τοποθετηθεί κάθε μπλοκ, όσο το δυνατόν πιο κοντά σε όλα τα μπλοκ που συνδέεται με σκοπό τη μείωση της καθυστέρησης διασύνδεσης και να επιτευχθεί καλύτερη συνολική απόδοση του κυκλώματος που απεικονίζεται πάνω στην επαναδιαμορφούμενη συσκευή.

Στο στάδιο της δρομολόγησης οι φυσικοί πόροι διασύνδεσης επιλέγονται, έτσι ώστε να συνδεθούν όλα τα μπλοκ με τον σωστό τρόπο για την υλοποίηση του κυκλώματος που έχει δοθεί από το χρήστη. Πιο συγκεκριμένα ο στόχος της δρομολόγησης είναι να βρεθεί η συντομότερη διαδρομή για κάθε σύνδεση μεταξύ των τοποθετημένων μπλοκ της εφαρμογής. Εφόσον οι πόροι δρομολόγησης του FPGA είναι πεπερασμένοι και προκαθορισμένοι, η συμφόρηση είναι ένα σοβαρό πρόβλημα που διέπει τις αποφάσεις δρομολόγησης ακόμη περισσότερο από ότι η εύρεση της συντομότερης διαδρομής. Δεδομένου ότι ο στόχος της διαδικασίας της δρομολόγησης είναι να βρεθούν τα συντομότερα μονοπάτια αποφεύγοντας παράλληλα την συμφόρηση, η αποτελεσματικότητά της εξαρτάται από την ποιότητα της λύσης στο βήμα της τοποθέτησης.

Τέλος, στο βήμα της παραγωγής bitstream δημιουργείται ένα binary αρχείο που καθορίζει το σύνολο των προγραμματιζόμενων bits στο FPGA, έτσι ώστε να διαμορφωθούν κατάλληλα τα στοιχεία λογικής και δρομολόγησης.

1.1.3 Εφαρμογή των FPGAs

Τα Field Programmable Gate Arrays συνδυάζουν την ευελιξία του λογισμικού με την απόδοση του υλικού. Ένας σχεδιαστής FPGA λοιπόν θα πρέπει να σκέφτεται διαφορετικά από τους σχεδιαστές που χρησιμοποιούν άλλες συσκευές. Οι προγραμματιστές λογισμικού γράφουν συνήθως σειριακά προγράμματα που εκμεταλλεύονται την ικανότητα του μικροεπεξεργαστή να υπολογίσει ταχέως μια σειρά από βήματα. Ακόμα και στον παράλληλο προγραμματισμό, ο παραλληλισμός είναι συνήθως σε επίπεδο συνάρτησης ή σε επίπεδο βρόχου. Σε αντίθεση, μία υψηλής ποιότητας FPGA εφαρμογή είναι χωρικά παράλληλη σε πολύ λεπτό επίπεδο, με την ταυτόχρονη χρήση πολλαπλών πόρων απλωμένων πάνω στο FPGA τσιπ, το οποίο θα αποδώσει ένα τεράστιο ποσό υπολογισμών. Αυτό οδηγεί σε επιτάχυνση του υπολογισμού ακόμη και αν το FPGA έχει συχνότητα μόνο μερικές εκατοντάδες MHz σε σύγκριση με τις συχνότητες GHz που προσφέρονται από μικροεπεξεργαστές.

Η ευελιξία των FPGAs δίνει στους σχεδιαστές νέες ευκαιρίες που δεν υποστηρίζονται σε ASICs και άλλες σταθερές συσκευές. Σχέδια για FPGAs μπορούν να αναπτυχθούν και να υλοποιηθούν, πιο γρήγορα από τα αντίστοιχα σε ASICs και μπορεί ακόμη και να επαναπρογραμματιστούν με νέες λειτουργικότητες. Δεν απαιτούν τις τεράστιες ομάδες σχεδιασμού και την προσπάθεια που απαιτείται για validation και verification στα κυκλώματα ASIC, μειώνοντας έτσι σημαντικά τα Non Recurring Engineering (NRE) κόστη, για την ανάπτυξη ή την αναβάθμιση ενός προϊόντος. Επιπλέον, η δυνατότητα αλλαγής

της διαμόρφωση, ακόμα και κατά το χρόνο εκτέλεσης, επιτρέπει νέες λειτουργικότητες, όπως βελτιστοποιήσεις κατά τον χρόνο εκτέλεσης, την εναλλαγή εργασιών, αυτοδιάγνωση προβλημάτων, κ.λπ. Ωστόσο, επειδή τα FPGAs είναι αισθητά πιο αργά και καταναλώνουν περισσότερη ισχύ από ότι τα ASIC, οι σχεδιαστές πρέπει να βελτιστοποιήσουν προσεκτικά την εφαρμογή τους.

1.2 Προκλήσεις και συναφής έρευνα

1.2.1 Ετερογενή FPGAs

Τα τελευταία χρόνια, επαναδιαμορφούμενες αρχιτεκτονικές και πιο συγκεκριμένα τα Field Programmable Gate Πίνακες (FPGAs) έχουν γίνει αποδεκτές εναλλακτικές λύσεις των Application Specific Integrated Circuits (ASICs). Τα χαρακτηριστικά και οι δυνατότητες των αρχιτεκτονικών αυτών έχουν αλλάξει και να βελτιωθεί σημαντικά τις τελευταίες δύο δεκαετίες, από συστοιχίες look-up tables (LUT), σε ετερογενείς συσκευές που ενσωματώνουν μια σειρά από στοιχεία υλικού (π.χ., LUTs με διαφορετικά μεγέθη, μικροεπεξεργαστές, ενότητες DSP, μπλοκ RAM, κ.λπ.). Η λογική δομή μιας FPGA άλλαξε σταδιακά από ομοιογενής και τακτική αρχιτεκτονική σε ετερογενής (ή piece-wise ομογενής).

Προηγούμενες μελέτες [10] [5] [11] έχουν αποδείξει ότι ένα από τα σημαντικά βήματα για το σχεδιασμό μιας αποδοτικής συσκευής FPGA είναι η εξερεύνηση σε επίπεδο αρχιτεκτονικής. Το βήμα αυτό καθορίζει την οργάνωση (δηλαδή κάτοψη), την τεχνολογία και την κυκλωματική αρχιτεκτονική του κάθε στοιχείου, καθώς και τις παραμέτρους για τις δομές που συνιστούν την FPGA πλατφόρμα (π.χ. μέγεθος look-up table, το πλάτος του καναλιού δρομολόγησης, το μέγεθος του πίνακα, κ.λπ.). Το πρόβλημα της επαρκούς και ακριβούς εξερεύνησης σε επίπεδο αρχιτεκτονικής γίνεται πολύ πιο σημαντικό στις μέρες μας, λόγω της αυξημένης πολυπλοκότητας που δημιουργείται από τα ετερογενή IP μπλοκ βρίσκονται σε πλατφόρμες FPGA.

Προκειμένου να επιλυθεί αυτό το πρόβλημα, μια σειρά μεθοδολογιών και Computer-Aided Design (CAD) εργαλεία έχουν προταθεί. Οι λύσεις αυτές κινούνται στο επίπεδο της σύνθεσης και του technology mapping [12] [13], τοποθέτησης και δρομολόγησης (P&R) [14] [5], καθώς της εκτίμησης κατανάλωσης ισχύος και ενέργειας [15].

Η ανάπτυξη νέων εργαλείων που στοχεύουν τις επαναδιαμορφούμενες αρχιτεκτονικές απασχολεί τόσο την ακαδημαϊκή όσο και την βιομηχανική έρευνα . Πιο συγκεκριμένα, εργαλεία που έχουν αναπτυχθεί στον ακαδημαϊκό χώρο έχουν κυρίως επικεντρωθεί στην εξερεύνηση σε επίπεδο αρχιτεκτονικής για ομοιογενή FPGAs (δηλαδή συσκευές

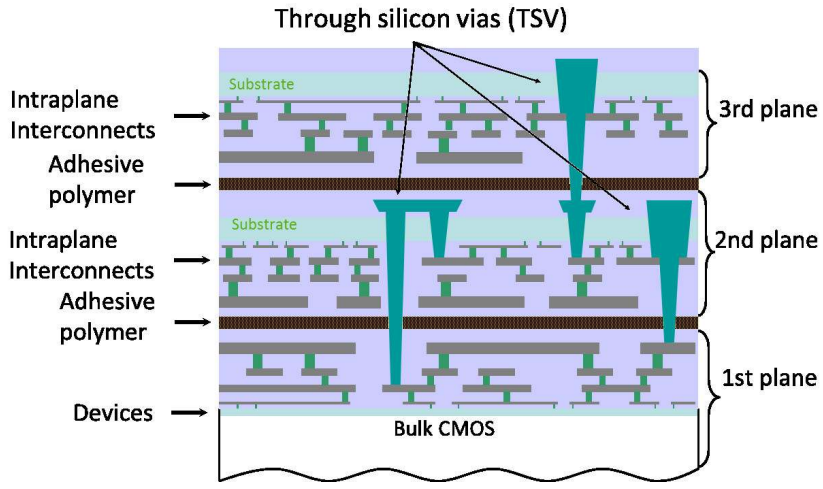
που αποτελείτο αποκλειστικά και μόνο από λογικά μπλοκ (CLBs)). Ακόμη και αν οι λύσεις αυτές είναι επαρκείς για την αξιολόγηση νέων CAD αλγορίθμων, δεν μπορούν να χειριστούν πρόσθετα IP μπλοκ (π.χ. μνήμες, τα DSP, ενσωματωμένους επεξεργαστές, κλπ) που βρίσκονται σε σύγχρονες επαναδιαμορφούμενες αρχιτεκτονικές. Από την άλλη πλευρά, εμπορικά λογισμικά πλαίσια υποστηρίζουν FPGA συσκευές με πολλά ετερογενή μπλοκ, αλλά, δυστυχώς, επιτρέπουν μόνο έναν ελάχιστο βαθμό εξερεύνησης σε επίπεδο αρχιτεκτονικής.

Το Verilog-to-Routing (VTR) project [14], είναι ένα ακαδημαϊκό open-source πλαίσιο για τη διεξαγωγή έρευνας στην αρχιτεκτονική δομή ενός FPGA και την ανάπτυξη CAD αλγορίθμων και εργαλείων. Αυτή η ροή λογισμικού αρχίζει με μια περιγραφή σε γλώσσα υλικού Verilog και ένα αρχείο που περιγράφει την υποθετική αρχιτεκτονική και έπειτα επεξεργάζεται, συνθέτει, πακετάρει, τοποθετεί και δρομολογεί το κύκλωμα, εκτελεί ανάλυση χρονισμού για την τελική λύση, καθώς και εκτίμηση κατανάλωσης ισχύος. Αυτό μπορεί συνδεθεί σε κάποιο βαθμό με τα εμπορικά εργαλεία όπως το Quartus της Altera [6] ή το Xilinx Vivado [16], αλλά η επικοινωνία μεταξύ των εργαλείων είναι προβληματική και περιορισμένη στην καλύτερη περίπτωση. Ακόμη και αν ο συνδυασμός αυτών των δύο λύσεων δυνητικά μπορεί να άρει τον περιορισμό σχετικά με την υποστήριξη της ετερογένειας, τα αποτελέσματα που προέρχονται υπολείπονται ακρίβειας.

1.2.2 Τρισδιάστατα FPGAs

Το κλειδί για την δημοτικότητα των FPGAs είναι το χαρακτηριστικό τους να υλοποιούν εφαρμογές μέσω (επανα)-διαμόρφωσης της λειτουργικότητας των πόρων του υλικού. Αυτό επιτρέπει στα FPGAs να παρέχουν μεγαλύτερη ευελιξία, ταχεία προτυποποίηση των προϊόντων και μειώνει σημαντικά τα non-recurring engineering (NRE) κόστη, σε σύγκριση με ASIC (Application-Specific Integrated Circuit) συσκευές. Δεδομένου ότι υπάρχει σήμερα μια ολοένα αυξανόμενη για ταχύτερες, μικρότερες, φθηνότερες και χαμηλής ενέργειας συσκευές τα FPGA αποκτούν μεγαλύτερη σημασία.

Για δεκαετίες, οι κατασκευαστές ημιαγωγών συρρικνώνουν το μέγεθος των τρανζίστορ στα ολοκληρωμένα κυκλώματα (ICs) για την επίτευξη των ετήσιων αυξήσεων στις επιδόσεις που περιγράφονται από το Νόμο του Moore, η οποία υπάρχει μόνο επειδή η καθυστέρηση RC ήταν αμελητέα, σε σχέση με την καθυστέρηση διάδοσης του σήματος. Για την τεχνολογίες στο επίπεδο των νανόμετρων, ωστόσο, η καθυστέρηση RC γίνεται κυρίαρχος παράγοντας. Επιπλέον, προηγούμενες μελέτες έδειξαν ότι στα 130nm περίπου το 51% της ενέργειας του μικροεπεξεργαστή καταναλώνεται από το δίκτυο διασύνδεσης [17]. Αυτό έχει προκαλέσει πολλές συζητήσεις σχετικά με το τέλος της



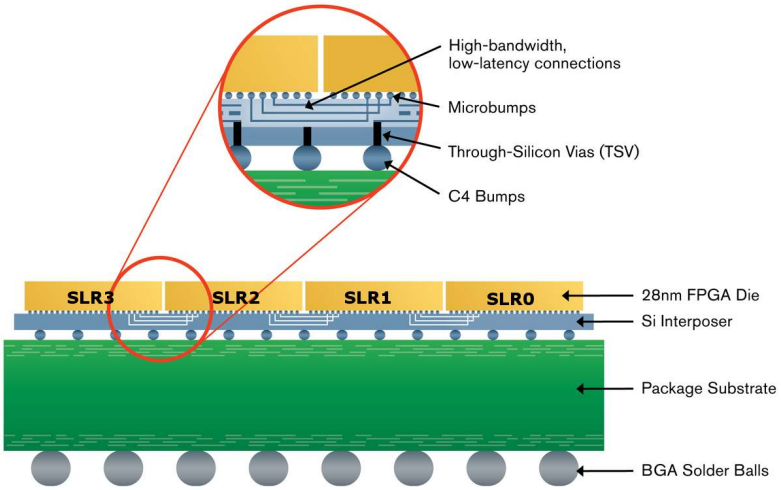
Σχήμα 1.1: 3-D chip με TSV τεχνολογία [1].

σμίκρυνσης των τρανζίστορ, όπως την γνωρίζουμε και επιτάχυνε την αναζήτηση λύσεων πέρα από τα όρια των σημερινών 2-D αρχιτεκτονικών.

Το τρισδιάστατο (3-D) chip stacking θεωρείται από πολλούς ως η λύση που θα ικανοποιήσει όλες τις προαναφερόμενες απαιτήσεις [18]. Η στοίβαξη πολλαπλών μητρών στον κατακόρυφο άξονα και τη διασύνδεση τους χρησιμοποιώντας πολύ λεπτά Through Silicon Vias (TSVs) επιτρέπει τη δημιουργία τσιπ με επίπεδα σε διαφορετικές τεχνολογίες διατηρώντας μικρό συντελεστή εμβαδού [19]. Λόγω γειτνίασης κατά μήκος του Z άξονα, οι διασυνδέσεις μεταξύ των στοιχείων του συστήματος είναι μικρότερες σε μήκος, το οποίο με τη σειρά του οδηγεί σε μειωμένη καθυστέρηση διάδοσης του σήματος σε σύγκριση με τις συμβατικές (δηλαδή 2-D) αρχιτεκτονικές [19] [18].

Το ενδιαφέρον για τρισδιάστατα FPGAs έχει εξετασθεί ήδη από τη βιομηχανία, δεδομένου ότι υπάρχουν εμπορικές προσεγγίσεις. Χαρακτηριστικά παραδείγματα είναι τα τρισδιάστατα FPGA στις συσκευές από την Xilinx (Virtex-7 & UltraScale FPGAs [2]) και την Tezzaron. Αυτές οι αρχιτεκτονικές διαφέρουν με τον τρόπο που τα στρώματα υλοποιούνται φυσικά. Συγκεκριμένα, οι συσκευές από Tezzaron υιοθετούν μια συμβατική τρισδιάστατη τεχνολογική διαδικασία, wafer-level stacking, όπου η σύνδεση ενδιάμεσων στρωμάτων παρέχεται μέσω TSVs (δείτε Σχήμα 1.1). Μία τέτοια σύνδεση πηγαίνει από την μπροστινή πλευρά του wafer (συνήθως συνδέεται με ένα από τα χαμηλότερα στρώματα μετάλλων) μέσω του wafer και τέλος έξω από την πίσω πλευρά [18]. Ανάλογα με την επιλεγμένη τεχνολογία, το TSVs ποικίλλει σε διάμετρο από 1 ως 10μm, με βάθος από 5 ως 10$\times$ το πλάτος του [20] [21].

Από την άλλη μεριά οι Virtex-7 συσκευές από την Xilinx, επίσης γνωστές ως 2.5D FPGAs, στηρίζονται στο Stacked Silicon Interconnect Technology (SSIT) [2] για να δρομολογήσουν σήματα μεταξύ στρωμάτων του ίδιου επιπέδου (Εικόνα 1.2). Η βελτίωση στον



Σχήμα 1.2: Xilinx Virtex-7 με SSIT τεχνολογία [2].

αριθμό λογικών στοιχείων στα 2.5D FPGAs σε σύγκριση με τις συμβατικές πλατφόρμες είναι πολύ σημαντική. Για παράδειγμα, το μεγαλύτερο interposer-based FPGA, το Virtex-7 XC7V2000T, έχει 4 στρώματα με 1.954 εκατομμύρια στοιχεία λογικής, ενώ το μεγαλύτερο non-interposer Virtex-7 (το XC7VX980T), έχει 979k στοιχεία λογικής και μεγαλύτερο FPGA της Altera, το Stratix B 5SGXBB, έχει 952k στοιχεία λογικής. Όλα τα αναφερόμενα FPGAs χρησιμοποιούν 28nm τεχνολογία, αλλά λόγω των SSIT τα 2.5D FPGAs έχουν δύο φορές περισσότερους πόρους [22].

1.2.3 Απεικόνιση εφαρμογών στον χρόνο εκτέλεσης.

Καθώς η πολυπλοκότητα των κυκλωμάτων και των FPGAs αυξάνεται, η αποτελεσματικότητα και η αποδοτικότητα των CAD εργαλείων που χρησιμοποιούνται γίνεται ακόμη πιο σημαντική. Η πλήρης διαδικασία απεικόνισης για μεγάλα FPGAs έχει τόσο μεγάλη διάρκεια ώστε να μπορεί να κρατήσει ένα σημαντικό μέρος της ημέρας για να εκτελεστεί, ή ακόμη χειρότερα να δηλώσει αποτυχία λόγω κάποιου λάθους.

Μέχρι τώρα, το έργο της απεικόνισης κυκλωμάτων σε FPGAs εφαρμόζεται κυρίως κατά τη διάρκεια της φάσης του σχεδιασμού ενός έργου, δεδομένου ότι οι αλγόριθμοι CAD επιβάλουν μεγάλους χρόνους εκτέλεσης. Τεχνικές που επιταχύνουν CAD αλγόριθμους μπορούν να επιφέρουν σημαντικές αλλαγές στο σχεδιασμό του προϊόντος, ενώ πολλοί σχεδιαστές μπορεί να είναι πρόθυμοι να θυσιάσουν την ποιότητα της λύσης για ένα βελτιωμένο χρόνο εκτέλεσης των εργαλείων. Προς αυτή την κατεύθυνση, ένας αριθμός αλγορίθμων έχουν προταθεί τελευταία χρόνια, που εκτείνονται από την διαδικασία της τοποθέτησης [23], δρομολόγησης [24], καθώς και FPGA προγραμματισμού [25].

Μια άλλη πρόκληση για αυτούς τους αλγορίθμους έχει να κάνει με τον κατακερματισμό της συσκευής, ένα πρόβλημα που εισάγει περιορισμούς στην απόδοση μελλοντικών εφαρμογών. Το πρόβλημα αυτό γίνεται πολύ πιο έντονο για FPGAs υψηλής πυκνότητας δεδομένου ότι με την πάροδο του χρόνου, καθώς φορτώνονται και αφαιρούνται εφαρμογές, οι πόροι του υλικού γίνονται όλο και πιο κατακερματισμένοι. Οι υπάρχουσες προσεγγίσεις έχουν ως στόχο να προσδιορίσουν την κατάλληλη περιοχή πάνω στην αρχιτεκτονική που έχει μία επαρκή ποσότητα συνεχόμενων μη χρησιμοποιούμενων πόρων υλικού, πριν από την τοποθέτηση μιας νέας διαμόρφωσης [26]. Ακόμη και αν οι λύσεις αυτές επιτρέπουν στους σχεδιαστές να προσαρμόζουν την λειτουργικότητα των FPGAs υπό τους περιορισμούς του χρόνου εκτέλεσης, οι εφαρμογές εξετάζονται ως προ-απεικονισμένα μακρο-μπλοκ, με σταθερό πλάτος και ύψος, με αποτέλεσμα την αύξηση του ποσοστού κατακερματισμού της συσκευής. Για το σκοπό αυτό, οι σχετικές προσεγγίσεις ενσωματώνουν κυρίως τεχνικές που στοχεύουν στην ανακατανομή bitstream [27].

1.2.4 FPGAs σε περιβάλλοντα cloud

Οι υπάρχουσες εφαρμογές επιβάλλουν συνεχώς αυξανόμενη ζήτηση για υπολογιστική ισχύ. Η τάση αυτή επηρεάζει όχι μόνο την επιστημονικές και βιομηχανικές εφαρμογές αλλά και καταναλωτές / τελικούς χρηστες. Αυτό έχει οδηγήσει σε νέες στρατηγικές και μεθοδολογίες ειδικά για την εκμετάλευση μαζικά παράλληλων και ετερογενών συστημάτων, όπως συστάδες, cloud και φάρμες από CPU, ή / και GPUs κλπ. Χαρακτηριστικό της τάσης αυτής είναι επιχειρήσεις όπως η Amazon [28] (Amazon Elastic Compute Cloud) που ενοικιάζουν εικονικούς υπολογιστικούς πόρους, όχι μόνο στον κλάδο της βιομηχανίας αλλά και σε απλούς καταναλωτές.

Πρόσφατα Field Programmable Gate Arrays (FPGAs) έχουν διερευνηθεί ως μια βιώσιμη πλατφόρμα για διάφορες εφαρμογές HPC, κυρίως λόγω των χαρακτηριστικών του εγγενούς παραλληλισμού και του επανα-προγραμματισμού που μπορεί να εφαρμοστεί είτε στο χρόνο σχεδιασμού είτε το χρόνο εκτέλεσης. Υπάρχει μεγάλο ενδιαφέρον να χρησιμοποιηθούν πολλαπλά FPGAs ως επιταχυντές υλικού, πλατφόρμες προτυποποίησης και οι μονάδες επεξεργασίας [29], [30], [31].

Οι FPGA επιταχυντές υπόσχονται ελκυστικούς συμβιβασμούς μεταξύ απόδοσης και κατανάλωσης ισχύος, ένα σημαντικό πλεονέκτημα, δεδομένου ότι η επόμενη γενιά συστημάτων HPC αναμένεται να υπερβαίνει τα 10 MW [32]. Επιπλέον, ένα σημαντικό χαρακτηριστικό αυτών των επιταχυντών είναι το υψηλό εύρος ζώνης μνήμης που έχει σημαντικές επιπτώσεις μακροπρόθεσμα στην απόδοση των εφαρμογών. Επιπλέον, runtime επαναδιαμορφούμενα FPGAs μπορούν να υποστηρίξουν πλήθος νέων λειτουργιών

όπως απομακρυσμένη πρόσβαση, δυναμική αυξομείωση της υπολογιστικής ικανότητας και εξισορρόπηση φόρτου εργασίας, που είναι σημαντικά χαρακτηριστικά του cloud computing. Από την άλλη πλευρά τα FPGAs χαρακτηρίζονται από μειωμένη παραγωγικότητα, δεδομένου ότι ασυμβατότητες μεταξύ των συσκευών και εργαλείων εμποδίζουν τη φορητότητα των εφαρμογών και της επαναχρησιμοποίησης μονάδων.

Το πρόβλημα αυτό επιδεινώνεται σημαντικά αν υποθέσουμε ένα σύστημα με πολλαπλές διαφορετικές FPGA πλατφόρμες, ή την μετανάστευση εργασιών από μια συσκευή σε μια άλλη. Υπάρχουν αρκετές προσεγγίσεις που καθιστούν δυνατή τη φορητότητα των εφαρμογών μεταξύ διαφορετικών πλατφορμών FPGA, επιτρέποντας έτσι την ταχεία μοντελοποίηση, την ανάπτυξη και την επαλήθευση. Στο [33] ένα εικονικό πλαίσιο FPGA έχει αναπτυχθεί κυρίως για την εξερεύνηση μιας ενιαίας εικονικής διεπαφής όπου τα πρωτόκολλα επικοινωνίας μετατρέπονται στον χρόνο σχεδίασης σε πραγματικά φυσικά πρωτόκολλα FPGA. Ομοίως, ένα εικονικό στρώμα FPGA που απεικονίζεται επάνω σε ένα πραγματικό FPGA και έτσι υποστηρίζονται διάφορα χαρακτηριστικά, συζητείται στο [34].

Μεθοδολογίες και εργαλεία που υποστηρίζουν την αποτελεσματική αξιοποίηση πολλαπλών FPGAs, με έμφαση κυρίως σε μοντέλα προγραμματισμού και τυποποίησης, έχουν διερευνηθεί ευρέως. Στην εργασία [35] οι συγγραφείς προτείνουν ένα μοντέλο και παρέχουν μια βιβλιοθήκη που έχει ως στόχο την διαχείριση μνήμης μεταξύ πολλών FPGAs καθώς και μεταξύ FPGAs και επεξεργαστών. Η πρόβλεψη της απόδοσης μιας εφαρμογής όταν παραλληλοποιείται σε πολλαπλά FPGAs είναι μία άλλη περιοχή ενδιαφέροντος και έχει ερευνηθεί σε μια σειρά από εργασίες όπως [36]. Και οι δύο εργασίες επιτρέπουν την αποτελεσματική ανάπτυξη παράλληλων εφαρμογών που θα χαρτογραφηθούν σε πολλαπλά FPGAs.

Μια κρίσιμη μετρική σε κάθε run-time εργαλείο είναι ο χρόνος εκτέλεσης. Τα περισσότερο χρονοβόρα βήματα κατά την απεικόνιση μιας εφαρμογής σε ένα FPGA είναι η τοποθέτηση και δρομολόγηση (P&R). Διάφορες ερευνητικές εργασίες έχουν προτείνει λύσεις για την αντιμετώπιση αυτού του προβλήματος [37], [38], [26], [39]. Πιο συγκεκριμένα στην [38] οι συγγραφείς εξερευνούν την απεικόνιση στον χρόνο εκτέλεσης ενός προ-τοποθετημένου και προ-δρομολογημένου σχεδίου πάνω στην αρχιτεκτονική. Από την άλλη πλευρά οι [37] και [39] ασχολούνται με την γρήγορη τοποθέτηση και δρομολόγηση, αντίστοιχα. Οι προαναφερθείσες λύσεις αν και μειώνουν σημαντικά την επιβάρυνση στην απεικόνιση μιας εφαρμογής δεν μπορούν να κλιμακωθούν για να φιλοξενήσουν πολλαπλές αιτήσεις.

1.2.5 FPGAs σε ενσωματωμένα περιβάλλοντα

Ένας αριθμός από στρατηγικές σχεδιασμού και μεθοδολογιών έχουν προταθεί που λαμβάνουν υπόψη το πρόσθετο πλεονέκτημα της ευελιξίας που προσφέρουν τα ετερογενή συστήματα, που αποτελούνται από επεξεργαστές γενικής χρήσης και επιταχυντές υλικού. Τα συστήματα αυτά εκτείνονται από data centers και cloud μέχρι μικρά ενσωματωμένα συστήματα, όπως κινητά τηλέφωνα, έξυπνες τηλεοράσεις, ρολόγια και κάμερες.

Η βιομηχανική και ακαδημαϊκή κοινότητα έχουν διερευνήσει τα Field Programmable Gate Arrays (FPGAs) ως μια βιώσιμη πλατφόρμα για διάφορες εφαρμογές, κυρίως λόγω των χαρακτηριστικών του εγγενούς παραλληλισμού και του επανα-προγραμματισμού που μπορεί να εφαρμοστεί είτε στο χρόνο σχεδιασμού είτε το χρόνο εκτέλεσης. Η χρησιμοποίηση FPGAs ως επιταχυντές υλικού ή / και μονάδες επεξεργασίας είναι μια κοινή πρακτική σε διάφορους τομείς ενσωματωμένων συστημάτων. Παραδείγματα αυτής της τάσης μπορεί να βρεθεί στο dataflow computing center της εταιρίας Maxeler Technologies [40], σε high-end μοντέλα φωτογραφικών μηχανών της εταιρείας Sigma [41] και στο έργο $apertus$ [42] που στοχεύει να παραδώσει μία open-source επαγγελματική μηχανή κινηματογράφου.

Προκειμένου να ενταχθούν αποτελεσματικά σε αυτό το νέο τοπίο, τα FPGAs πρέπει να υποστηρίζουν ταχεία ανάπτυξη και υλοποίηση εφαρμογών. Η βιομηχανία έχει λάβει μέτρα για την ταχύτερη ανάπτυξη εφαρμογών, εξερευνώντας διάφορες λύσεις. Παραδείγματα αυτών των λύσεων μπορεί να βραθεί στα εργαλεία EDA από κορυφαίες εμπορικές εταιρείες FPGA, όπως η Xilinx που έχει ενσωματώσει ένα περιβάλλον High Level Synthesis (HLS) στο εργαλείο της Vivado [16] και η Altera που υποστηρίζει OpenCL [43] πυρήνες. Για την ταχύτερη υλοποίηση εφαρμογών το κύριο σώμα της έρευνας επικεντρώνεται σε ταχύτερους αλγόριθμους και εργαλεία απεικόνισης εφαρμογών.

Οι Computer-aided design (CAD) αλγόριθμοι έχουν μεγάλο χρόνο εκτέλεσης γενικά και έτσι περιορίζουν το έργο της υλοποίησης εφαρμογής πάνω FPGAs στη φάση του σχεδιασμού. Τεχνικές που επιταχύνουν τους βασικούς αλγόριθμους μπορούν να επιφέρουν σημαντικές αλλαγές στο σχεδιασμό ενός προϊόντος. Η πιο εντατική υπολογιστική εργασία κατά τη διάρκεια της απεικόνισης μιας εφαρμογής σε FPGA, είναι το βήμα τοποθέτησης και δρομολόγησης (P&R). Προς αυτήν την κατεύθυνση, ένας αριθμός από αλγόριθμους έχουν προταθεί τα τελευταία χρόνια, όπου ασχολούνται με την γρήγορη τοποθέτηση [44], [45], [46], [47], δρομολόγηση [48], [49] και τον προγραμματισμό FPGA [25].

Η πιο εντατική υπολογιστική εργασία κατά τη διάρκεια της απεικόνισης μιας εφαρμογής πάνω σε ένα FPGA, είναι η τοποθέτηση και δρομολόγηση (P&R). Για να ξεπεραστεί αυτό το εμπόδιο ερευνητές έχουν ήδη προτείνει μια σειρά από λύσεις [50], [51], [52]. Οι

συγγραφείς στην [50] έχουν αναπτύξει έναν παράλληλο placer που βασίζεται σε έναν simulated annealing αλγόριθμο, προκειμένου να μειωθεί ο χρόνος εκτέλεσης. Οι [51] και [52], ενσωματώνουν γνωστές τεχνικές από το πεδίο των Application Specific Integrated Systems (ASICs), προκειμένου να μειωθεί ο χρόνος εκτέλεσης της τοποθέτησης. Είναι σημαντικό να σημειωθεί ότι οι επαναδιαμορφούμενες πλατφόρμες στους [50] και [51] είναι ρεαλιστικά ετερογενή FPGAs που αποτελούνται από λογική, DSP, μνήμη και I/O μπλοκ.

Μια άλλη προσέγγιση για την υποστήριξη γρήγορης απεικόνισης εφαρμογών βασίζεται για την επαναδιαμόρφωση μερών του FPGA κατά τον χρόνο εκτέλεσης για να αλλάξει μια ήδη υλοποιημένη εφαρμογή ή να αντικατασταθεί από μία άλλη. Η έρευνα σε αυτό το θέμα αυτό στοχεύει συνήθως στο να εντοπιστεί μία κατάλληλη περιοχή πάνω στην αρχιτεκτονική, με επαρκή ποσότητα πόρων υλικού, πριν από την τοποθέτηση ενός νέου bitstream [34], [38], [53], [54], [55], [56]. Ένα σχετικό πρόβλημα έχει να κάνει με την περίπτωση όπου δεν είναι δυνατόν να προσδιοριστεί μια τέτοια περιοχή πάνω στην αρχιτεκτονική. Αλγόριθμοι που ασχολούνται με αυτό το πρόβλημα εκτελούν αναδιάταξη των πόρων υλικού [26], [57]. Δυστυχώς, αυτοί οι αλγόριθμοι ανακατανομής πόρων εφαρμόζονται σχεδόν αποκλειστικά κατά το χρόνο σχεδίασης (off-line), λόγω των περιορισμών όπως η αυξημένη υπολογιστική προσπάθεια και τα data hazard των δεδομένων κατά τη μεταφορά της εφαρμογής.

1.3 Επισκόπηση Της Διδακτορικής Διατριβής

- Στο κεφάλαιο 1 παρουσιάζονται οι αρχιτεκτονικές και τα CAD toolflow των FPGAs. Στη συνέχεια, αναφέρονται οι σχετικές εργασίες που αφορούν τις προκλήσεις του ερευνητικού πεδίου των FPGA στα πλαίσια της παρούσας διατριβής. Πιο συγκεκριμένα αναλύονται οι προκλήσεις και το state-of-the-art για ετερογενείς και 3D αρχιτεκτονικές.

- Στο Κεφάλαιο 2 παρέχονται οι συνεισφορές της παρούσας διατριβής μαζί με περαιτέρω διευκρίνιση των προκλήσεων και των προτεινόμενων λύσεων.

- Στο κεφάλαιο 3 παρουσιάζεται μία νέα μεθοδολογία για τη γρήγορη και ακριβή διερεύνηση των οργανώσεων μνήμης σε συσκευές FPGA. Η προτεινόμενη μεθοδολογία υποστηρίζεται από ένα λογισμικό πλαίσιο, που ονομάζεται NAROUTO.

- Στο κεφάλαιο 4, παρουσιάζεται μια νέα αρχιτεκτονική 3-D FPGA, καθώς και το λογισμικό πλαίσιο που υποστηρίζει την εξερεύνηση και την αξιολόγηση της. Πιο συγκεκριμένα, οι πόροι λογικής, μνήμης και I/O ανατίθενται σε διαφορετικά στρώματα δημιουργώντας με αυτόν τον τρόπο ένα 3D FPGA με ετερογενή στρώματα.

- Στο Κεφάλαιο 5 παρουσιάζεται μια μεθοδολογία, καθώς και τα κατάλληλα εργαλεία, για τη γρήγορη υλοποίηση εφαρμογών σε επαναδιαμορφούμενες αρχιτεκτονικές με τη χρήση ενός Just-in-time (JIT) πλαισίου απεικόνισης.

- Στο κεφάλαιο 6 παρουσιάζεται μία μεθοδολογία καθώς και το λογισμικό πλαίσιο, που στοχεύουν σε αποδοτική απεικόνιση εφαρμογών σε FPGAs. Το εν λόγω πλαίσιο είναι πλήρως κλιμακούμενο υποστηρίζοντας παράλληλη απεικόνιση εφαρμογών.

- Στο κεφάλαιο 7 προτείνονται μια νέα μεθοδολογία και το αντίστοιχο εργαλείο ροής για την πραγματοποίηση ταχείας απεικόνισης εφαρμογών σε ενσωματωμένα συστήματα. Μια ετερογενής πλατφόρμα FPGA θεωρείται ως ένα σύνολο πόρων υλικού, συμπεριλαμβανομένων μπλοκ λογικής, μνήμης και DSP, όπου οι εφαρμογές μπορούν να απεικονιστούν δυναμικά. Η έμφαση αυτού του toolflow είναι στην ταχύτητα εκτέλεσης και την χαμηλή χρήση μνήμης, προκειμένου να εκτελεστεί σε ενσωματωμένα συστήματα.

- Στο κεφάλαιο 8 τα συμπεράσματα αυτής της διδακτορικής διατριβής παρουσιάζονται συνοπτικά μαζί με τις συνεισφορές στον τομέα των Field Programmable Gate Arrays.

- Τέλος, στο κεφάλαιο 9 αναφέρονται οι μελλοντικές κατευθύνσεις για την επέκταση του παρόντος ερευνητικού έργου.

I intend to live forever, or die trying.

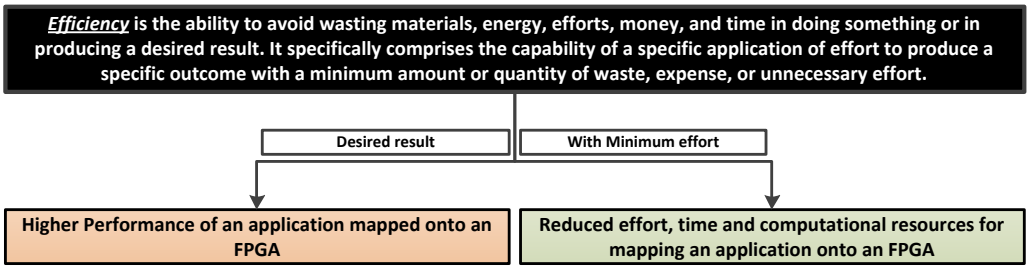
Groucho Marx

Κεφάλαιο 2

Συνεισφορά

2.1 Εισαγωγή

Το θέμα της παρούσας διδακτορικής διατριβής είναι ο σχεδιασμός αρχιτεκτονικών και η απεικόνιση εφαρμογών σε επαναδιαμορφούμενες πλατφόρμες με εργαλεία λογισμικού. Στόχος είναι να αναλυθούν και να σκιαγραφηθούν τα εμπόδια που περιορίζουν την αποδοτικότητα της διαδικασίας της απεικόνισης και να προταθούν νέες λύσεις που άρουν αυτούς τους περιορισμούς.



Σχήμα 2.1: Ο ορισμός της αποδοτικότητας στα πεδία της παρούσας διδακτορικής διατριβής.

Στα πλαίσια της παρούσας διδακτορικής διατριβής, για τον προσδιορισμό της αποδοτικότητας (Σχήμα 2.1) το επιθυμητό αποτέλεσμα είναι η επίτευξη υψηλότερων επιδόσεων για εφαρμογές απεικονισμένες επάνω σε ένα FPGA με καταβολή της ελάχιστης προσπάθειας κατά την διαδικασία της απεικόνισης (mapping). Ελάχιστη προσπάθεια στην διαδικασία της απεικόνισης μεταφράζεται σε ελάχιστο χρόνο στο σχεδιασμό, καλύτερους συμβιβασμούς στο χρόνο εκτέλεσης σε σχέση με ποιότητα της λύσης και ήπια χρησιμοποίηση υπολογιστικών πόρων.

Οι συνεισφορές που αφορούν την αποδοτικότητα της διαδικασίας της απεικόνισης σε FPGAs συνοψίζονται ως εξής:

Συνεισφορές Αρχιτεκτονικού Επιπέδου

- *3D Αρχιτεκτονική FPGA με ετερογενή στρώματα.* Επιτυγχάνεται υψηλότερη απόδοση εφαρμογών υλοποιημένων σε ένα FPGA.
- *Ταχεία διερεύνηση ιεραρχιών μνήμης σε 2D FPGAs.* Μειώνεται η προσπάθεια για την επιλογή της κατάλληλης αρχιτεκτονικής FPGA για να απεικονιστεί μια εφαρμογή.

Συνεισφορές σε επίπεδο CAD εργαλείων

- *JiT απεικόνιση εφαρμογών σε FPGA.* Επιτυγχάνεται ανώτερη απόδοση της εφαρμογής που έχει απεικονιστεί επάνω σε ένα FPGA και μειωμένο χρόνο εκτέλεσης για την απεικόνιση πολλαπλών εφαρμογών σε ένα FPGA.
- *Απεικόνιση εφαρμογών σε FPGAs κατά τον χρόνο εκτέλεσης σε περιβάλλον cloud.* Επιτυγχάνεται ανώτερη απόδοση της εφαρμογής που έχει απεικονιστεί επάνω σε ένα FPGA και μειωμένο χρόνο εκτέλεσης για την απεικόνιση πολλαπλών εφαρμογών σε ένα FPGA.
- *Απεικόνιση εφαρμογών σε FPGAs κατά τον χρόνο εκτέλεσης σε περιβάλλον ενσωματωμένων συστημάτων.* Επιτυγχάνεται ανώτερη απόδοση της εφαρμογής που έχει απεικονιστεί επάνω σε ένα FPGA, μειωμένο χρόνο εκτέλεσης και μειωμένες απαιτήσεις σε υπολογιστικούς πόρους για την απεικόνιση πολλαπλών εφαρμογών σε ένα FPGA.

Οι προκλήσεις και οι προτεινόμενες λύσεις αναλύονται στην επόμενη ενότητα.

2.2 Προκλήσεις και οι προτεινόμενες λύσεις σε επίπεδο αρχιτεκτονικής

Πρόκληση I

Οι εμπορικές λύσεις δεν προσαρμόζονται εύκολα για να αξιολογηθούν FPGA αρχιτεκτονικές που διαφέρουν από τις ήδη κατασκευασμένες συσκευές, ενώ οι ακαδημαϊκές προσεγγίσεις δεν λαμβάνουν συνήθως υπόψη τους εγγενείς περιορισμούς που τίθενται από ετερογενή μπλοκ που συναντώνται σε σύγχρονες πλατφόρμες. Οι κύριοι περιορισμοί που έχουν τα υπάρχοντα εργαλεία για την υποστήριξη της εξερεύνησης σε επίπεδο αρχιτεκτονικής, καθώς και την απεικόνιση εφαρμογών σε FPGAs που αποτελούνται από ετερογενή στοιχεία είναι τα ακόλουθα:

- Η λειτουργικότητα της εφαρμογής που περιγράφεται στα ακαδημαϊκά εργαλεία σε BLIF netlist διαφέρει από την RTL περιγραφή της εφαρμογής, δεδομένου ότι τα

BlackBoxes που βρίσκονται σε αυτή την μορφή BLIF δεν περιέχουν καμία λειτουργική πληροφορία.

- Οι μνήμες δεν μπορούν να οργανωθούν με έναν ιεραρχικό τρόπο, αλλά θεωρούνται μόνο ως προκαθορισμένα στοιχεία.
- Οι υπάρχουσες προσεγγίσεις δεν μπορούν να υποστηρίξουν την αξιολόγηση αρχιτεκτονικών επιλογών που βασίζονται σε διαφορετικές οργανώσεις μνήμης ή/και ιεραρχίες.

Λύση I

Οι συνεισφορές της παρούσας διατριβής, για την αντιμετώπιση των προαναφερθέντων προβλημάτων συνοψίζονται ως εξής:

- Εισαγωγή μιας νέας μεθοδολογίας και του αντίστοιχου λογισμικού πλαισίου που επιτρέπει την ταχεία εξερεύνηση σε επίπεδο αρχιτεκτονικής για ετερογενή FPGAs που αποτελούνται από διαφορετικές οργανώσεις ή / και ιεραρχίες μνήμης.
- Ανάπτυξη ενός νέου λογισμικού πλαισίου που επιτρέπει την απεικόνιση των εφαρμογών πάνω σε αυτά τα ετερογενή FPGAs.
- Το προτεινόμενο πλαίσιο επιτρέπει, μεταξύ άλλων, χειρισμό και άλλων ετερογενών μπλοκ με διαφορετικές ιεραρχίες.

Πρόκληση II

Η πολυπλοκότητα των σημερινών εφαρμογών εισάγει συνήθως περιορισμούς στην αρχιτεκτονική οργάνωση των FPGA. Ακόμη και αν η ζήτηση για επιπλέον λογικούς πόρους ικανοποιείται με πλατφόρμες που αποτελούνται από πιο πολύπλοκα λογικά μπλοκ, ή CLBs, (π.χ. με περισσότερα LUTs), το πρόβλημα αυτό εξακολουθεί να υφίσταται με τις πιο απαιτητικές σε θέμα επικοινωνίας εφαρμογές (π.χ. τηλεπικοινωνίες, κρυπτογράφηση και την επεξεργασία εικόνας, βίντεο), δεδομένου ότι η απόδοσή τους εξαρτάται συνήθως από τη διαθεσιμότητα σε I/O bandwidth. Προηγούμενες μελέτες [58] έχουν ήδη επισημάνει ότι η επικοινωνία μεταξύ λογικής και μπλοκ μνήμης βελτιώνεται σημαντικά με τη χρήση μιας τρισδιάστατης αρχιτεκτονικής. Ως εκ τούτου, τα μπλοκ μνήμης πρέπει να τοποθετηθούν σε διαφορετικό στρώμα, από ότι η υπόλοιπη λογική υποδομή. Ωστόσο, σε καμία από αυτές τις εργασίες δεν έχουν μελετηθεί έως τώρα οι επιπτώσεις της συμφόρησης στην επικοινωνία μεταξύ I/O μπλοκ και της υπόλοιπης αρχιτεκτονικής.

Λύση II

Στην παρούσα διατριβή, προκειμένου να μειωθεί η συμφόρηση στην επικοινωνία προτείνεται ένα νέο αρχιτεκτονικό πρότυπο για τον σχεδιασμό ετερογενών 3-D FPGAs με στρώματα που περιέχουν μπλοκ υλικού διαφορετικού τύπου. Πιο συγκεκριμένα, εξετάζεται η περίπτωσή μιας 3-D αρχιτεκτονικής που αποτελείται από τρία στρώματα, όπου η λογική, η μνήμη και τα I/O μπλοκ είναι τοποθετημένα σε διαφορετικά στρώματα. Επιπλέον, έχουμε εισαγάγει ένα νέο λογισμικό πλαίσιο που υποστηρίζει την εξερεύνηση σε επίπεδο αρχιτεκτονικής, καθώς και την διαδικασία απεικόνισης μιας εφαρμογής πάνω σε αυτά τα 3-D FPGAs. Ακόμα κι αν σε αυτή την ερευνητική εργασία μελετάμε 3-D FPGAs που αποτελούνται από τρία στρώματα, τα προτεινόμενα CAD εργαλεία και η μεθοδολογία υποστηρίζουν επίσης οποιαδήποτε άλλη 3-D αρχιτεκτονική (π.χ. με περισσότερα στρώματα ή που περιέχουν διαφορετικούς τύπους μπλοκ).

2.3 Προκλήσεις και λύσεις σε επίπεδο CAD εργαλείων

Πρόκληση III

Ένα βασικό χαρακτηριστικό των FPGAs, όπως προαναφέρθηκε είναι η δυνατότητα επαναπρογραμματισμού τους. Στις σύγχρονες FPGA πλατφόρμες μπορούμε να βρούμε αυτήν τη λειτουργία ενεργοποιημένη, ακόμη και για το χρόνο εκτέλεσης. Αν και αυτό δίνει νέες ευκαιρίες και ευελιξία στους σχεδιαστές, δυστυχώς η διαδικασία της απεικόνισης είναι αγκιστρωμένη στον χρόνο σχεδίασης. Αυτό συμβαίνει κυρίως λόγω του εκτεταμένου χρόνου εκτέλεσης των εργαλείων που μπορεί να φτάσει ακόμα και μέρες. Οι μόνες λύσεις που είχαν προταθεί μέχρι τώρα περιλαμβάνουν πάντα την εκτέλεση πολλών βημάτων κατά τον χρόνο σχεδιασμού, προκειμένου να ελαχιστοποιηθεί η προσπάθεια που απαιτείται στον χρόνο εκτέλεσης. Η στρατηγική αυτή επιβάλλει σημαντικούς περιορισμούς στην τελική απεικόνιση της εφαρμογής και προκαλεί κατακερματισμό των πόρων του FPGA.

Λύση III

Για την αντιμετώπιση αυτών των περιορισμών σε αυτή τη διατριβή, εισάγεται μια νέα μεθοδολογία με στόχο να πραγματοποιηθεί γρήγορη απεικόνιση εφαρμογών σε FPGAs. Ο στόχος αυτής της προσέγγισης είναι να μειωθεί σημαντικά ο χρόνος εκτέλεσης με την ελάχιστη δυνατή υποβάθμιση των επιδόσεων της εφαρμογής. Προς αυτή την κατεύθυνση και σε αντίθεση με άλλες σύγχρονες προσεγγίσεις που οι εφαρμογές προτοποθετούνται και προ-δρομολογούνται, προτείνεται η χρήση ενός Just-In-Time (JIT) πλαισίου απεικόνισης. Συγκεκριμένα, τα δεδομένα διαμόρφωσης του FPGA υπολογίζονται κατά το χρόνο εκτέλεσης εκτελώντας τοποθέτηση, δρομολόγηση και παραγωγή bitstream της εφαρμογής.

Πρόκληση IV

Οι FPGA επιταχυντές υπόσχονται έναν ελκυστικό συνδυασμό μεταξύ απόδοσης και κατανάλωσης ισχύος για συστήματα cloud και HPC, αλλά χαρακτηρίζονται από μειωμένη παραγωγικότητα, δεδομένου ότι υπάρχουν ασυμβατότητες μεταξύ συσκευών και ροών εργαλείων που εμποδίζουν την μεταφορά εφαρμογών και επαναχρησιμοποίησης κομματιών. Μια προϋπόθεση για να χρησιμοποιηθούν FPGAs σε ένα περιβάλλον cloud θα ήταν να έχουν ελάχιστο χρόνο απεικόνισης των εφαρμογών και επεκτασιμότητα για να φιλοξενούν πολλαπλές εφαρμογές. Παρά το γεγονός ότι υπάρχουν λύσεις που μειώνουν σημαντικά την επιβάρυνση σε χρόνο της απεικόνισης μίας εφαρμογής, δεν μπορούν να κλιμακωθούν έτσι ώστε να φιλοξενήσουν πολλαπλές εφαρμογές.

Λύση IV

Σε αυτή την διδακτορική διατριβή προτείνεται μια μεθοδολογία και το λογισμικό πλαίσιο προκειμένου να καταστεί δυνατή η αποτελεσματική απεικόνιση πολλαπλών εφαρμογών κατά το χρόνο εκτέλεσης σε μία μόνο ή πολλαπλές FPGAs. Η προτεινόμενη λύση άρει τα προαναφερθέντα εμπόδια προσφέροντας γρήγορους χρόνους εκτέλεσης και επιτρέποντας φορητότητα εφαρμογών.

Πρόκληση V

Τα υπάρχοντα ενσωματωμένα συστήματα υλοποιούν μια μεγάλη ποικιλία από πυρήνες, συνήθως με ποικίλες λειτουργίες, από συσκευές αναπαραγωγής πολυμέσων μέχρι τηλεπικοινωνιακές πλατφόρμες. Πρέπει να σημειωθεί είναι ότι σε πολλές περιπτώσεις αυτές οι λειτουργίες δεν είναι γνωστές στο χρόνο σχεδίασης, δεδομένου ότι είναι ορίζονται από το χρήστη. Σε αυτά τα σενάρια, όπου ο τελικός χρήστης επιλέγει το λειτουργικότητα και η χρήση των πόρων επεξεργασίας του, υπάρχει μια ανάγκη για πολλαπλούς επιταχυντές -που δεν είναι γνωστοί εκ των προτέρων- να συνυπάρξουν σε ένα FPGA. Σε αυτές τις περιπτώσεις προκύπτουν δύο μεγάλες προκλήσεις: α. Πώς να απεικονιστούν πολλαπλές ανεξάρτητες εφαρμογές και β. Πως να εκτελεστεί η απεικόνιση της εφαρμογής στη συσκευή του τελικού χρήστη.

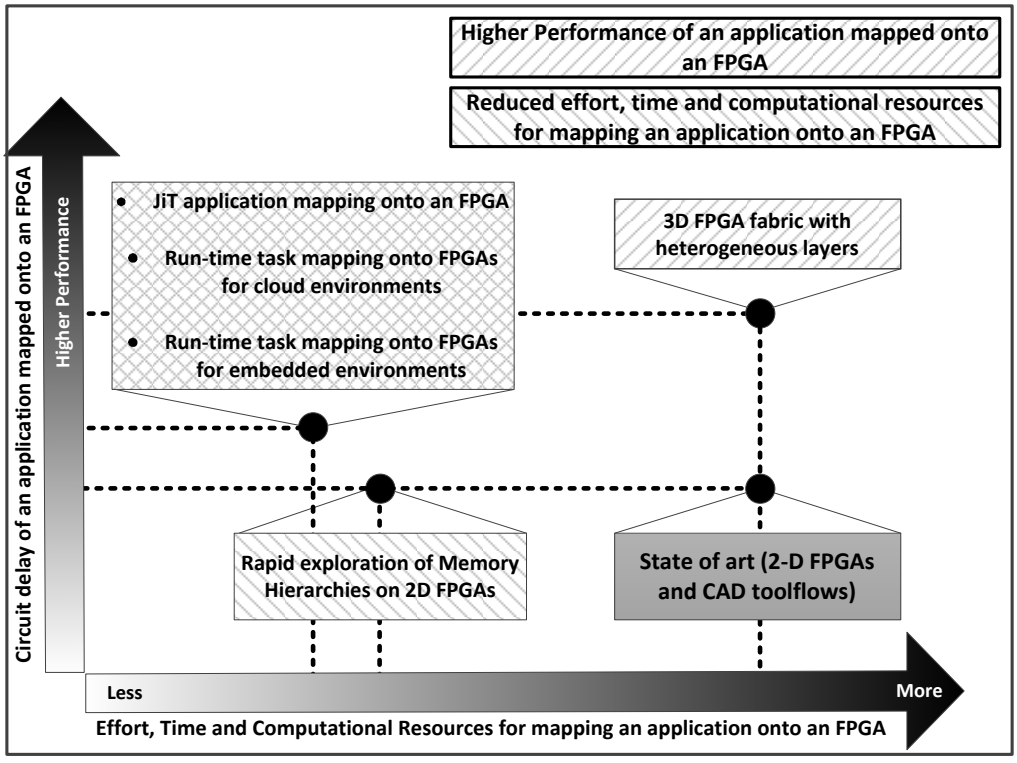
Λύση V

Για την αντιμετώπιση αυτών των προκλήσεων σε αυτήν την διδακτορική διατριβή προτείνεται μια νέα μεθοδολογία και τα κατάλληλα εργαλεία με σκοπό να επιτευχθεί ταχεία απεικόνιση εφαρμογών. Θεωρούμε μία ετερογενή πλατφόρμα FPGA ως ένα σύνολο πόρων υλικού, συμπεριλαμβανομένων λογικών κυκλωμάτων, μνήμης και DSP μπλοκ, όπου οι εφαρμογές μπορούν να απεικονιστούν ως δυναμικοί πυρήνες του υλικού σε αυτούς τους πόρους. Κάθε ένας από αυτούς τους εικονικούς δυναμικούς πυρήνες, που ονομάζονται VKernels, μπορούν να υλοποιήσουν μόνο μία εφαρμογή ενώ πολλαπλοί πυρήνες μπορούν να φορτωθούν σε ένα ενιαίο FPGA. Προκειμένου να καταστεί εφικτό

να φορτωθούν αυτοί οι VKernels κατά το χρόνο εκτέλεσης, ακόμη και για ενσωματωμένα συστήματα τελικού χρήστη, προτείνονται αλγοριθμικές βελτιστοποιήσεις και βελτιστοποιήσεις μνήμης που μειώνουν σημαντικά τόσο τον χρόνο εκτέλεσης όσο και το αποτύπωμα μνήμης.

2.4 Σύνοψη

Όπως αναφέρθηκε στην ενότητα 2.1 η αποδοτικότητα ως ποσοτικός ή ποιοτικός όρος αποτελείται από δύο στοιχεία: το επιθυμητό αποτέλεσμα, που στην περίπτωσή μας είναι η απόδοση της εφαρμογής που θα απεικονιστεί σε FPGA, και την ελάχιστη προσπάθεια, η οποία μπορεί να είναι ο χρόνος ανάπτυξης, εκτέλεσης και χαμηλές υπολογιστικές ανάγκες της διαδικασίας της απεικόνισης.



Σχήμα 2.2: Τα επιτεύγματα κάθε μέρους του διδακτορικού έργου που εκφράζονται ως προς τις επιδόσεις και την προσπάθεια που δαπανάται για την επίτευξη αυτής της απόδοσης.

Η εικόνα 2.2 εκφράζει τις συνεισφορές αυτής της διδακτορικής διατριβής σε σχέση με τα δύο αυτά στοιχεία. Στον άξονα Y όπως ανεβαίνουμε επιτυγχάνονται υψηλότερες επιδόσεις -χαμηλότερη καθυστέρηση ή/και κατανάλωση ενέργειας- ενώ όσο περαιτέρω προχωρούμε στον X άξονα τόσο μεγαλύτερη προσπάθεια καταβάλουν τα εργαλεία για να επιτευχθεί αυτό το επίπεδο απόδοσης. Στην ιδανική περίπτωση μια λύση θα

ήταν όσο το δυνατόν ψηλότερα και πιο κοντά στον Y άξονα. Μια συμβατική 2D FPGA αρχιτεκτονική και τα αντίστοιχα CAD εργαλεία τους θεωρούνται το σημείο αναφοράς.

Οι προτεινόμενες 3D αρχιτεκτονικές FPGA προσφέρουν υψηλότερες επιδόσεις, χωρίς να απαιτείται περισσότερος χρόνος ή υπολογιστικοί πόροι για τα CAD εργαλεία σε σύγκριση με τα 2D CAD εργαλεία. Για να έχουμε μια δίκαιη σύγκριση, η 3D προτεινόμενη αρχιτεκτονική έχει τον ίδιο αριθμό πόρων, όπως η 2D αρχιτεκτονική, μόνο που αυτοί εκτείνονται σε πολλαπλά στρώματα. Η μεθοδολογία για την ταχεία διερεύνηση ιεραρχιών και οργανώσεων μνήμης επηρεάζει αφενός τον χρόνο ανάπτυξης ή επιλογής μιας συσκευής FPGA, και αφετέρου επιτρέπει μια πιο αποτελεσματική χρήση μνήμης μέσω της εξερεύνησης διαφορετικών ιεραρχιών.

Το έργο των ερευνητικών εργασιών σχετικά με τα CAD εργαλεία, επιτρέπει τη δυναμική απεικόνιση εφαρμογών πάνω σε FPGAs, μειώνοντας το χρόνο εκτέλεσης και τους πόρους που απαιτούνται για την ίδια την διαδικασία της απεικόνισης, και προσφέρει υψηλότερη απόδοση σε εφαρμογές που απεικονίζονται ως stand-alone. Έτσι επιτυγχάνει βελτιώσεις στην απόδοση με λιγότερη προσπάθεια από τα εργαλεία.

Κεφάλαιο 3

Δισδιάστατες επαναδιαμορφούμενες αρχιτεκτονικές

3.1 Εισαγωγή

Η λογική δομή μιας FPGA πλατφόρμας έχει αλλάξει σταδιακά από μια ομοιογενή και τακτική αρχιτεκτονική σε μια ετερογενή (ή piece-wise ομογενή) αρχιτεκτονική. Το πρόβλημα της εξερεύνησης σε επίπεδο αρχιτεκτονικής γίνεται πολύ πιο σημαντικό, λόγω της αυξημένης πολυπλοκότητας που δημιουργείται λόγω των ετερογενών IP μπλοκ που βρίσκονται σε σύγχρονες πλατφόρμες FPGA.

3.1.1 Motivation Example

Όπως έχει ήδη αναφερθεί, τα υπάρχοντα λογισμικά πλαίσια (είτε ακαδημαϊκά είτε εμπορικά) δεν είναι ακόμη σε θέση να χειριστούν επαρκώς το έργο της εξερεύνησης σε επίπεδο αρχιτεκτονικής. Πιο συγκεκριμένα, οι εμπορικές λύσεις δεν προσαρμόζονται εύκολα για την αξιολόγηση επαναδιαμορφούμενων αρχιτεκτονικών που διαφέρουν από τις εμπορικές συσκευές, ενώ οι ακαδημαϊκές προσεγγίσεις δεν είναι ενημερωμένες σχετικά με τους εγγενείς περιορισμούς που δημιουργούνται από ετερογενή στοιχεία. Αυτή η ενότητα εστιάζει στους κυριότερους περιορισμούς που βρέθηκαν στα υπάρχοντα εργαλεία για την υποστήριξη της εξερεύνηση σε επίπεδο αρχιτεκτονικής, καθώς και στην απεικόνιση μιας εφαρμογής πάνω σε FPGAs που αποτελείται από ετερογενή στοιχεία.

Ξεκινώντας από την περιγραφή μιας εφαρμογής σε VHDL ή Verilog μορφή, πρώτα απ ’όλα εκτελείται η σύνθεση με τη χρήση του Altera Quartus Framework [6], λαμβάνοντας υπόψη ότι η έξοδος έχει οριστεί σε BLIF (Berkeley Logic Interchange Format) μορφή [59].

Αυτή η μορφή αντιστοιχεί σε ένα netlist επίπεδου πύλης, με βασικά μοντέλα για την αναπαράσταση εισόδου, εξόδου, λογικής πύλης, flip / flops, κτλ. Το BLIF αρχείο αν και είναι μία ευρέως αποδεκτή μορφή στα ακαδημαϊκά εργαλεία, είναι περιοριστική, καθώς δεν είναι σε θέση να εκφράσει ετερογενή αρχιτεκτονικά στοιχεία, όπως μπλοκ μνήμης RAM, μπλοκ DSP, επεξεργαστές, κτλ. Επιπλέον, δεν μπορεί να εκφράσει αριθμητικές carry chains χωρίς την μετατροπή τους σε πύλες.

Αντί αυτών των συστατικών, η BLIF netlist χρησιμοποιεί "BlackBoxes" (BBs) για να υποστηρίξει διαφανή διάδοση του σήματος. Ωστόσο, δεδομένου ότι τα BBs δεν έχουν καμία πληροφορία για την λειτουργικότητα τους, η παραγόμενη netlist υστερεί σε ακρίβεια. Επιπλέον, όπως θα παρουσιαστεί αργότερα, τα υπάρχοντα εργαλεία δεν παρέχουν κάποιον αποδοτικό τρόπο για τη διαχείριση κυκλωμάτων με BBs.

Τα κύρια μειονεκτήματα των υπαρχόντων (ακαδημαϊκών / εμπορικών) λύσεων λογισμικού συνοψίζονται ως εξής:

- Η λειτουργικότητα της εφαρμογής που περιγράφεται σε BLIF netlist διαφέρει από την RTL περιγραφή της, δεδομένου ότι τα BBs δεν παρέχουν καμία λειτουργία.
- Για ένα δεδομένο κύκλωμα, όλα τα BBs σημειώνονται με την ίδια λέξη-κλειδί (".blackbox"), ανεξάρτητα από την πραγματική λειτουργικότητα τους μέσα στο BLIF αρχείο. Αυτό επιβάλλει ότι κάθε εφαρμογή μπορεί να χρησιμοποιήσει μόνο έναν τύπο BB (π.χ. μόνο μνήμη, DSP, ή ενσωματωμένη CPU).
- Επιπλέον, όλα αυτά τα BBs υποτίθεται ότι έχουν τις ίδιες ιδιότητες (π.χ., μέγεθος, απόδοση, την κατανάλωση ενέργειας / ενέργειας, κτλ.), ανεξάρτητα από τη χρήση τους.
- Τελευταίο και σημαντικότερο, οι υπάρχουσες προσεγγίσεις δεν μπορούν να υποστηρίξουν την αξιολόγηση των αρχιτεκτονικών επιλογών που βασίζονται σε διαφορετικές οργανώσεις μνήμης και ιεραρχίες.

Η εργασία αυτή εισάγει μια νέα μεθοδολογία για τη υποστήριξη μιας γρήγορης διερεύνησης των ιεραρχιών μνήμης σε συσκευές FPGA. Η προτεινόμενη μεθοδολογία υποστηρίζεται από ένα λογισμικό πλαίσιο ανοικτού κώδικα, που ονομάζεται NAROUTO. Αυτό το λογισμικό πλαίσιο προσφέρει λύσεις για την αξιολόγηση όσον αφορά την καθυστέρηση, τη δύναμη, την ενέργεια και το εμβαδόν διαφορετικών οργανώσεων και ιεραρχιών μνήμης σε reconfigurable συσκευές. Τα πειραματικά αποτελέσματα αποδεικνύουν την αποτελεσματικότητα της προτεινόμενης λύσης, σε σύγκριση με παρόμοιες προσεγγίσεις. Οι συνεισφορές αυτής της εργασίας, συνοψίζονται ως εξής:

- Η εισαγωγή ενός νέου λογισμικού που υποστηρίζει μια μεθοδολογία για την ταχεία εξερεύνηση αρχιτεκτονικών ετερογενών FPGAs που αποτελούνται από διαφορετικές οργανώσεις μνήμης ή/και ιεραρχίες.

- Ανάπτυξη ενός framework που επιτρέπει την απεικόνιση μιας εφαρμογή σε αυτά τα ετερογενή FPGAs.
- Εκτός από την καθυστέρηση, η εφαρμογή υλοποιημένη σε ετερογενή FPGA αξιολογείται και σε σχέση με την κατανάλωση ενέργειας, ισχύος (στατικής και δυναμικής).

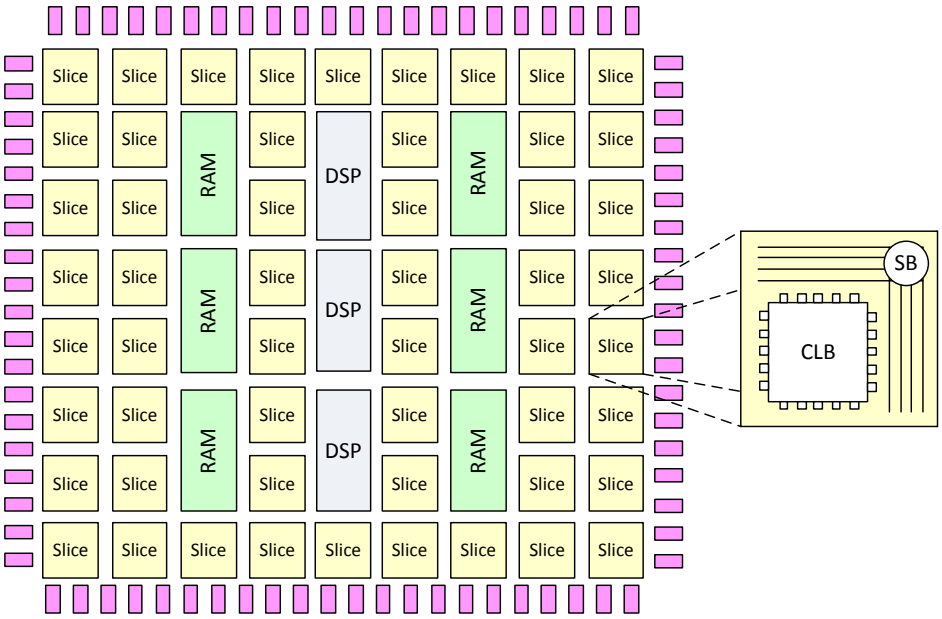
Το υπόλοιπο της εργασίας οργανώνεται ως εξής: η Ενότητα 2 αναδεικνύει τα βασικά μειονεκτήματα που βρέθηκαν σε παρόμοιες προσεγγίσεις με στόχο την εξερεύνηση σε επίπεδο αρχιτεκτονικής, ενώ η ενότητα 3 παρέχει μια επισκόπηση των αρχιτεκτονικών των ετερογενών FPGAs. Η προτεινόμενη μεθοδολογία, καθώς και το προτεινόμενο framework περιγράφονται στις ενότητες 4 και 5, αντίστοιχα. Η ενότητα 6 παρέχει μια σειρά από ποιοτικές και ποσοτικές συγκρίσεις που αποδεικνύουν την αποτελεσματικότητα της προτεινόμενης λύσης, σε σύγκριση με την state-of-art προσέγγιση. Τέλος, τα συμπεράσματα συνοψίζονται στην ενότητα 7.

3.2 Δομή της επαναδιαμορφούμενης αρχιτεκτονικής.

Η δομή της αρχιτεκτονικής που χρησιμοποιήθηκε είναι αυτή ενός κοινού FPGA παρόμοιου με μοντέρνα FPGAs από την Altera (Stratix) [60] και την Xilinx (Virtex) [61], τα οποία αποτελούνται από πόρους λογικής, μνήμης, στοιχεία ειδικού σκοπού (π.χ. Ενσωματωμένους επεξεργαστές, μπλοκ ψηφιακής επεξεργασίας σήματος, κ.τ.λ.) και εισόδους/εξόδους. Τα στοιχεία λογικής μιας FPGA πλατφόρμας είναι οργανωμένα σε έναν x, y πίνακα από τεμάχια υλικού (slices), ενώ η επικοινωνία μεταξύ αυτών των μπλοκ υλικού γίνεται μέσω ενός ιεραρχικού δικτύου γρήγορων και ευέλικτων διασυνδέσεων. Ο όρος slice (τα τεμάχια υλικού), αναφέρεται σε μια διάταξη που αποτελείται από το CLB, τα πάνω και τα δεξιά (από το CLB) στοιχεία διασύνδεσης, καθώς και τον αντίστοιχο μεταγωγέα.

Στο επόμενο προς τα κάτω επίπεδο ιεραρχίας υποθέτουμε ότι το CLB αποτελείται από έναν αριθμό βασικών λογικών μπλοκ (BLEs). Το κάθε BLE με την σειρά του αποτελείται από έναν πίνακα αναζήτησης, , κάποια flip/flops, κάποιους πολυπλέκτες οι οποίοι λειτουργούν ως είσοδοι/έξοδοι, καθώς και τα απαιτούμενα καλώδια για τις τοπικές συνδέσεις. Αυτό το αρχιτεκτονικό μοτίβο επιτρέπει να γίνουν τοπικές βελτιστοποιήσεις μεταξύ των BLEs [5]. Η εικόνα 3.1 απεικονίζει μια πρότυπη αρχιτεκτονική με ενσωματωμένα RAM και DSP μπλοκ [10].

Οι αρχιτεκτονικές παράμετροι των CLBs οι οποίες αναφέρθηκαν προηγουμένως διαφέρουν ανάλογα με τον κατασκευαστή και την οικογένεια στην οποία ανήκει το FPGA. Αυτές οι διαφορές έχουν επίδραση στην επίδοση και στην κατανάλωση ενέργειας της συσκευής. Για παράδειγμα στα Altera Stratix FPGAs ομαδοποιούνται 10 BLEs με σκοπό



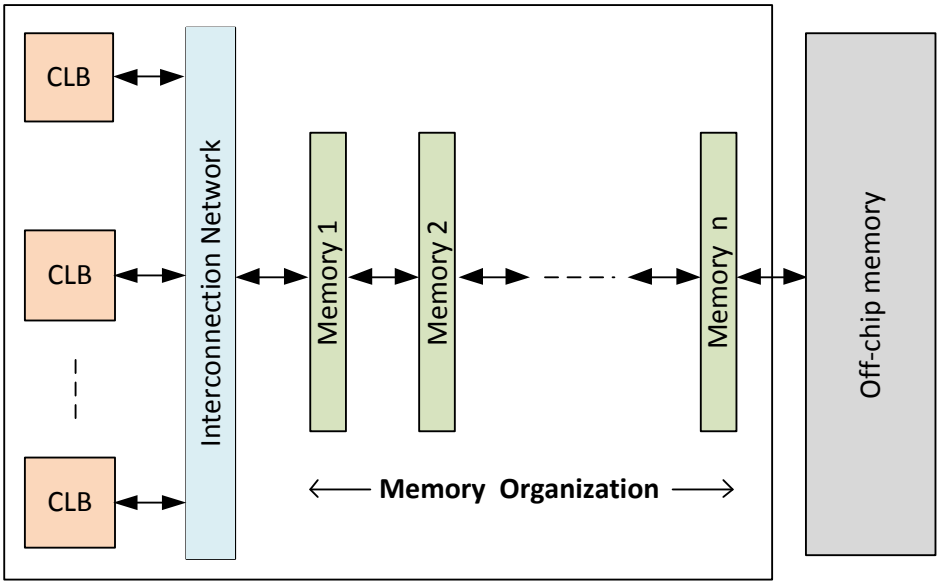
Σχήμα 3.1: Πρότυπο της χρησιμοποιούμενης ετερογενούς FPGA πλατφόρμας.

να σχηματίσουν έναν Logic Array Block (LAB) [60]. Παρομοίως στις συσκευές Xilinx Virtex-II-Pro, δύο LUTs σχηματίζουν ένα BLE, ενώ 4 BLEs ενώνονται για να δημιουργήσουν ένα slice [61].

Εκτός από τις λογικές υποδομές και τις υποδομές διασύνδεσης, μια FPGA αρχιτεκτονική, περιλαμβάνει και ένα αριθμό από ετερογενή μπλοκ. Σε αυτή την ερευνητική εργασία αυτά τα μπλοκ χρησιμοποιούνται με σκοπό να μελετηθεί ο αντίκτυπος διαφορετικών ιεραρχιών μνήμης. Πιο συγκεκριμένα, δύο διαφορετικές προσεγγίσεις, οι οποίες απεικονίζονται στις εικόνες 3.2 και 3.3, αξιολογούνται με την προτεινόμενη μεθοδολογία μέσω του λογισμικού που αναπτύχθηκε. Αυτές οι ιεραρχίες μνήμης συνοψίζονται ακολούθως:

- **Σενάριο 1.** Απεικονίζεται σχηματικά στην εικόνα 3.2 και ασχολείται με την περίπτωση αρχιτεκτονικής κοινής μνήμης. Σε αυτόν τον τύπο οργάνωσης μνήμης, υποθέτουμε ότι διαφορετικά CLBs προσπελαύνουν ένα μεγάλο μπλοκ RAM. Ενώ η διαδικασία της αποτύπωσης μια εφαρμογής σε μια FPGA συσκευή που παρέχει τέτοια οργάνωση μνήμης, είναι σχετικά εύκολη, ένας αριθμός περιορισμών δημιουργείται όταν πολλαπλά CLB χρειάζονται γρήγορη προσπέλαση στην μνήμη. Επιπρόσθετα, μια αρχιτεκτονική με διαμοιραζόμενη μνήμη δεν κλιμακώνεται πολύ αποδοτικά.

- **Σενάριο 2.** Ασχολείται με την διαμοιραζόμενη-διανεμημένη αρχιτεκτονική μνήμης. Αυτή η προσέγγιση, η οποία απεικονίζεται σχηματικά στην εικόνα 3.3, εκτός από έναν αριθμό από διαμοιραζόμενες μνήμες (όπως συζητήθηκε προηγουμένως), ενσωματώνει και έναν μηχανισμό που επιτρέπει κάθε CLB να έχει άμεση πρόσβαση σε μια ιδιωτική μνήμη. Το κύριο πλεονέκτημα μιας τέτοιας αρχιτεκτονικής μνήμης



Σχήμα 3.2: Ένα σχηματικό παράδειγμα της αρχιτεκτονικής διαμοιραζόμενης μνήμης (Σενάριο 1).

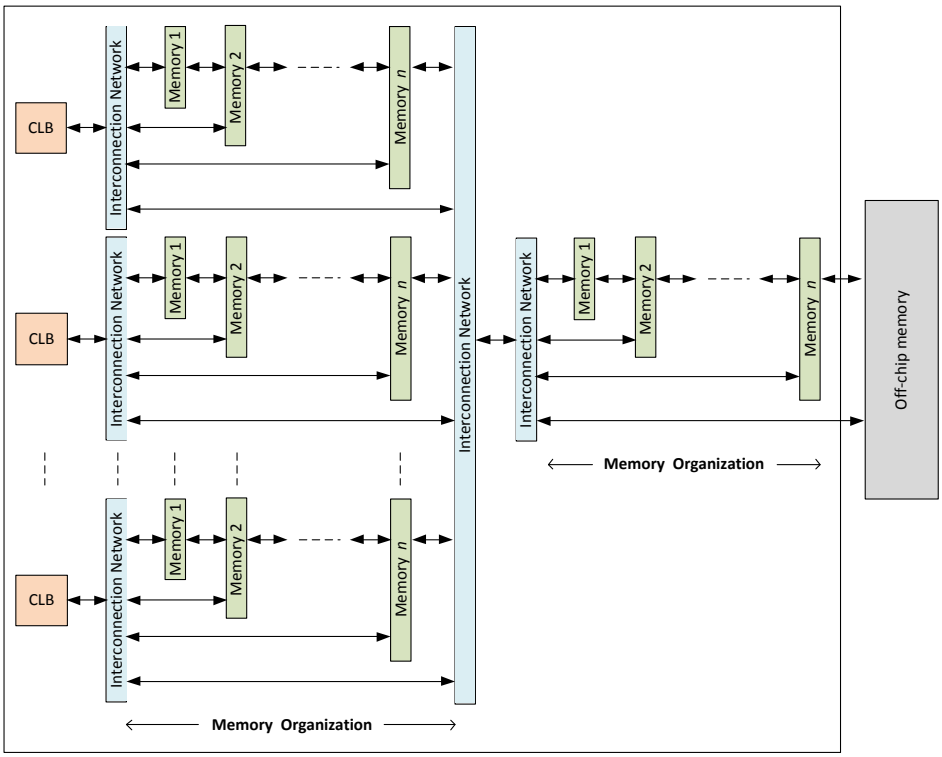
είναι ότι ο χώρος διευθύνσεων όλων των δεδομένων είναι ενοποιημένος. Επιπρόσθετα αυτή η ιεραρχία μνήμης κλιμακώνεται πιο εύκολα ανάλογα με τις ανάγκες της εκάστοτε εφαρμογής.

Και για τα δυο μοντέλα δεδομένων-μνήμης, υποθέτω ότι υπάρχει μια κοινή μνήμη εκτός chip. Καθ' όλη την έκταση της παρούσας εργασίας δεν εξετάζονται θέματα που αφορούν το πως θα απεικονιστούν τα δεδομένα στα μπλοκ μνήμης. Τέτοια θέματα ανακύπτουν και λύνονται στο στάδιο της σύνθεσης και technology mapping. Επίσης και για τις δύο ιεραρχίες, γίνεται η υπόθεση ότι μπορούν ταυτόχρονα να προσπελαστούν από πολλαπλά CLBs. Για να υλοποιηθούν αυτές οι ιεραρχίες, χρησιμοποιείται ένας αριθμός από ειδικού σκοπού κανάλια διασύνδεσης, τα οποία παρέχουν επικοινωνία μεταξύ των μπλοκ μνήμης. Τα χαρακτηριστικά αυτών των καναλιών όσο αναφορά την επίδοση (π.χ. καθυστέρηση, κατανάλωση ενέργειας), έχουν ληφθεί υπόψιν κατά την απεικόνιση της εφαρμογής πάνω στο FPGA.

Το προτεινόμενο λογισμικό μπορεί να υποστηρίξει οποιαδήποτε αρχιτεκτονική μνήμης, αρκεί να μοντελοποιηθεί κατάλληλα. Τα δύο σενάρια που εξετάζονται σε αυτή την εργασία επιλέχτηκαν λόγω του ότι είναι ευρέως αποδεκτά στο πεδίο της αρχιτεκτονικής υπολογιστών.

3.3 Προτεινόμενη μεθοδολογία

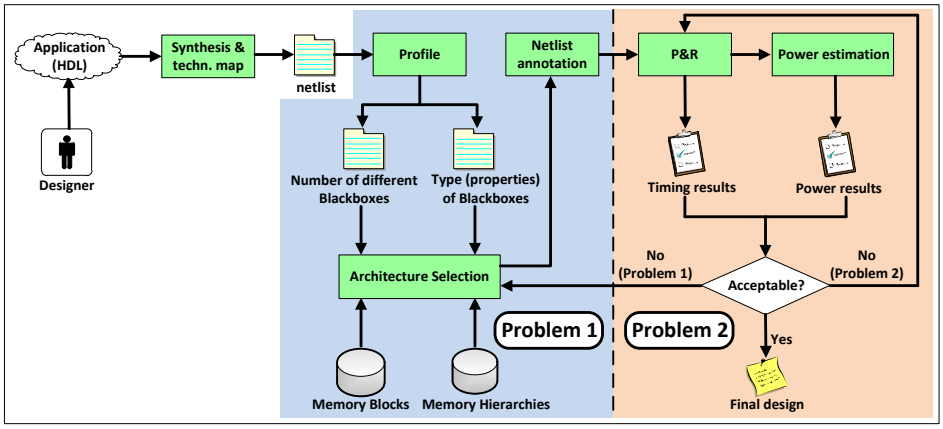
Σε αυτό το κεφάλαιο θα περιγραφεί αναλυτικά η προτεινόμενη μεθοδολογία που στόχο έχει την εξερεύνηση σε επίπεδο αρχιτεκτονικής σε ετερογενή FPGA. Πιο συγκεκριμένα



Σχήμα 3.3: Ένα σχηματικό παράδειγμα της αρχιτεκτονικής διαμοιραζόμενης μνήμης (Σενάριο 2).

η αναφερόμενη μεθοδολογία, η οποία απεικονίζεται σχηματικά στην εικόνα 3.4, μελετά δύο συμπληρωματικά σχεδιαστικά προβλήματα: πρόβλημα (1) η εξερεύνηση σε επίπεδο αρχιτεκτονικής με στόχο τον καθορισμό των παραμέτρων αυτών που επηρεάζουν τα ετερογενή στοιχεία του FPGA και πρόβλημα (2) η υλοποίηση εφαρμογών σε τέτοιες ετερογενείς FPGA πλατφόρμες. Η προτεινόμενη μεθοδολογία είναι δυνατόν να υποστηρίξει συσκευές με πολλαπλούς τύπους ετερογενών μπλοκ, όμως για τους σκοπούς αυτής της εργασίας η αποδοτικότητα της αξιολογείται μόνο σε αρχιτεκτονικές που περιέχουν μπλοκ μνήμης. Σε αυτήν την περίπτωση η ετερογένεια έχει να κάνει και με τις ιδιότητες αυτών των μπλοκ, όπως μέγεθος, κυκλωματικά χαρακτηριστικά, απόδοση και κατανάλωση ενέργειας.

Ως είσοδο η μεθοδολογία δέχεται την περιγραφή της εφαρμογής σε VHDL ή Verilog, η οποία περνάει από το στάδιο της σύνθεσης και technology mapping. Η έξοδος παράγεται σε μορφή αρχείου BLIF. Όπως έχει αναφερθεί ήδη, το BLIF αρχείο έχει περιορισμένη υποστήριξη για κυκλώματα με ετερογενή στοιχεία. Γι' αυτό το λόγο, για να διατηρηθεί η λειτουργικότητα της εφαρμογής μας χρειάζεται να τροποποιηθεί κατάλληλα η netlist στο BLIF αρχείο. Πριν γίνει η οποιαδήποτε τροποποίηση στο αρχείο είναι σημαντικό να σκιαγραφηθεί η εφαρμογή έτσι ώστε να διερευνηθούν οι διαφορετικοί τύποι από μαύρα κουτιά (BBs) του κυκλώματος (π.χ. μνήμες με διαφορετικά χαρακτηριστικά), καθώς και ο αριθμός των στιγμιότυπων από κάθε BB (καθένα από τα οποία έχει διαφορετικές ιδιότητες). Η διαδικασία της σκιαγράφησης καθίσταται ακόμα πιο σημαντική λόγω του



Σχήμα 3.4: Η προτεινόμενη μεθοδολογία.

ότι ένα ετερογενές μπλοκ συνήθως σπάει σε πολλαπλά στιγμιότυπα από τα εργαλεία της σύνθεσης και technology mapping.

Έπειτα, κατά την επιλογή αρχιτεκτονικής τα στιγμιότυπα BBs αντιστοιχούνται με στοιχεία από μια βιβλιοθήκη που περιλαμβάνει περιγραφές πραγματικών μπλοκ. Για επιπλέον ακρίβεια, η καθυστέρηση, η κατανάλωση ενέργειας και το εμβαδόν των μπλοκ της βιβλιοθήκης, είναι βασισμένα σε ευρέως διαδεδομένα και αποδεκτά μοντέλα [62] [63]. Επιλέγοντας κατάλληλους συνδυασμούς από αυτά τα BBs εξερευνούμε ικανοποιητικά αρχιτεκτονικές παραμέτρους όπως ο αριθμός των BBs και η μεταξύ τους οργάνωση. Το αποτέλεσμα αυτού του σταδίου είναι ένα σύνολο από Pareto καμπύλες οι οποίες παρουσιάζουν τους συμβιβασμούς μεταξύ των διαφορετικών κριτηρίων. Βασιζόμενος σε αυτές τις καμπύλες ένας σχεδιαστής μπορεί να σχεδιάσει μια βελτιστοποιημένη FPGA πλατφόρμα.

Τέλος η netlist της εφαρμογής τοποθετείται και δρομολογείται πάνω στο επιλεγμένο FPGA. Μετά την δρομολόγηση, το προτεινόμενο λογισμικό μας παρέχει έναν αριθμό από μετρήσεις (π.χ. καθυστέρηση, ισχύς, εμβαδόν), τα οποία επιτρέπουν την αξιολόγηση της υλοποίησης της εφαρμογής. Σε περίπτωση που η λύση δεν ικανοποιεί τα κριτήρια που έχουν τεθεί, υπάρχει ένας βρόγχος ανατροφοδότησης για επιπλέον βελτιώσεις στην τελική λύση. Πιο συγκεκριμένα, αν ο κύριος στόχος μας είναι να βρεθεί η ιδανική οργάνωση από πόρους υλικού ή BBs σε ένα FPGA (“Πρόβλημα (1)”), ο βρόγχος ανατροφοδότησης επηρεάζει τις αρχιτεκτονικές παραμέτρους. Κατά την διάρκεια αυτού του βήματος, διαφορετικές τοπολογίες και στοιχεία BB επιλέγονται, π.χ. μπλοκ μνήμης με διαφορετική οργάνωση. Από την άλλη, αν ο στόχος είναι να μεγιστοποιήσουμε την επίδοση της υλοποίησης της εφαρμογής, (“Πρόβλημα 2”), τότε ο βρόγχος ανατροφοδότησης πάει στο βήμα της τοποθέτησης και δρομολόγησης.

3.4 Το προτεινόμενο NAROUTO Framework

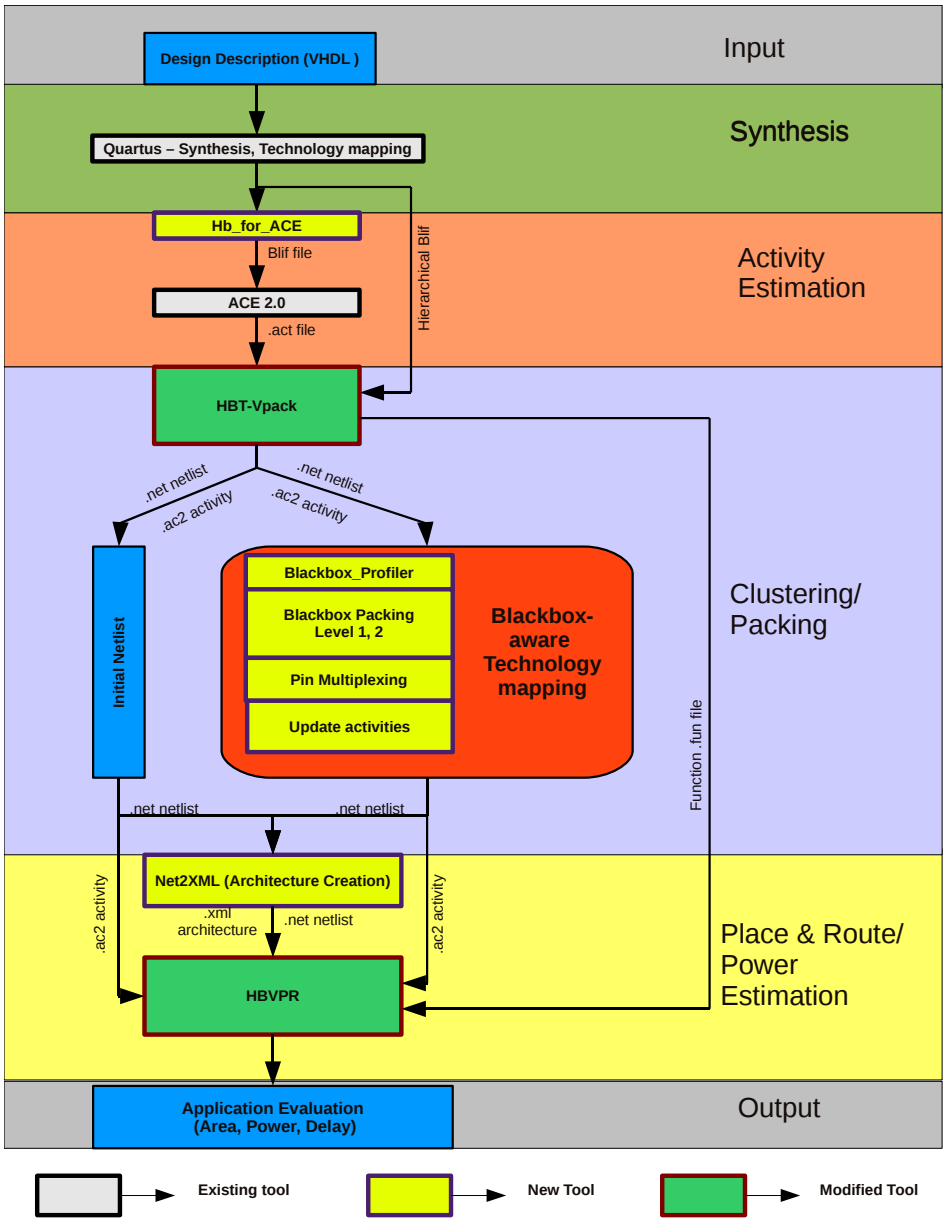
Σε αυτό το τμήμα της διατριβής θα παρουσιαστεί το NAROUTO framework [11], το οποίο υποστηρίζει από πλευράς λογισμικού την μεθοδολογία για εξερεύνηση σε επίπεδο αρχιτεκτονικής ετερογενών FPGAs συσκευών. Αυτό το λογισμικό πλαίσιο, που απεικονίζεται στην εικόνα 3.5, αποτελείται από έναν αριθμό CAD εργαλείων ανοιχτού λογισμικού, τα οποία είτε έχουν αναπτυχθεί από την αρχή είτε είναι τροποποιημένα εργαλεία τα οποία τώρα υποστηρίζουν κυκλώματα με πολλαπλά BBs. Το NAROUTO framework υποστηρίζει συσκευές που περιέχουν διαφορετικούς τύπους ετερογενών στοιχείων, καθ όλη όμως την έκταση αυτής της μελέτης τα BBs αντιπροσωπεύουν BlockRAMs. Μέσα σε αυτό το πλαίσιο δύο υποψήφιες αρχιτεκτονικές μνημών έχουν αξιολογηθεί, που αναπαριστώνται στα Σχήματα 3.2 and 3.3.

3.4.1 Σύνθεση και Technology Mapping

Το πρώτο βήμα στο NAROUTO framework έχει να κάνει με την σύνθεση και το technology mapping της εφαρμογής. Αν και αυτό το βήμα υποστηρίζεται από πολλά ακαδημαϊκά εργαλεία (π.χ. ABC [12], SIS [13]), επιλέχτηκε ένα ώριμο και ευρέως αποδεκτό εμπορικό εργαλείο. Αυτή η επιλογή είναι το Quartus tool [6] της Altera καθώς η έξοδος του (hierarchical netlist in BLIF format), υποστηρίζεται από ακαδημαϊκά εργαλεία και έτσι δεν χρειάζονται περαιτέρω μετατροπές. Να σημειωθεί ότι το BLIF format είναι αναγκαία προϋπόθεση για την πλειοψηφία των ακαδημαϊκών εργαλείων, που ασχολούνται με FPGAs.

Για να βγάζει το Quartus έξοδο σε BLIF format, όπου τα ετερογενή στοιχεία έχουν αντικατασταθεί με BBs, η παρακάτω TCL εντολή χρησιμοποιήθηκε:
set_global_assignment -name INI_VARS "no_add_ops = on;
dump_m:blif_after_lut_map = on"

Ένας περιορισμός της εξόδου του flow, είναι ο υπερβολικά υψηλός αριθμός των BBs τα οποία βρίσκονται στην BLIF netlist. Αυτός ο αριθμός δεν αντιστοιχεί σε πραγματικό αριθμό από χρησιμοποιούμενα macroblocks. Αυτό το πρόβλημα εντείνεται αν συλλογιστούμε ότι δεν υπάρχει μέσα στο αρχείο διαχωρισμός των BBs που ανήκουν σε διαφορετικά μπλοκ του ίδιου τύπου (π.χ. διαφορετικά μπλοκ μνήμης). Τα εργαλεία που περιγράφονται παρακάτω υλοποιούν μηχανισμούς έτσι ώστε να αντιμετωπιστεί αυτό το πρόβλημα.



Σχήμα 3.5: Το προτεινόμενο NAROUTO Framework.

3.4.2 Εκτίμηση Activity

Το επόμενο βήμα στο προτεινόμενο framework έχει να κάνει με την δημιουργία αρχείων activity για την εκτίμηση ισχύος / ενέργειας. Γι αυτό το λόγο ένας αριθμός από ευρέως αποδεκτά μοντέλα χρησιμοποιήθηκαν [15] [62] [63]. Επιπρόσθετα υπάρχουσες εκδόσεις του ACE tool [15] δεν υποστηρίζουν BLIF netlists με BBs. Γι αυτό τον λόγο έχει προστεθεί ένα ενδιάμεσο βήμα προεργασίας που ασχολείται με τον υπολογισμό των στατικών πιθανοτήτων και πυκνοτήτων μετάβασης, από τις πρωτογενείς εισόδους στις

πρωτογενείς εξόδους για όλα τα δίκτυα του κυκλώματος που περιλαμβάνουν έστω και ένα BB.

Το καινούριο εργαλείο, το οποίο ονομάζεται HB_for_ACE, αρχικά αφαιρεί όλα τα BBs από την BLIF netlist και έπειτα συνδέει τις εισόδους και εξόδους των BBs με τις εξόδους και εισόδους του BLIF αρχείου αντίστοιχα. Με την εφαρμογή αυτής της τεχνικής, είναι εφικτό να αφαιρεθούν από την περιγραφή του κυκλώματος όλα τα BBs και ως εκ τούτου τα μοντέλα του ACE 2.0 εργαλείου μπορούν να τρέξουν επιτυχώς. Από την άλλη πλευρά, όσον αφορά τα δίκτυα που περιλαμβάνουν τουλάχιστον ένα BB, οι αντίστοιχες τιμές της στατικής πιθανότητας και της πυκνότητας μετάβασης μπορούν να ανακτηθούν από προσομοιώσεις.

Ο αλγόριθμος 1 του open-source HB_for_ACE (transform Hierarchical Blifs for ACE) εργαλείου παρουσιάζεται με την μορφή ψευδοκώδικα παρακάτω:

ALGORITHM 1: Ψευδοκώδικας για το εργαλείο HB_for_ACE.

```
function hb_for_ace (Input_m:blif) {
  // Input: m:blif netlist with BBs
  // Output: m:blif netlist compatible with ACE

  BB_inputs[ ]; // Array for storing all BBs input pins
  BB_outputs[ ]; // Array for storing all BBs output pins
  primary_inputs[ ]; // Array for storing primary input pins
  primary_outputs[ ]; // Array for storing primary output pins

  // Get the primary I/O pins of the design
  primary_inputs[ ] = get_primary_inputs(Input_m:blif);
  primary_outputs[ ] = get_primary_outputs(Input_m:blif);

  // Get the blackboxes' I/O pins
  BB_inputs[ ] = get_blackbox_inputs(Input_m:blif);
  BB_outputs[ ] = get_blackbox_outputs(Input_m:blif);

  // Delete any reference to blackboxes from the m:blif netlist
  delete_blackbox_subcircuits(Input_m:blif);
  delete_blackbox_models(Input_m:blif);

  // Connect the BBs I/Os to the design's primary O/I pins
  append(primary_inputs[ ], BB_outputs[ ]);
  append(primary_outputs[ ], BB_inputs[ ]);

  // Print the ACE compatible m:blif netlist
  printout_final_m:blif(Output_m:blif_filename);
}
```

3.4.3 Technology mapping σε ετερογενή FPGA

Έχοντας ως είσοδο την περιγραφή της εφαρμογής σε BLIF, που περιλαμβάνει επίσης πληροφορίες σχετικά με BBs, η επόμενη εργασία στη μεθοδολογία ασχολείται με το

packing των technology mapped στοιχείων σε λογικά μπλοκ (CLBs). Το μέγεθος των clusters που παράγεται, εξαρτάται από την χρησιμοποιούμενη FPGA αρχιτεκτονική. Αυτό το βήμα υποστηρίζεται από ένα σύνολο εργαλείων CAD, τα οποία βασίζονται στο T-VPack [64] [5]. Τα εργαλεία αυτά έχουν επεκταθεί έτσι ώστε να αναγνωρίζουν πολλών είδη BB, καθένα από τα οποία μπορεί να έχει διαφορετικές ιδιότητες. Επιπλέον, αυτά τα εργαλεία ξεπερνάνε τους περιορισμούς του Quartus synthesizer σχετικά με τον χειρισμό netlists με BBs.

Στις επόμενες ενότητες περιγράφονται με περισσότερη λεπτομέρεια τα εργαλεία που έχουν αναπτυχθεί για την υποστήριξη του technology mapping σε ετερογενή FPGA.

3.4.3.1 BlackBox Profiler

Ο BlackBox_Profiler αναλύει την περιγραφή της εφαρμογής, έτσι ώστε να προσδιοριστούν οι διαφορετικοί τύποι των BB, καθώς και το πόσα στιγμιότυπα από το καθένα τύπο χρησιμοποιούνται για την εφαρμογή. Μέρος αυτής της διαδικασίας ασχολείται επίσης με την κατάλληλη μοντελοποίηση αυτών των BB, προκειμένου να ανταποκριθούν καλύτερα στις προδιαγραφές των ετερογενών φυσικών στοιχείων που αντικαθιστούν. Τυπικά παραδείγματα αυτών των προδιαγραφών είναι η λειτουργικότητα των ετερογενών στοιχείων (π.χ. μνήμη, DSP, κλπ.), το μέγεθος τους, καθώς ο αριθμός των I/O. Για να ανακτηθούν αυτές τις ιδιότητες, το netlist της εφαρμογής αναλύεται, για να προσδιοριστούν όλα τα BBs που ανήκουν σε ένα ενιαίο macroblock. Αυτό είναι εφικτό για να επιτευχθεί διότι όλα τα BB που ανήκουν σε ένα macroblock χρησιμοποιούν τα ίδια σήματα (π.χ. το ανάγνωσης / εγγραφής / ενεργοποίησης της μνήμης RAM) για τον έλεγχο και την επικοινωνία με τα υπόλοιπα στοιχεία του FPGA. Στη συνέχεια, οι προδιαγραφές για κάθε BB ανακτώνται από μια βιβλιοθήκη διαφορετικών τεχνολογιών όπως αναφέρθηκε στο προηγούμενο κεφάλαιο. Οι τιμές αυτές θα χρησιμοποιηθούν αργότερα για την αξιολόγηση της εφαρμογής πάνω στο FPGA όσον αφορά την καθυστέρηση, ισχύ / ενέργεια και το εμβαδόν του κυκλώματος. Ο Αλγόριθμος του BlackBox Profiler απεικονίζεται σε ψευδοκώδικα στην εικόνα 2.

3.4.3.2 BlackBox Packing

Η έξοδος από τον BlackBox_Profiler δίνει μια σειρά από κατευθυντήριες γραμμές σχετικά με το πώς να ενωθούν όλα τα επιμέρους BBs που ανήκουν στο ίδιο macroblock, σε ένα ενιαίο BB. Αυτό το βήμα, που αναφέρεται ως “Single-Packed” ή SP, στο NAROUTO framework υποστηρίζεται με το εργαλείο BlackBox_Packing. Επιπλέον, το προτεινόμενο framework υποστηρίζει ένα ακόμη επίπεδο packing που αναφέρονται ως “Full-Packed” ή FP. Ο στόχος αυτής της πρόσθετης επιλογής packing είναι να ενώσει αναδρομικά όλα τα BB του ίδιου τύπου, σε ένα μεγαλύτερο σούπερ-BB. Για παράδειγμα, ας υποθέσουμε ότι οι απαιτήσεις μνήμης για μία δεδομένη εφαρμογή είναι 16× 1kByte μπλοκ μνήμης RAM.

ALGORITHM 2: Ψευδοκώδικας για το εργαλείο Blackbox_Profiler.

```
function blackbox-aware_technology_mapping {
  struct Blackbox {
    blackbox_name; blackbox_inputs[ ]; blackbox_outputs[ ];
  };
  struct Type {
   blackbox_name; blackbox_inputs[ ]; blackbox_outputs[ ];
   instances_num; blackbox_func;
  };
  struct Type blackbox_types[ ];
  struct Blackbox blackboxes[ ];

  // Find BBs utilized into the design
  blackboxes[ ] = get_blackboxes_instances();
  blackboxes_array_size=get_size(blackboxes[]);
  blackbox_types_array_size=0;

  new_type_flag = 1;
  for (i=0;i<blackboxes_array_size;i++) {
    for (j=0;j<blackbox_types_array_size;j++) {
      // Search all known BB types by comparing control signals
      if (control_pins_match(blackboxes[i],blackbox_types[j])) {
        blackbox_types[j].instances num++;
        new_type_flag=0;
        break;
      }
    }
    if (new_type_flag==1) {
      // Create a new instance for this BB type
      struct Type new;
      new.blackbox_name = blackboxes[i].name;
      new.blackbox_inputs = blackboxes[i].inputs;
      new.blackbox_outputs = blackboxes[i].outputs;
      new.blackbox_instances_num = 1;
      add element to array(new, blackbox_types[ ]);
      blackbox_types_array_size++;
    }
  }
  // Find properties for this BB from a technology library
  for (i=0;i<blackbox_types_array_size;i++) {
    blackbox types[i]c = get_info_from_tech_lib();
  }
}
```

Η BLIF netlist, όπως προκύπτει από το Quartus αναφέρει ότι η εφαρμογή περιλαμβάνει 16.384 (16 × 1,024) BB, καθένα από τα οποία στην πραγματικότητα αντιστοιχεί σε ένα byte. Μετά την εφαρμογή του SP, η προκύπτουσα netlist έχει 16 BBs, καθένα από τα οποία αντιπροσωπεύει το 1 kByte, ενώ με το δεύτερο επίπεδο της συσκευασίας (FP), η netlist θα περιέχει μόνο 1 σούπερ-BB με 16 kBytes μέγεθος. Να σημειωθεί ότι κατά τη διάρκεια του SP και FP packing, η επιθυμητή ιεραρχία μνήμης λαμβάνεται υπόψη, όπως συζητήθηκε ήδη στο κεφάλαιο 3.

Οι ψευδοκώδικες 3 και 4 παρουσιάζουν τον αλγόριθμο για BB packing επίπεδο 1 (SP) and επίπεδο 2 (FP), αντίστοιχα.

ALGORITHM 3: Αλγόριθμος για black-box Packing level 1.

```
function BB_Packing_Level_1 {

  // Stores the BB types. This info was already extracted during
// BB profiling
  blackbox_types[ ];

  // Stores all the BB instances, as they found during BB profiling
  blackboxes[ ];

  // Stores the new packed BBs
  packed_blackboxes[ ] = blackbox_types[ ];

  for (i=0;i$<blackboxes_array_size;i++) {
    // For each BB instance
    for (j=0;j$<blackbox_types_array_size;j++) {
    // Search all known BB types by comparing their control signals
      if (control_pins_match(blackboxes[i], blackbox_types[j])) {
      // If the BB's type is found, then it is merged with the
      // BB instance
        packed_blackboxes[j] =
          merge(packed_blackboxes[j], blackboxes[i]);
        break;
      }
    } // End of the BB types loop
  } // End of the BB instances loop
}
```

3.4.3.3 Πολύπλεξη I/O

Εκτός από τον αριθμό των partial BB που ανακτώνται μετά τη σύνθεση και το technology mapping, θα δείξουμε αργότερα ότι καθένα από αυτά τα BB απαιτούν υπερβολικό αριθμό I/O. Αυτό επιβάλλει ότι η FPGA αρχιτεκτονική που θα χρησιμοποιηθεί χρειάζεται ένα ευρύτερο κανάλι δρομολόγησης, το οποίο με τη σειρά του οδηγεί σε μεγαλύτερη καθυστέρηση στο κύκλωμα, περισσότερη ενέργεια και μεγαλύτερο εμβαδόν. Πιο συγκεκριμένα, με βάση την ανάλυση, βρέθηκε ότι του μόνο ένα υποσύνολο των I/O απαιτείται πραγματικά για τη διατήρηση της λειτουργικότητας της εφαρμογής. Ως εκ τούτου,

ALGORITHM 4: Αλγόριθμος για black-box Packing level 2.

```
function BB_Packing_Level_2 {
  // Stores the packed BBs, as they already retrieved from BB
  // packing level 1
  packed_blackboxes[ ];

  // An super-block which stores the FP BB
  full_packed_blackbox;
  for (i=0;i<packed_blackboxes_array_size;i++) {
    full_packed_blackbox =
      merge(full_packed_blackbox, packed_blackboxes[i]);
  } // End of the packed BB loop
}
```

το NAROUTO framework παρέχει ένα μηχανισμό που προσδιορίζει τον απαιτούμενο αριθμό για κάθε BB και εξαλείφει τα περιττά I/O. Ο αλγόριθμος αυτού του εργαλείου, που ονομάζεται Pin_Multiplexing, απεικονίζεται σε ψευδοκώδικα στην εικόνα 5.

Κατά τη διάρκεια αυτού του βήματος δεν γίνεται ακόμα συγχώνευση σημάτων, δεδομένου ότι αυτό θα υπονόμευε τη δομική και λειτουργική ακεραιότητα του τελικού netlist. Αντίθετα, η μείωση των pins βασίζεται στην υλοποίηση μιας σειράς πολυπλεκτών σε CLBs. Πιο συγκεκριμένα, τα σήματα εισόδου ενός BB περνούν αρχικά μέσω CLB πολύπλεξης και οι νέες πολυπλεγμένες πια έξοδοι τροφοδοτούνται ως είσοδοι στο BB. Παρομοίως, τα σήματα εξόδου από BBs είναι πολυπλεγμένα και περνάνε μέσα από CLBs απόπλεξης πριν από τη σύνδεση με το υπόλοιπο του netlist. Με βάση τις προδιαγραφές του κυκλώματος, όπως αυτές εξάγονται από τη βιβλιοθήκη (Σχήμα 3.4) macroblock, τα I/O για κάθε BB μπορεί να πολυπλεχθούν αναδρομικά πολλές φορές, προκειμένου να αντιπροσωπεύουν τον πραγματικό αριθμό των ακίδων των αντίστοιχων ετερογενών μπλοκ που αντικαθιστούν.

3.4.3.4 Ενημέρωση Activity

Η τεχνική της πολύπλεξης I/O που συζητήθηκε προηγουμένως, επιβάλλει μεταβολές στην δρομολόγηση της εφαρμογής. Αυτές οι αλλαγές συμβαίνουν κυρίως επειδή τα BBs πρέπει να συνδέονται με το υπόλοιπο του σχεδιασμού μέσω λιγότερων I/O pins. Προκειμένου να ληφθεί υπόψη η επίπτωση της πολύπλεξης κατά τη διάρκεια της εκτίμησης κατανάλωσης ισχύος, πληροφορίες σχετικά με τη δραστηριότητα των σημάτων πρέπει να ενημερωθούν αναλόγως. Σημειώνεται ότι κατά τη διάρκεια αυτού του βήματος τα πρόσθετα δίκτυα που υλοποιούν την πολύπλεξη λαμβάνονται επίσης υπόψη. Ο ψευδοκώδικας 6 παρουσιάζει τον αλγόριθμο για τον υπολογισμό της στατικής πιθανότητας και πυκνότητας μετάβασης σχετικά με την πολύπλεξη σημάτων.

ALGORITHM 5: Αλγόριθμος για πολύπλεξη I/O.

```
function pin_multiplexing {

   // Array for storing the packed BBs, as it was derived from FP1
   sp/fp_blackboxes[ ];

   // Define the aggressiveness for pin multiplexing.
   // Levels 1, 2, ... denote that I/Os of BBs will be
   // multiplexed once, twice, etc
   multiplexion_level;

   // Each CLB multiplex a number of I/O pins equals to its number
   // of inputs minus 1 (for clock input)
   clb_mux_pin_num = CLB_input_num - 1 ;

   // Each CLB demultiplex a number of I/O pins equal to its number
   // of LUTs
   clb_demux_pin_num = CLB_LUT_num;

   for (i=0;i<sp/fp_blackboxes_array_size;i++) { // For each BB
     // Temporary storage of I/Os for a BB
     input_pins[ ] = get_inputs(sp/fp_blackboxes[i]);
     output_pins[ ] = get_outputs(sp/fp_blackboxes[i]);
     in_pin_num=get_length(input_pins[ ]);
     out_pin_num=get_length(output_pins[ ]);
     for (j=0;j<multiplexion_level;j++) {
       // Multiplex the I/O of BBs ``multiplexion_level'' times
       for (k=0;k<in_pin_num;k+=clb_mux_pin_num) {
         // Multiplex ``clb_mux_pin_num'' pins in every
         // multiplexing CLB
         create_mux_clb (input_pins[k],
           input_pins[k+clb_mux_pin_num]);
       }
       for (k=0;k<out_pin_num;k+=clb_demux_pin_num) {
         // Demultiplex ``clb_demux_pin_num'' pins in every
         // demultiplexing CLB
         create_demux_clb (output_pins[k],
           output_pins[k+clb_demux_pin_num]);
       }
       // I/Os are updated with the new multiplexed pins to enable
       // re-multiplexing
       input_pins[ ] = get_multiplexed_input_pins();
       output_pins[ ] = get_multiplexed_output_pins();
     }
   }
}
```

ALGORITHM 6: Αλγόριθμος για το εργαλείο Activity_Updater.

```
function update_activities {

  // Identify all the I/O signals of BBs
  io_signals_of_BBs[ ];
  // Identify static_probability and transitional_density for each
  // BB signal
  activities_of_BBs[ ];

  for (i=0;i<io_signals_of_BBs_array_size;i++) {
    // For all the multiplexed signals
    // tmp_signals[ ] array stores all the multiplexed signals
    tmp_signals[ ] =
      get_all_signals_multiplexed_in(io_signals_of_BBs[i]);
    // Store the static_probability and transitional_density
    // of a multiplexed signal
    tmp_prob = get_signal_probability(io_signals_of_BBs[i]);
    tmp_dens = get_signal_density(io_signals_of_BBs[i]);

    for (j=0;j<tmp_signals_array_size;j++) {
      // Compute static_probability and transitional_density for
      // multiplexed signals
      static_probability = calculate_static_prob(tmp_prob);
      transitional_density = calculate(tmp_dens);
      // Update the signal's activity
      update_activity(tmp_signals[j], static_probability
        ,transitional_density);
    }
  }
}
```

3.4.4 Τοποθέτηση και δρομολόγηση

Το τελευταίο στάδιο στο προτεινόμενο framework, είναι η τοποθέτηση και η δρομολόγηση της εφαρμογής πάνω στο FPGA. Για το σκοπό αυτό ένας simulated annealing αλγόριθμος χρησιμοποιείται για την τοποθέτηση και η δρομολόγηση γίνεται μέσω ενός congestion pathfinder αλγόριθμου. Και οι δύο αλγόριθμοι αναπτύχθηκαν με βάση το VPR [64] [5], αλλά έχουν τροποποιηθεί εκτενώς για να αντιμετωπίζουν τους περιορισμούς που τίθενται από τα ετερογενή στοιχεία. Πιο συγκεκριμένα, η εφαρμογή αυτών των αλγορίθμων, στο NAROUTO framework, παρέχει τεχνικές για αποτελεσματικό χειρισμό πολλαπλών τύπων ετερογενών BB, καθώς και εκτίμηση της κατανάλωσης ενέργειας μέσω κατάλληλης επέκτασης του Powermodel εργαλείου [15]. Το νέο εργαλείο μπορεί να χειριστεί ετερογενή FPGAs με ενσωματωμένα macroblock, εκτός από μνήμες, αρκεί να είναι κατάλληλα ορισμένα στη βιβλιοθήκη στοιχείων.

3.5 Πειραματικά αποτελέσματα

Αυτή η ενότητα παρέχει μια σειρά από ποιοτικές και ποσοτικές συγκρίσεις που αποδεικνύουν αποτελεσματικότητα του NAROUTO framework . Η σύνθεση και το technology mapping τόσο για το προτεινόμενο framework, όσο και για τις υπάρχουσες συγκρινόμενες λύσεις, πραγματοποιήθηκαν με τη χρήση του εργαλείου Quartus [6].

Ο πίνακας 3.1 δίνει μια ποιοτική σύγκριση μεταξύ του προτεινόμενου framework, της state-of-the-art λύσης, καθώς και ένα σύνολο εργαλείων που διατίθενται στο εμπόριο. Η σύγκριση αυτή γίνεται με γνώμονα έναν αριθμό διαφορετικών κριτηρίων προσανατολισμένων στην αρχιτεκτονική (π.χ. υποστήριξη ετερογένειας), στις εφαρμογές (π.χ. περιορισμοί στην τοποθέτηση της εφαρμογής) και στην εφαρμογή όλου του framework (π.χ. παράμετροι toolflow).

Πίνακας 3.1: Ποιοτική σύγκριση σε υποστηριζόμενες λειτουργίες.

Feature	**NAROUTO**	**VTR [14]**	**QUARTUS [6]**
Architecture-level exploration	Yes	Yes	No
Support BBs	Yes	Yes	Yes
Support for different BB hierarchies	Yes	No	No
Different types of BBs	Yes	Yes	-
Power estimation	Yes	Yes	Yes
Modular tools	Yes	Yes	No

Ένας αριθμός συμπερασμάτων μπορεί διεξαχθεί από αυτόν τον πίνακα. Το NAROUTO υποστηρίζει πιο αποτελεσματικά σχέδια με BB, ενώ η εκτίμηση κατανάλωσης ισχύος και ενέργειας είναι παρόμοια με εκείνη που συναντάται σε σχετικές εμπορικές προσεγγίσεις. Επιπλέον, παρατηρούμε ότι μόνο ακαδημαϊκά toolflows (π.χ. NAROUTO και VTR) επιτρέπουν την εξερεύνηση σε επίπεδο αρχιτεκτονικής. Ως εκ τούτου, τα εμπορικά εργαλεία αντιμετωπίζουν αποκλειστικά το "Πρόβλημα II" (βλέπε σχήμα 3.3), ενώ η προτεινόμενη λύση υποστηρίζει επίσης το "Πρόβλημα I".

Για τους σκοπούς της αξιολόγησης, χρησιμοποιήθηκαν DSP εφαρμογές από την Quip εργαλειοθήκη της Altera [6]. Ο πίνακας 3.2 συνοψίζει τα κύρια χαρακτηριστικά των δοκιμαστικών εφαρμογών, ενώ η πολυπλοκότητα αυτών των εφαρμογών εγγυάται ότι τα συμπεράσματα που προέρχονται ισχύουν για την πλειοψηφία των ψηφιακών σχέδιων που υλοποιούνται σε FPGAs.

Το framework δεν επικεντρώνεται στην ελαχιστοποίηση των απαιτήσεων μνήμης, ή των προσβάσεων στην μνήμη, αφού υποτίθεται, ότι τα προβλήματα αυτά αντιμετωπίζονται κατά την σύνθεση με Altera Quartus. Τα λογικά στοιχεία του FPGA, αποτελούνται από 10 LUTs 4 εισόδων και 22/10 ακροδέκτες εισόδου/εξόδου ανά CLB, ενώ μέγεθος του FPGA, καθώς και το πλάτος του καναλιού δρομολόγησης, εξαρτάται από την εκάστοτε εφαρμογή. Ειδικότερα, η τιμές των δύο αυτών παραμέτρων αντιστοιχούν στο ελάχιστο

Πίνακας 3.2: Δοκιμαστικές εφαρμογές από [6].

Benchmark	**Functionality**	**4-LUT**	**F/Fs**	**RAM bits**	**I/Os**
oc_aes_core_inv	Encryption	5,144	536	34,176	389
oc_ata_ocidec3	Processor	1,589	594	224	130
oc_hdlc	Processor	859	926	2,048	82
oc_minirisc	Processor	908	300	1,025	389
oc_oc8051	Processor	4,306	754	4,608	189
os_blowfish	Encryption	5,368	891	67,168	585
Average:		**3,092**	**666.8**	**18,208**	**294**

μέγεθος και πλάτος καναλιού, αντίστοιχα, που απαιτούνται για την επιτυχή εφαρμογή P&R.

3.5.1 Αξιολόγηση των διαφορετικών ιεραρχιών μνήμης

Αρχικά αξιολογούμε την μέγιστη συχνότητα λειτουργίας και την κατανάλωση ενέργειας όσον αφορά την δύο ιεραρχίες μνήμης. Γι 'αυτό το σκοπό ο πίνακας 3.3 ποσοτικοποιεί την μέγιστη συχνότητα λειτουργίας για τις εναλλακτικές ιεραρχίες μνήμης, που αναφέρονται ως "Σενάριο 1" και "Σενάριο 2" στα Σχήματα 3.2 και 3.3, αντίστοιχα. Ο πίνακας 3.4 δείχνει τη σύγκριση όσον αφορά την κατανάλωση ενέργειας (mWatt) για τις ίδιες ιεραρχίες μνήμης.

Πίνακας 3.3: Αξιολόγηση στην διάρκεια της μέγιστης συχνότητας λειτουργίας (MHz) για διαφορετικές ιεραρχίες μνήμης.

Benchmark	**Scenario 1**	**Scenario 2**
cc_aes_core_inv	13.870	15.798
oc_ata_ocidec3	17.212	15.848
oc_hdlc	17.889	18.587
oc_minirisc	14.245	15.432
oc_oc8051	4.762	7.463
os_blowfish	9.524	9.615
ucsb_152_tap_fir	9.009	8.929
Average:	**12.359**	**13.096**

Τα αποτελέσματα αυτά υποδηλώνουν ότι οι ιεραρχίες μνήμης οδηγούν σε καλύτερη απόδοση λόγω της βελτιωμένης μεταχείρισης της μεταφοράς δεδομένων. Δεδομένου ότι το FPGA θα πρέπει να εμφανίζει όσο το δυνατόν υψηλότερη απόδοση, για το υπόλοιπο της εργασίας στην αρχιτεκτονική που χρησιμοποιείται, τα μπλοκ μνήμης οργανώνονται με βάση την ιεραρχία που απεικονίζεται στο Σχήμα 3.3 ("Σενάριο 2"). Να σημειωθεί ότι σε όλη αυτή τη μελέτη ο στόχος δεν είναι να βρεθεί η βέλτιστη ιεραρχία μνήμης που μεγιστοποιεί την απόδοση. Αντιθέτως, το framework μπορεί ποσοτικοποιήσει την απόδοση μιας συγκεκριμένης ιεραρχίας μνήμης, ενώ επίσης υποστηρίζει μια αποτελεσματική απεικόνιση της εφαρμογής επάνω σε αυτήν τη συσκευή. Πρόσθετες ιεραρχίες μνήμης που παρέχουν ακόμα μεγαλύτερες επιδόσεις μπορούν να βρεθούν σε

Πίνακας 3.4: Αξιολόγηση της κατανάλωσης ισχύος της εφαρμογής (mWatt) για διαφορετικές ιεραρχίες μνήμης.

Benchmark	**Scenario 1**	**Scenario 2**
cc_aes_core_inv	664.40	590.04
oc_ata_ocidec3	16.97	15.70
oc_hdlc	127.03	99.73
oc_minirisc	32.69	25.67
oc_oc8051	414.78	313.77
os_blowfish	124.24	117.12
ucsb_152_tap_fir	675.07	461.36
Average:	**293.59**	**231.91**

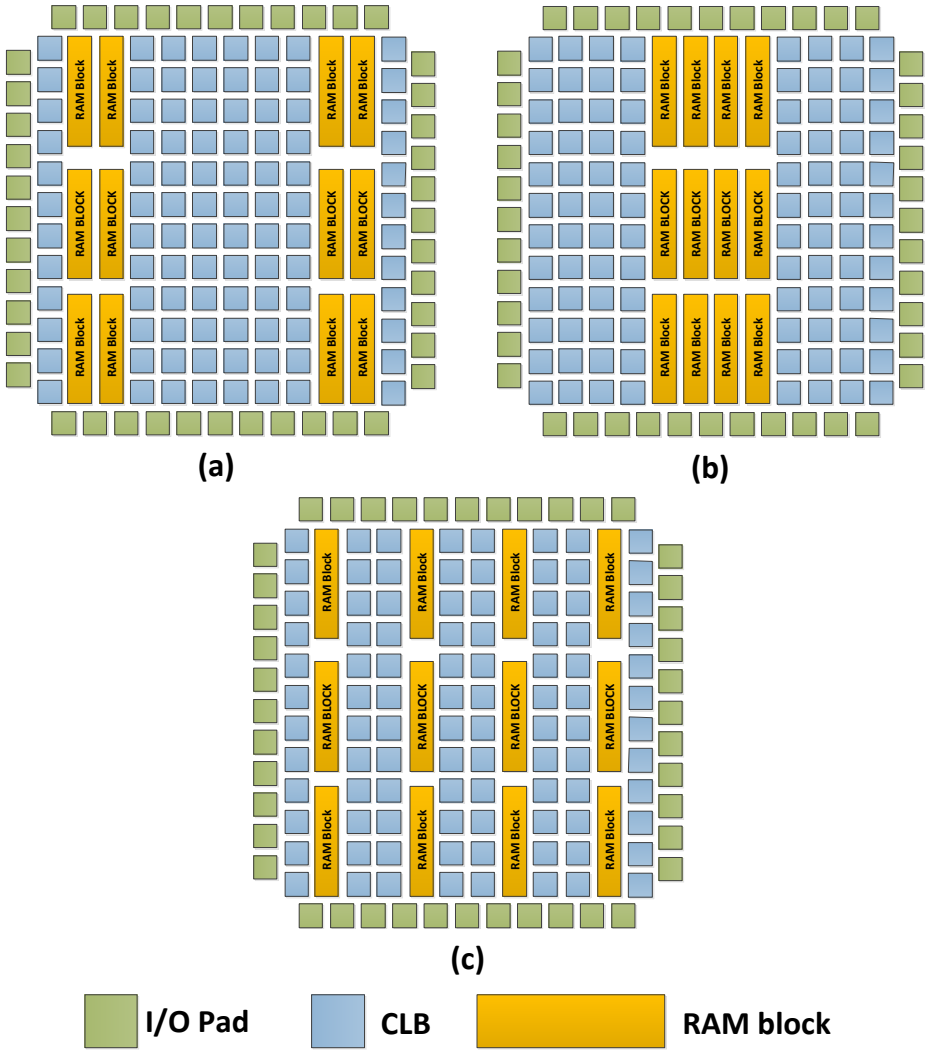
σχετική βιβλιογραφία, αλλά αυτό είναι πέρα από το πεδίο ενδιαφέροντος αυτής της εργασίας.

3.5.2 Αξιολόγηση των εναλλακτικών floor-plans της μνήμης

Σε αυτή την ενότητα μελετάται ένας αριθμός διαφορετικών floor-plans για τα μπλοκ μνήμης που ακολουθούν την ιεραρχία που απεικονίζεται στο "Σενάριο 2". Η ανάλυση θα καθορίσει τα χαρακτηριστικά των διαφορετικών χωρικών κατανομών των μπλοκ μνήμης πάνω στην αρχιτεκτονική FPGA. Για το σκοπό αυτό, τρία αντιπροσωπευτικά floor-plans αξιολογούνται, όπως αυτά απεικονίζονται στο Σχήμα 3.6. Πιο συγκεκριμένα, μελετώνται FPGAs όπου οι μνήμες έχουν ανατεθεί τα σύνορα της συσκευής (Σχήμα 3.6 (a)), προς το κέντρο της συσκευής (Σχήμα 3.6 (b)), καθώς και ένα σενάριο όπου οι μνήμες είναι ομοιόμορφα κατανεμημένες σε όλη την αρχιτεκτονική FPGA (Σχήμα 3.6 (c)). Για το υπόλοιπο της εργασίας, τα εν λόγω floor-plans δηλώνονται ως "Border", "Center" and "Uniform", αντίστοιχα. Σε αυτό το σχήμα, τα γκρι κουτιά υποδηλώνουν λογικά cells (CLBs), ενώ τα μπλοκ μνήμης (BBs) απεικονίζονται με διαφορετικά χρώματα. Εκτός από τα floor-plans που εξετάζονται εδώ, οποιαδήποτε άλλα μπορούν επίσης να αξιολογηθούν με το NAROUTO framework.

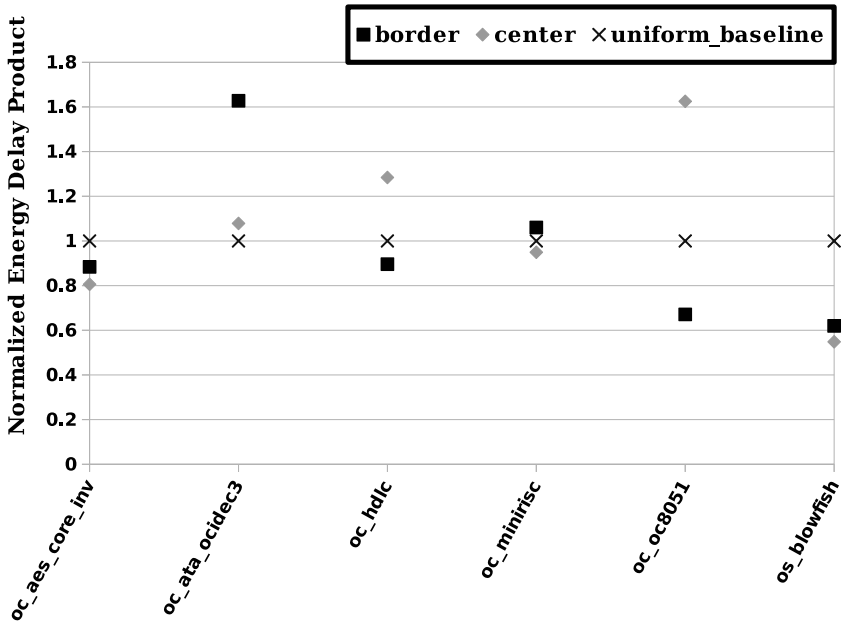
Πίνακας 3.5: Αξιολόγηση της μέγιστης συχνότητας λειτουργίας (MHz) για τα διάφορα floor-plans των μπλοκ μνήμης.

Benchmark	**Max. Operation Frequency (MHz)**		
	Border	**Center**	**Uniform**
oc_aes_core_inv	17.52	17.57	17.04
oc_ata_ocidec3	22.39	28.57	31.63
oc_hdlc	41.67	30.58	43.95
oc_minirisc	19.82	21.20	21.20
oc_oc8051	9.72	5.91	8.57
os_blowfish	14.93	14.80	13.52
Average:	**21.01**	**19.77**	**22.65**



Σχήμα 3.6: Εναλλακτικά floor-plans για μπλοκ μνήμης: (a) στα σύνορα, (b) στο κέντρο, και (c) ομοιόμορφα κατανεμημένα.

Η χωρική κατανομή των μπλοκ μνήμης, όπως ανακτώνται από τα εναλλακτικά floor-plans που συζητήθηκαν στην παρούσα ενότητα, οδηγεί σε διακύμανση του μήκους των μονοπατιών δρομολόγησης και ως εκ τούτου αναμένεται να επηρεάσει σημαντικά την καθυστέρηση της εφαρμογής και κατανάλωση ισχύος. Δεδομένου ότι η συσκευή είναι ένα FPGA γενικού σκοπού, η επιλογή του κατάλληλου floor-plan βασίζεται στην ελαχιστοποίηση του Energy×Delay product (EDP). Η εικόνα 3.7 αναπαριστά την γραφική παράσταση του EDP για τις δοκιμαστικές εφαρμογές, ενώ οι πίνακες 3.5 και 3.6 δίνουν την μέγιστη συχνότητα λειτουργίας και την κατανάλωση ισχύος για τις τρεις εναλλακτικές λύσεις αντίστοιχα. Αν και οι συχνότητες λειτουργίας δεν διαφέρουν πολύ, υπάρχει σημαντική διακύμανση στην κατανάλωση ενέργειας. Εάν οι μνήμες είναι στο κέντρο της συσκευής FPGA, η κατανάλωση ενέργειας μειώνεται, ενώ όταν είναι κατανεμημένες ομοιόμορφα αποδεικνύεται ότι απαιτείται περισσότερη ενέργεια.



Σχήμα 3.7: Energy×Delay product για διαφορετικά floor-plans, κανονικοποιημένα ως προς την ομοιόμορφο floor-plan.

Από το σχήμα 3.7 καταλήγουμε στο συμπέρασμα, ότι κάθε φορά που τα μπλοκ μνήμης τοποθετούνται στα σύνορα του FPGA, αυτό οδηγεί στην ελάχιστη τιμή EDP κατά μέσο όρο, 5% λιγότερο σε σύγκριση με την ομοιόμορφη κατανομή και 10% με την τοποθέτηση στο κέντρο.

Πίνακας 3.6: Αξιολόγηση της κατανάλωση ισχύος (mWatt) για τα διάφορα floor-plans των μπλοκ μνήμης.

Benchmark	**Power Consumption (mWatt)**		
	Border	**Center**	**Uniform**
oc_aes_core_inv	375.298	344.102	401.915
oc_ata_ocidec3	49.4953	53.4011	60.716
oc_hdlc	99.6392	76.8953	123.678
oc_minirisc	34.1706	35.0072	36.8794
oc_oc8051	628.028	563.022	727.1
os_blowfish	327.127	284.425	432.708
Average:	**252.29**	**226.14**	**297.17**

3.6 Συμπεράσματα

Στο παρόν κεφάλαιο προτάθηκε μια νέα μεθοδολογία, καθώς και το λογισμικό framework, για τη διευκόλυνση της εξερεύνησης σε επίπεδο αρχιτεκτονικής για ετερογενή FPGAs. Το framework αυτό επιτρέπει την αποτελεσματική διαχείριση των ιεραρχιών μνήμης

σε γενικής χρήσης reconfigurable συσκευές. Τα πειραματικά αποτελέσματα αποδεικνύουν την αποτελεσματικότητα της προτεινόμενης λύσης, δεδομένου ότι οι διαφορετικές ιεραρχίες μνήμης και τα αρχιτεκτονικά floor-plans μελετήθηκαν και χαρακτηρίστηκαν επιτυχώς.

Κεφάλαιο 4

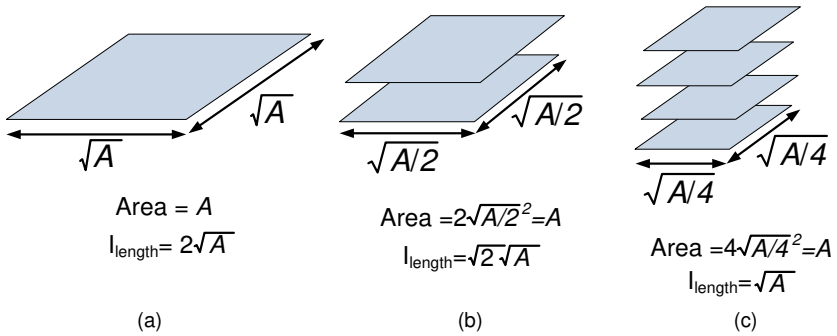
3D Επαναδιαμορφούμενες Αρχιτεκτονικές

4.1 Εισαγωγή

Οι δομές διασύνδεσης των FPGA συμβάλλουν με την σμίκρυνση των τρανζίστορ όλο και περισσότερο στην καθυστέρηση και στην κατανάλωση ενέργειας. Η ζήτηση για ακόμη υψηλότερες συχνότητες ρολογιού καθιστά το πρόβλημα ακόμη πιο σημαντικό. Οι τρισδιάστατες (3-D) αρχιτεκτονικές μπορούν να κρατήσουν ζωντανό τον νόμο του Moore και να τροφοδοτήσουν το επόμενο κύμα των καταναλωτικών ηλεκτρονικών προϊόντων.

Μια κοινή πρακτική για τους σχεδιαστές είναι να εκτιμούν ότι η μακρύτερη διασύνδεση ενός κυκλώματος ισούται με δύο φορές το μήκος της ακμής του FPGA [63]. Τα πιθανά κέρδη της νέας 3D προσέγγισης παρατηρούνται στην εικόνα 4.1, όπου το μήκος διασύνδεσης σε πλατφόρμες 3-D είναι σημαντικά μειωμένο σε σύγκριση με τις συμβατικές αρχιτεκτονικές 2-D. Πιο συγκεκριμένα, για μια δεδομένη συνολική επιφάνεια τσιπ A (τόσο για 2-D και 3-D), καθώς ο αριθμός των τρισδιάστατων επιπέδων αυξάνει, η έκταση ανά επίπεδο και κατά συνέπεια το αντίστοιχο μακρύτερο μονοπάτι διασύνδεσης μειώνονται. Για παράδειγμα, το μακρύτερο μονοπάτι δρομολόγησης για μία 3-D αρχιτεκτονική που αποτελείται από τέσσερα στρώματα είναι σχεδόν το μισό σε σύγκριση με την 2-D αρχιτεκτονική.

Επιπλέον, η στοίβαξη μικρότερων στρωμάτων αντί της κατασκευής μιας μεγάλης επιφάνειας οδηγεί σε yield-cost βελτίωση επειδή η πιθανότητα μια μήτρα να είναι ελαττωματική συσχετίζεται με το εμβαδόν της. Κατά συνέπεια, η μετάβαση από οριζόντια σε κάθετη στοίβαξη κυκλωματικών στοιχείων δίνει τη δυνατότητα να επαναπροσδιοριστούν οι συμβάσεις στο σχεδιασμό υλικού.



Σχήμα 4.1: Διαφορές στο μήκος διασύνδεσης για το (α) μία συσκευή 2-D, (β) μια 3-D αρχιτεκτονική με δύο στρώματα και (γ) μία 3-D αρχιτεκτονική με τέσσερα στρώματα.

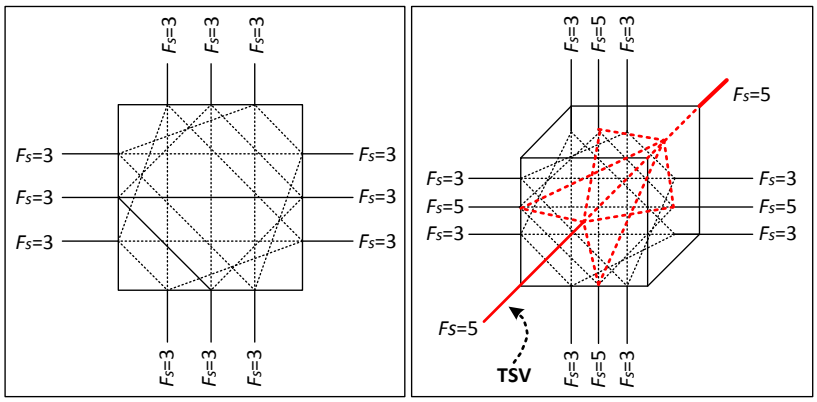
4.1.1 3-D Επαναδιαμορφούμενες πλατφόρμες

Υπάρχουν δύο προσεγγίσεις ενσωμάτωσης για την κατασκευή 3-D FPGAs. Στην πρώτη προσέγγιση, κάθε φυσικό στρώμα αντιμετωπίζεται ως ένα 2-D FPGA και η επικοινωνία μεταξύ των στρωμάτων παρέχεται από 3-D SwitchBoxes (SBs). Ένα παράδειγμα αυτής της κατασκευαστικής προσέγγισης παρουσιάζεται στο [65] όπου μία 3D αρχιτεκτονική με πέντε στρωμάτα μείωσε το area×delay product μιας 2-D FPGA αρχιτεκτονικής κατά 36%.

Εναλλακτικά, κάθε στρώση μπορεί να περιλαμβάνει ένα μόνο λειτουργικό στοιχείο της αρχιτεκτονικής του FPGA, όπως μνήμη, I/Os, ή λογικές πύλες (που οδηγεί, στην περίπτωση αυτή, σε τριών-στρωμάτων 3-D FPGA). Πειραματικά αποτελέσματα δείχνουν ότι αυτή η συσκευή επιτυγχάνει μια 1,7× βελτίωση σε απόδοση σε σύγκριση με ένα 2-D FPGA [66]. Ο αριθμός κάθε στοιχείου (π.χ. λογική, μνήμη, I/O) δεν αυξάνεται με την ίδια αναλογία όπως αυξάνεται το μέγεθος του FPGA. Αυτή η διαφορά μπορεί να οδηγήσει σε ανόμοια περιοχή πυριτίου για κάθε στρώμα, το οποίο με τη σειρά του, οδηγεί σε σπατάλη πυριτίου (στρώματα ίδιου εμβαδού προτιμώνται από κατασκευαστικής άποψης). Επιπλέον, κάθε στρώμα θα απαιτήσει ένα διαφορετικό σύνολο από μάσκες, αυξάνοντας έτσι το κόστος κατασκευής. Η επαναχρησιμοποίηση της ίδιας (πλήρους ή μερικής) μάσκας μπορεί να μειώσει το κόστος, το οποίο είναι ένα θεμελιώδες χαρακτηριστικό της κατασκευής των FPGA.

Η εικόνα 4.2 απεικονίζει τις διαφορές μεταξύ των συμβατικών SBs (που βρίσκονται σε 2-D FPGA) και 3-D SBs. Ένα 2-D SB μπορεί να χρησιμοποιηθεί όταν ένα εισερχόμενο καλώδιο δρομολόγησης συνδέεται με καλώδια του ίδιου στρώματος ($F_s = 3$). Η ευελιξία των SB, F_s υποδηλώνει τον αριθμό των κατευθύνσεων στα οποία μπορεί να συνδεθεί κάθε εισερχόμενο καλώδιο. Εναλλακτικά τα 3-D SBs υποστηρίζουν συνδέσεις στην τρίτη διάσταση (άνω και κάτω στρώματα, $F_s = 5$), εκτός από αυτά που βρίσκονται στα τελευταία άνω και κάτω στρώματα, όπου $F_s = 4$.

Επιπρόσθετα στις προαναφερθείσες προσεγγίσεις για 3-D FPGAs, οι συσκευές αυτές μπορούν να υποστηρίξουν διάφορους τύπους SBs, λόγω της επιπρόσθετης ελευθερίας



(a) 2-D Switch Box (b) 3-D Switch Box

Σχήμα 4.2: Template διαφορετικών τύπων SBs: (a) 2-D SB and (b) 3-D SB.

σχεδιασμού όπου η τρίτη φυσική διάσταση εισάγει [67]. Αυτοί οι διαφορετικοί τύποι των SB είναι δυνατό, με τη σειρά τους, να χρησιμοποιηθούν για να μειωθεί ο αριθμός των TSVs για τις συνδέσεις των διαφορετικών στρωμάτων [65]. Μια σύγκριση μεταξύ των δύο προσεγγίσεων (αρχιτεκτονικές με ίδια ή διαφορετικά SBs) έχει δείξει ότι οι τελευταίες απαιτούν λιγότερα TSVs για να υλοποιήσουν μια εφαρμογή [58]. Επιπλέον, ένα πρωτότυπο ασύγχρονο τρισδιάστατο FPGA έχει καταδείξει τη δυνατότητα τέτοιων συσκευών [68], στις οποίες οι πόροι λογικής αυτής της αρχιτεκτονικής είναι ίδιοι με το αντίστοιχο 2-D σχέδιο, ενώ τα SBs υποστηρίζουν και κανάλια επικοινωνίας μεταξύ στρωμάτων.

Μία διαφορετική προσέγγιση στο σχεδιασμό ενός 3-D FPGA συζητείται στο [58]. Η ετερογένεια εισάγεται στο δίκτυο διασύνδεσης και όχι στην κατανομή των διαφόρων συστατικών του κυκλώματος που συνθέτουν ένα FPGA. Η βασική ιδέα ήταν να συνδυαστούν υπάρχοντα 2-D [5] [67] και 3-D [3] [69] SBs, έτσι ώστε οι κάθετες διασυνδέσεις να χρησιμοποιηθούν πιο αποτελεσματικά. Με την αποτελεσματική αξιοποίηση των κάθετων διασυνδέσεων, ένα 3-D FPGA εμφανίζει συγκρίσιμη ή ανώτερη απόδοση σε σχέση με άλλες 3-D FPGA προσεγγίσεις [3] [69] [70] [68] [71] [66]. Από κατασκευαστικής άποψης, λιγότερες συνδέσεις στα ενδιάμεσα στρώματα μέσα σε ένα 3-D FPGA έχουν ως αποτέλεσμα την μείωση της επιφάνειας. Αυτό, με τη σειρά του, μειώνει το κόστος κατασκευής, ενώ το πρόσθετο πυρίτιο μέσα σε κάθε στρώμα μπορεί να χρησιμοποιηθεί για λογικά μπλοκ [20] [21].

4.1.2 CAD Αλγόριθμοι για 3-D Επαναδιαμορφούμενες Αρχιτεκτονικές

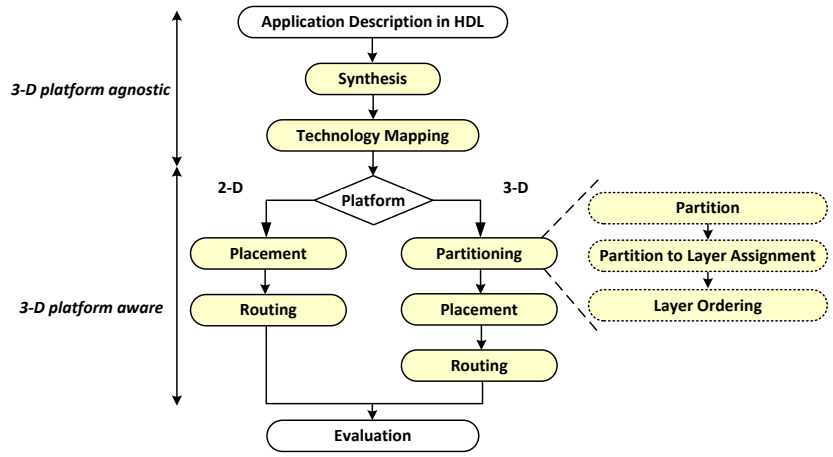
Τα εργαλεία Computer-aided design (CAD) καθιστούν τώρα πιθανό να αυτοματοποιηθούν πολλές πτυχές της διαδικασίας σχεδιασμού. Αυτό έχει γίνει κυρίως δυνατόν με την χρήση αποτελεσματικών και αποδοτικών αλγόριθμων και αντίστοιχων δομών λογισμικού. Η very large scale integration (VLSI) είναι εξαιρετικά σύνθετη και ακόμα και μετά από το σπάσιμο της διαδικασίας σε διάφορα εννοιολογικά ευκολότερα βήματα, το

κάθε βήμα είναι ακόμα υπολογιστικά πολύ βαρύ. Αν και διάφορα EDA (Electronic Design Automation) εργαλεία παρέχουν αυτοματοποιημένη υλοποίηση μιας εφαρμογής επάνω σε πλατφόρμες υλικού, η εκτέλεσή τους απαιτεί έναν πολύ μεγάλο χρόνο εκτέλεσης. Αυτό το πρόβλημα γίνεται εντονότερο όταν λάβει κανείς υπόψη ότι ο αριθμός των πόρων σε επίπεδο λογικής αυξάνεται σταθερά σε ποσοστό που προσδοκάται από το νόμο Moore. Ως εκ τούτου, τα EDA εργαλεία πρέπει να συνθέσουν, να τοποθετήσουν και να δρομολογήσουν περισσότερα στοιχεία λογικής και δίκτυα διασύνδεσης με κάθε νέα πλατφόρμα. Λαμβάνοντας υπόψη την αυξανόμενη πολυπλοκότητα των εφαρμογών που υλοποιούνται σε FPGAs, αναμένεται ότι τα εργαλεία σχεδιασμού θα δυσκολευτούν πολύ να παραδώσουν βελτιστοποιημένες λύσεις στα πλαίσια πρακτικών προϋπολογισμών χρόνου εκτέλεσης.

Ο χρόνος compile έχει πρόσφατα αναγνωριστεί ως ένα σημαντικό ζήτημα για FPGAs. Υπάρχουν αρκετοί σχεδιαστές που είναι πρόθυμοι να δεχτούν μία μείωση στην ποιότητα των αποτελεσμάτων (π.χ. ποινή στην απόδοση) σε αντάλλαγμα για compilation υψηλής ταχύτητας [37]. Επιπλέον, καθώς η χωρητικότητα των συσκευών FPGA και το μέγεθος των εφαρμογών μεγαλώνουν, υπάρχει μεγάλο ενδιαφέρον για τη μείωση του χρόνου εκτέλεσης των εργαλείων CAD που στοχεύουν σε επαναδιαμορφούμενες πλατφόρμες.

Για να διατηρηθεί ο χρόνος εκτέλεσης υπό έλεγχο, οι δύο κύριες εταιρείες που προσφέρουν FPGAs υψηλής χωρητικότητας, οι Xilinx και Altera, βελτιστοποιούν συνεχώς τα CAD εργαλεία τους. Ακόμα κι αν τέτοιες στρατηγικές βοηθούν στον χρόνο εκτέλεσης, είναι απίθανο οι εν λόγω αλγόριθμοι να βελτιωθούν σε ποσοστό που απαιτείται από πολλές ακόμη γενιές του νόμου του Moore. Η συνεχής κλιμάκωση της τεχνολογίας, χωρίς συγκρίσιμα κλιμάκωση του χρόνου εκτέλεσης για την υλοποίηση εφαρμογών σε συσκευές FPGA αναμένεται να οδηγήσει σε μια run-time κρίση. Η κρίση αυτή, μεταξύ άλλων, εκδηλώνεται ως μια μείωση της παραγωγικότητας και αντίστοιχη αύξηση στα NRE κόστη. Με βάση τις σχετικές ερευνητικές προσεγγίσεις, υπάρχουν τρεις τρόποι για να μειωθεί η εκτέλεση του χρόνου εκτέλεσης των εργαλείων CAD ως εξής:

- Αποφυγή επίπεδου compilation ολόκληρου του σχεδιασμού και υποχρέωση των χρηστών να κάνουν compile των σχεδίων τους αυξητικά και συναρμολόγηση των κομματιών στο τέλος. Ακόμα κι αν αυτή η προσέγγιση μετριάζει τον χρόνο εκτέλεσης, επιβάλλει αυξημένη πολυπλοκότητα στο σχεδιασμό, ενώ επίσης δεν επιτρέπει βελτιστοποιήσεις που πρέπει να εφαρμόζονται μεταξύ των κομματιών της εφαρμογής.
- Προσπάθεια να βρεθούν πιο γρήγοροι μονονηματικοί αλγόριθμοι, το οποίο μπορεί να οδηγήσει σε σημαντική επιτάχυνση της εκτέλεσης με μια μικρή ποινή στην ποιότητα της υλοποίηση εφαρμογών.

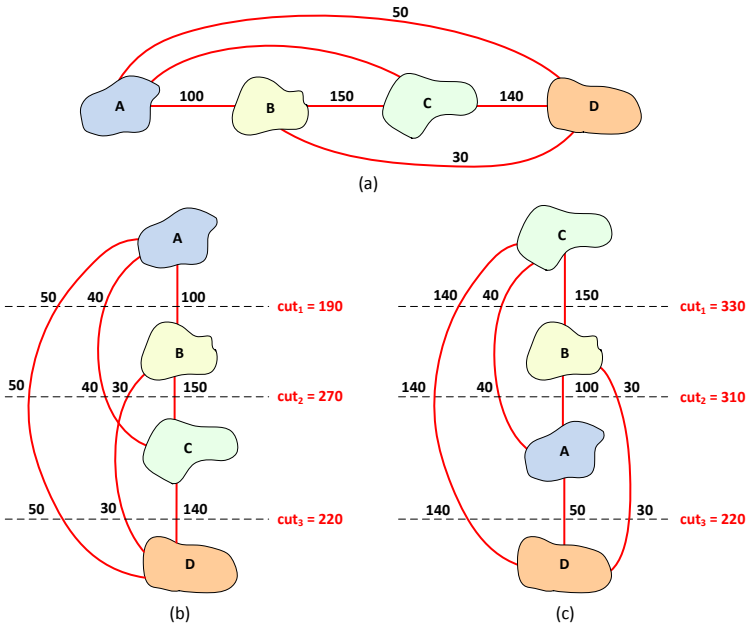


Σχήμα 4.3: Toolflow για την εκτέλεση mapping της εφαρμογής σε πλατφόρμες FPGA.

- Ανάπτυξη νέων παράλληλων αλγόριθμων, που επωφελούνται από τους παρόντες και επερχόμενους multi-core επεξεργαστές. Με την τρέχουσα τάση της αγοράς για αύξηση του αριθμού των πυρήνων της CPU αντί για το σχεδιασμό ταχύτερων πυρήνων CPU, η χρήση των παράλληλων αλγόριθμων CAD υπόσχεται να μειώσει σημαντικά τον χρόνο εκτέλεσης. Αυτοί οι αλγόριθμοι επιτρέπουν η χωρητικότητα των FPGA και ο αριθμός των πυρήνων του επεξεργαστή, να κλιμακωθούν παράλληλα. Προς αυτή την κατεύθυνση, τόσο η Xilinx όσο και η Altera έχουν αρχίσει να υλοποιούν παράλληλες εκδοχές για τους CAD αλγορίθμους τους που προσφέρουν κάποιες βελτιώσεις στον χρόνο εκτέλεσης.

Προκειμένου να μειωθεί η πολυπλοκότητα της διαδικασίας σχεδιασμού, διάφορα ενδιάμεσα επίπεδα αφαίρεσης εισάγονται. Μια μεθοδολογία σχεδιασμού top-down χωρίζει το σύνολο του σχεδιασμού σε έναν αριθμό από διακριτές φάσεις, όπως απεικονίζεται στο Σχήμα 4.3. Ξεκινώντας από την netlist μιας εφαρμογής μετά τη σύνθεση και το technology mapping (3-D platform agnostic αλγόριθμοι), η μεθοδολογία προχωρά στην υλοποίηση εφαρμογών πάνω στην επιλεγμένη 3-D πλατφόρμα. Αυτή η διαδικασία αποτελείται από τρεις διαδοχικές εργασίες, (i) partitioning της εφαρμογής, (ii) τοποθέτηση της εφαρμογής και (iii) δρομολόγηση.

Το partitioning της εφαρμογής είναι ζωτικής σημασίας τόσο για την επίτευξη υψηλότερης απόδοσης όσο και για το βαθμό χρησιμοποίησης των πόρων. Για να υποστηριχθεί το βήμα αυτό, μια σειρά αλγορίθμων και εργαλείων έχουν προταθεί. Ακολούθως, τα partitions ανατίθενται σε διαθέσιμα φυσικά στρώματα της 3-D αρχιτεκτονικής. Συνήθως, ο πρωταρχικός στόχος αυτού του βήματος είναι να ελαχιστοποιηθούν οι συνδέσεις μεταξύ των ήδη υπαρχόντων επιπέδων. Στη συνέχεια, τα στρώματα πρέπει να διαταχθούν προκειμένου να σχηματιστεί η 3-D στοίβα. Παρόμοια με τα προηγούμενα, το βήμα αυτό έχει ως στόχο να ελαχιστοποιηθούν οι συνδέσεις στα ενδιάμεσα στρώματα και ως εκ τούτου στα TSVs, μέσω της κατάλληλης σειράς των μερών της εφαρμογής στα διαθέσιμα επίπεδα. Επιπλέον, η διαδικασία της επιλογής της σειράς των στρωμάτων



Σχήμα 4.4: Γράφημα για την υλοποίηση εφαρμογών σε 3-D αρχιτεκτονικές: (α) partitioning της εφαρμογής και (β) - (γ) εναλλακτικές 3-D στοίβες που προέρχονται από διαφορετική ανάθεσης partition σε 3D στρώματα και την σειρά τοποθέτησης των στρωμάτων

μπορεί να αντιμετωπίσει μια σειρά από σημαντικά θέματα του σχεδιασμού, όπως βελτίωση της θερμικής διανομής, ενισχύοντας έτσι την αξιοπιστία της 3-D στοίβας. Ακόμα κι αν το hypergraph partitioning είναι ένα καλά μελετημένο πρόβλημα, αυτοί οι αλγόριθμοι σπανίως οδηγούν σε ικανοποιητικά αποτελέσματα, επειδή δεν λαμβάνουν υπόψη εγγενείς περιορισμούς που τίθενται από τις 3-D πλατφόρμες. Συγκεκριμένα, η προσέγγιση αυτών των αλγορίθμων στοχεύει αποκλειστικά και μόνο για την αντιμετώπιση του πρόβληματος του διαμερισμού, ενώ δεν αντιμετωπίζουν καθόλου το πρόβλημα της ανάθεσης partition σε 3D στρώματα και την σειρά τοποθέτησης των στρωμάτων.

Για να τονιστεί η σημασία του partitioning, της ανάθεσης partition σε 3D στρώματα και της σειράς τοποθέτησης των στρωμάτων, η Εικόνα 4.4 δίνει ένα παράδειγμα ενός ψηφιακού κυκλώματος υλοποιημένο επάνω σε μία πλατφόρμα 3-D. Σε αυτό το σχήμα, δύο εναλλακτικές 3-D στοίβες (φαίνεται στα Σχήματα 4.4 (β) και 4.4 (γ)) αξιολογούνται μετά την επιτυχή εφαρμογή partitioning σε τέσσερα τμήματα, ήτοι A, B, C και D (όπως απεικονίζεται στο Σχήμα 4.4 (α)). Η αξιολόγηση των αποτελεσμάτων που λαμβάνονται για την ανάλυση αυτή είναι με βάση την *net-cut* παράμετρο. Ο όρος αυτός αναφέρεται στην δρομολόγηση της εφαρμογής με δίκτυα μεταξύ διαφορετικών partitions (στρωμάτων). Ακόμα κι αν οι στοίβες 3-D που απεικονίζονται σε αυτό το σχήμα ανακτώνται από το ίδιο partitioning, οι μεταβολές στην ανάθεση partition σε 3D στρώματα και η σειρά τοποθέτησης των στρωμάτων οδηγούν σε διαφορετικά net-cuts. Επιπλέον, τα net-cuts μεταξύ συνεχόμενων στρωμάτων, $\left(\sum_{i=1}^{i=3}(cut_i)\right)$, που αντιστοιχούν στον αριθμό των TSV συνδέσεων, επηρεάζονται και αυτά από την έξοδο του partitioning.

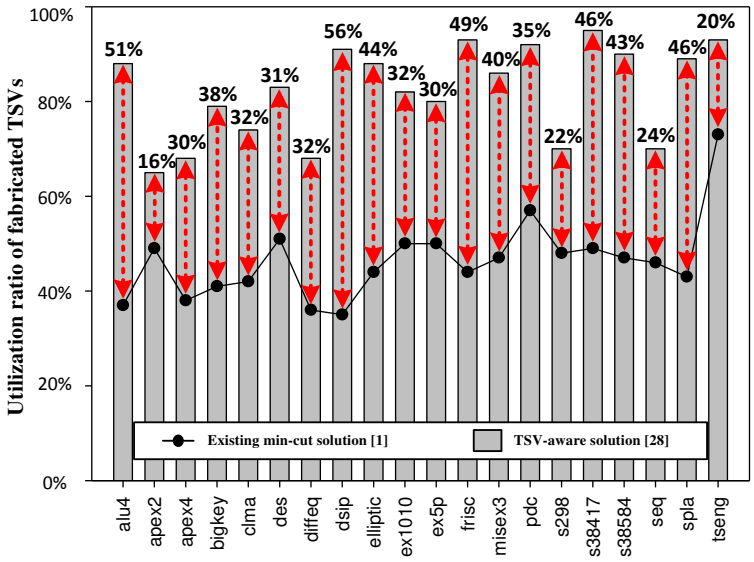
Το τελευταίο βήμα αφορά την τοποθέτηση της εφαρμογής και την δρομολόγηση (P&R) στην 3-D FPGA πλατφόρμα. Κάθε μία από αυτές τις εργασίες πραγματοποιείται ταυτόχρονα για όλα τα στρώματα της αρχιτεκτονικής, έτσι ώστε οι κατάλληλοι περιορισμοί να διαδίδονται μεταξύ των στρωμάτων. Προκειμένου να επιτευχθεί μια λύση που θα εξισορροπεί την αξιοποίηση των πόρων του υλικού με την βελτίωση των επιδόσεων, οι συνδέσεις των ενδιάμεσων στρωμάτων θα πρέπει να χρησιμοποιούνται μόνο για τα χρονικά κρίσιμα μονοπάτια δρομολόγησης.

4.1.3 Partitioning Εφαρμογής

Για πολλές υφιστάμενες και νέες εφαρμογές σε VLSI (Very Large Scale Integration), το αποτελεσματικό partitioning έχει μεγάλη σημασία. Το πρόβλημα είναι να γίνει partitioning στις κορυφές ενός hypergraph σε k περίπου ίσα μέρη, έτσι ώστε ο αριθμός των hyperedges που συνδέει κορυφές σε διαφορετικά μέρη να ελαχιστοποιείται. Ένας hypergraph είναι μια γενίκευση ενός γραφήματος, όπου το σύνολο των ακμών αντικαθίσταται από ένα σύνολο hyperedges. Συγκεκριμένα, μία hyperedge επεκτείνει την έννοια της ακμής επιτρέποντας περισσότερες από δύο κορυφές να συνδεθούν με μια hyperedge. Τυπικά, ένας hypergraph $H = (V, E^h)$ ορίζεται ως ένα σύνολο κορυφών VE και ένα σύνολο hyperedges E^h, όπου κάθε hyperedge είναι ένα υποσύνολο του συνόλου V κορυφών και το μέγεθος μιας hyperedge είναι το πλήθος αυτού του υποσυνόλου.

Το partitioning ενός κυκλώματος είναι NP-hard πρόβλημα [72]. Δηλαδή, καθώς το μέγεθος του προβλήματος αυξάνεται γραμμικά, η προσπάθεια που χρειάζεται για να βρούμε μια βέλτιστη λύση αναπτύσσεται με ταχύτερο ρυθμό από οποιαδήποτε πολυωνυμική συνάρτηση. Μέχρι σήμερα, δεν υπάρχει κανένας γνωστός πολυωνυμικού χρόνου, συνολικά βέλτιστος αλγόριθμος για balance constrained partitioning. Ωστόσο, αρκετές αποτελεσματικές ευριστικές μέθοδοι έχουν αναπτυχθεί στο παρελθόν. Αυτοί οι αλγόριθμοι μπορούν να δώσουν ένα κύκλωμα με αρκετά καλό partitioning και για πρακτικούς λόγους τρέχουν σε χαμηλής τάξης πολυωνυμικό χρόνο. Τυπικό παράδειγμα τέτοιων αλγορίθμων είναι ο Fiduccia-Mattheyses (FM) [73], ο Kernighan-Lin (KL) [74], ο hMetis [72], ο annealing/tabu [4], καθώς και διάφορες επεκτάσεις αυτών.

Όσον αφορά τους αλγορίθμους partitioning για 3-D FPGA πλατφόρμες [3] [69] [72] η πλειοψηφία τους επικεντρώνονται στην ανάκτηση μιας min-cut λύσης. Στη θεωρία γραφημάτων, μια ελάχιστη περικοπή ενός γραφήματος είναι μια περικοπή (ένα partition των κορυφών ενός γράφου σε δύο ανεξάρτητα υποσύνολα που ενώνονται με τουλάχιστον μία ακμή) του οποίου το cut set έχει το μικρότερο αριθμό ακμών (μη σταθμισμένη περίπτωση), ή μικρότερο άθροισμα από βάρη. Αντί να χρησιμοποιηθούν πολλά TSVs προκειμένου να επιτευχθεί η μέγιστη βελτίωση στις επιδόσεις, τα min-cut του αλγόριθμου εκμεταλλεύονται την μείωση της χρησιμοποιούμενης κάθετης διασύνδεσης. Ακόμα κι αν αυτός ο στόχος είναι αποδεκτός για συσκευές multi-chip, σπάνια μπορεί να θεωρηθεί ως μια αποτελεσματική προσέγγιση στην εκμετάλλευση των αρχιτεκτονικών 3-D,

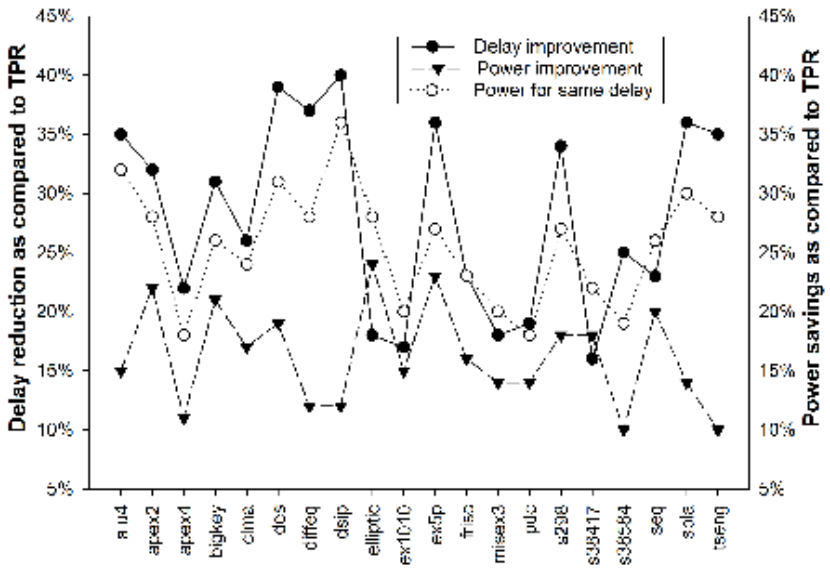


Σχήμα 4.5: Αξιοποίηση των TSVs με τη χρήση ενός min-cut [3] και ενός max-cut [4] αλγόριθμου για 3-D FPGAs

αφού τα ηλεκτρικά χαρακτηριστικά των TSVs είναι διαφορετικά από ό,τι τα αντίστοιχα για off-chip συνδέσεις. Μια πρώτη σύγκριση μεταξύ καλωδίων δρομολόγησης στα 45nm και στα TSV μπορεί να βρεθεί στο [75].

Η εικόνα 4.5 (συνεχής γραμμή) ποσοτικοποιεί την αξιοποίηση των κατασκευασμένων TSVs στα 20 μεγαλύτερα MCNC benchmarks. Η πειραματική διάταξη συνοψίζεται ως εξής: τα 3-D FPGAs αποτελούνται από τρία στρώματα, η λογική και η δρομολόγηση των πόρων μεταξύ των στρωμάτων είναι πανομοιότυπες, οι συνδέσεις μεταξύ των στρωμάτων (TSVs) γίνονται μέσα από 3-D Switchbox (SBs) και υπάρχουν τέσσερα TSVs ανά 3-D SB. Ο κάθετος άξονας σε αυτό το σχήμα δίνει με κανονικοποιημένο τρόπο το ποσοστό της χρησιμοποίησης του συνόλου των TSVs. Για τη μελέτη αυτή, η τοποθέτηση και εφαρμογή δρομολόγησης (P&R) εκτελείται με εργαλεία από [3].

Με βάση το Σχήμα 4.5, η χρήση τις min-cut προσέγγισης οδηγεί σε μια μέση χρησιμοποίηση των TSVs περίπου 46% (54% των κατασκευασμένων TSVs παραμένουν αναξιοποίητα), λαμβάνοντας υπόψη ότι η αναλογία αυτή είναι σχεδόν σταθερή στα benchmarks. Με άλλα λόγια, η απόδοση των 3-D FPGAs δεν βελτιώνεται από συσκευές που ενσωματώνουν με υψηλότερη πυκνότητα κάθετων συνδέσεων. Ως εκ τούτου, σημαντική βελτίωση των επιδόσεων είναι δυνατή, αν υπολογίζουμε cuts που χρησιμοποιούν πιο αποτελεσματικά τα διαθέσιμα (κατασκευασμένα) TSVs. Δεν είναι όλες τις λειτουργίες της εφαρμογής υποψήφιες για να ανατεθούν σε διαφορετικά partitions. Συγκεκριμένα, είναι πιθανό λειτουργίες που ανήκουν σε κρίσιμα δίκτυα (π.χ. Δίκτυα με αυξημένη δραστηριότητα μεταγωγής) να τοποθετηθούν στο ίδιο επίπεδο, δεδομένου ότι αυτό οδηγεί σε μεγαλύτερη βελτίωση των επιδόσεων.



Σχήμα 4.6: Καθυστέρηση και εξοικονόμηση ενέργειας max-cut partitioning σε σύγκριση με το εργαλείο TPR. [3].

Σε αντίθεση με αυτή την προσέγγιση, στο Σχήμα 4.5 τα αποτελέσματα για partitioning εφαρμογής παρέχονται με τη χρήση μιας max-cut υλοποίησης [4]. Με βάση αυτά τα αποτελέσματα, οι συγγραφείς καταλήγουν στο συμπέρασμα ότι μια τέτοια προσέγγιση οδηγεί σε partitions που παρουσιάζουν μεγαλύτερη χρησιμοποίηση της κάθετης συνδεσιμότητας σε σύγκριση με τα υπάρχοντα εργαλεία CAD που στοχεύουν σε 3-D FPGAs. Πιο συγκεκριμένα, ο TSV-aware αλγόριθμος χρησιμοποιεί κατά μέσο όρο 82% των κατασκευασμένων TSVs, το οποίο με τη σειρά του είναι 36% υψηλότερο από ό,τι η αντίστοιχη αναλογία από υπάρχοντα εργαλεία [3].

Ο στόχος αυτού του αλγορίθμου είναι να χρησιμοποιήσει με έναν πιο αποτελεσματικό τρόπο τις διαθέσιμες συνδέσεις μεταξύ των στρωμάτων, προκειμένου να επιτευχθεί η μέγιστη δυνατή βελτίωση στην απόδοση. Για να αξιολογηθεί η βελτίωση της απόδοσης από ένα τέτοιο partitioning, το Σχήμα 4.6 απεικονίζει με κανονικοποιημένο τρόπο την καθυστέρηση και την εξοικονόμηση ενέργειας για τις εναλλακτικές προσεγγίσεις. Για λόγους επίδειξης, οι τιμές για τον TSV-aware αλγόριθμο κανονικοποιούνται με τις αντίστοιχες τιμές που ανακτώνται από ένα min-cut εργαλείο (TPR) [3].

Με βάση αυτό το γράφημα, το max-cut partitioning οδηγεί σε μέση μείωση της καθυστέρησης και κατανάλωσης ισχύος περίπου 28% και 17%, αντίστοιχα, σε σύγκριση με την συμβατική min-cut προσέγγιση που χρησιμοποιείται σε VLSI κυκλώματα. Αυτά τα κέρδη παρατηρούνται κυρίως λόγω της υψηλότερης και της πιο αποτελεσματικής αξιοποίησης των κατασκευασμένων TSVs, η οποία με τη σειρά της οδηγεί σε μικρότερα μονοπάτια στα κρίσιμα δίκτυα. Στόχος μιας τέτοιας προσέγγισης δεν μπορεί να είναι αποκλειστικά και μόνο η μεγιστοποίηση της χρησιμοποίησης των TSVs, διότι θα μπορούσε να οδηγήσει σε unroutable λύσεις (λόγω προβλημάτων συμφόρησης δρομολόγησης). Μία

ακόμα σύγκριση φαίνεται στην Εικόνα 4.6. Οι λύσεις που χαρακτηρίζονται ως "Power for same delay" αξιολογούνται ως προς την μέγιστη δυνατή εξοικονόμηση ενέργειας όταν έχει ρυθμιστεί η συχνότητα λειτουργίας να είναι ίση με εκείνη που προκύπτει με min-cut προσέγγιση. Από την ανάλυση αυτή, το TSV-aware partitioning οδηγεί σε μέση εξοικονόμηση ενέργειας περίπου 26% για τις ίδιες συχνότητες λειτουργίας. Ως εκ τούτου, μια τέτοια λύση μπορεί επίσης να χρησιμοποιηθεί ως power-aware partitioning αλγόριθμος.

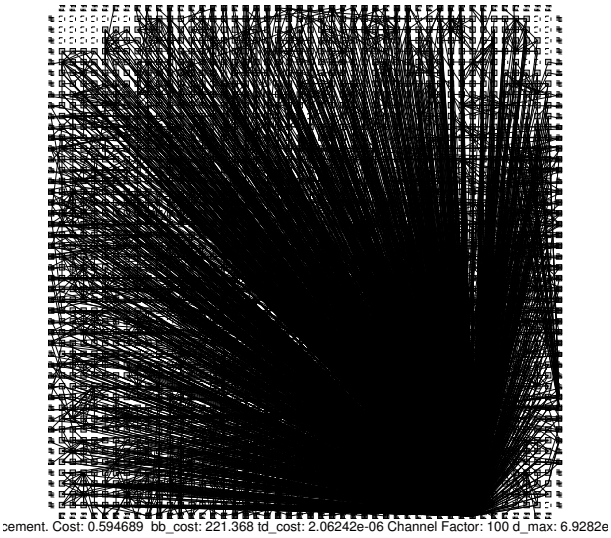
Το υπόλοιπο αυτού του κεφαλαίου είναι οργανωμένο ως εξής: Η ενότητα 2 περιγράφει τα κίνητρα πίσω από αυτή την ερευνητική δουλειά, ενώ η ενότητα 3 παρουσιάζει την προτεινόμενη ετερογενή 3-D αρχιτεκτονική. Το framework για την εξερεύνηση σε επίπεδο αρχιτεκτονικής, καθώς και το mapping μιας εφαρμογή πάνω σε αυτές τις συσκευές περιγράφονται στην ενότητα 4. Τα Πειραματικά αποτελέσματα που δείχνουν την αποτελεσματικότητα της προτεινόμενης λύσης αναφέρονται στην ενότητα 5. Τέλος, τα συμπεράσματα που συνοψίζονται στην ενότητα 6.

4.2 Κίνητρα

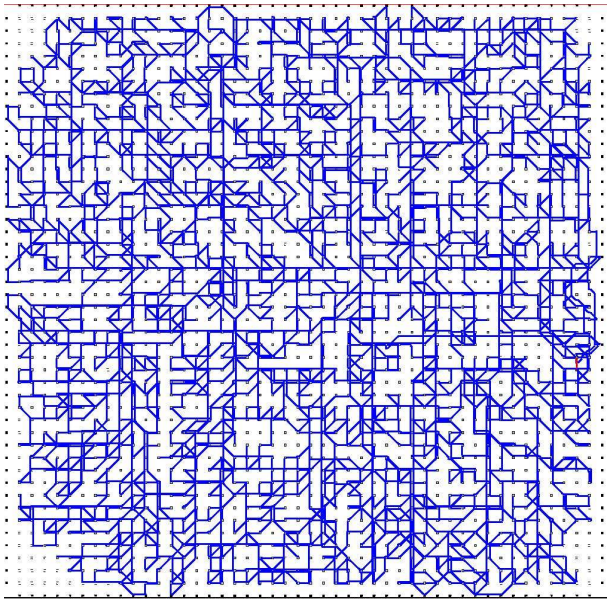
Η πολυπλοκότητα των εφαρμογών εισάγει συνήθως περιορισμούς στην αρχιτεκτονική οργάνωση των FPGA. Ακόμη και αν η ζήτηση για επιπλέον λογικούς πόρους ικανοποιείται με πλατφόρμες που αποτελούνται από πιο πολύπλοκα λογικά μπλοκ, ή CLBs, (π.χ. με περισσότερα LUTs), το πρόβλημα αυτό εξακολουθεί να υφίσταται με τις πιο απαιτητικές σε θέμα επικοινωνίας εφαρμογές (π.χ. τηλεπικοινωνίες, κρυπτογράφηση και την επεξεργασία εικόνας, βίντεο), δεδομένου ότι η απόδοσή τους εξαρτάται συνήθως από τη διαθεσιμότητα σε I/O bandwidth.

Τα σημερινά FPGAs έχουν περιορισμένο αριθμό από I/O, τα οποία χωρικά βρίσκονται στην περιφέρεια της συσκευής [5]. Μια τέτοια τοπολογική διάταξη των I/O pads επιβάλλει ότι τα δίκτυα διασύνδεσης που παρέχουν μετάδοση σημάτων μεταξύ λογικών μπλοκ και I/O pads εμφανίζουν αυξημένο wirelength, αντίσταση και τιμές χωρητικότητας. Ως εκ τούτου, τα δίκτυα αυτά συνήθως δρομολογούνται από συμφορημένες περιοχές της συσκευής και ως εκ τούτου, κυριαρχούν στην καθυστέρηση της εφαρμογής (μέγιστη συχνότητα λειτουργίας) ή/και στην κατανάλωση ενέργειας. Ένα τέτοιο πρόβλημα γίνεται πολύ πιο σημαντικό στις εφαρμογές που στηρίζονται στην ταχύτητα της επικοινωνίας, όπου η πλειοψηφία των πόρων που χρησιμοποιούνται για την διασύνδεση επηρεάζουν τα μονοπάτια δρομολόγησης μεταξύ CLBs και I/O pads.

Αυτός ο περιορισμός, φαίνεται από το Σχήμα 4.7 που δείχνει την επιτυχή τοποθέτηση και δρομολόγηση για ένα I/O-bound benchmark (όνομα $bigkey$), ενώ η Εικ. 4.8 δείχνει την δρομολόγηση για το πιο κρίσιμο μονοπάτι (μετά την ολοκλήρωση και της δρομολόγησης). Η αρχιτεκτονική που μελετάται είναι ένα συμβατικό island-style 2-D FPGA, παρόμοιο με συσκευές Xilinx Virtex, ενώ η τοποθέτηση και δρομολόγηση πραγματοποιήθηκε με το εργαλείο VPR [5]. Αυτό το εργαλείο χρησιμοποιεί έναν timing-driven αλγόριθμο

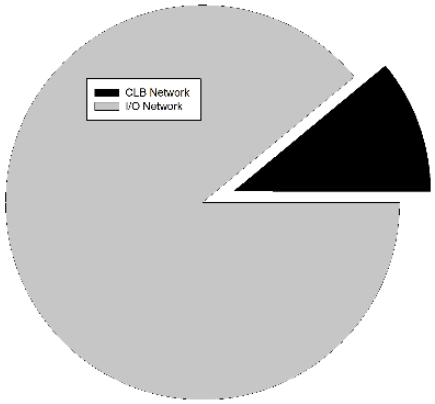


Σχήμα 4.7: Επιτυχής τοποθέτηση και δρομολόγηση με το εργαλείο VPR για το $bigkey$ benchmark.



Σχήμα 4.8: Δρομολόγηση για το critical path για το $bigkey$ benchmark.

τοποθέτησης που εκχωρεί λειτουργίες της εφαρμογής στα διαθέσιμα μπλοκ του FPGA (λογικά και I/O) και έχει ως στόχο την επίτευξη μιας όσο το δυνατόν περισσότερο συμπαγούς σχεδίασης. Ως εκ τούτου, προκύπτουν μικρότερες συνδέσεις που με τη σειρά τους αντιστοιχούν σε βελτιώσεις στην καθυστέρηση και στην κατανάλωση ισχύος. Και στα δύο σχήματα, τα λογικά μπλοκ που χρησιμοποιούνται σημειώνονται με γκρι χρώμα, ενώ οι συνδέσεις μεταξύ τους (όπως ανακτήθηκε μετά την διαδικασία δρομολόγησης) απεικονίζονται με μαύρες (Εικ. 4.7.) και μπλε (Εικ. 4.8) γραμμές.



Σχήμα 4.9: Μέσος όρος μήκους καλωδίων για τα I/O δίκτυα και τα δίκτυα CLB στο $bigkey$ benchmark.

Με βάση τα σχήματα 4.7 και 4.8, βγαίνει το συμπέρασμα, ότι η πλειοψηφία των συνδέσεων επηρεάζουν τα μονοπάτια δρομολόγησης που παρέχουν μετάδοση σημάτων μεταξύ λογικών μπλοκ κατανεμημένων σε ολόκληρη την αρχιτεκτονική και μερικά I/O (τοποθετημένα κυρίως στην κάτω δεξιά γωνία της συσκευής). Αυτά τα μονοπάτια δρομολόγησης παρουσιάζουν αυξημένο μήκος και ως εκ τούτου παρουσιάζουν ως αποτέλεσμα, μεταξύ άλλων, αυξημένη καθυστέρηση και ισχύ. Για να τονιστεί περαιτέρω η σημασία αυτού του προβλήματος, το Σχήμα 4.9 παρέχει το μέσο μήκος σύρματος για τα δίκτυα I/O και CLB που αφορούν το $bigkey$ benchmark. Ειδικότερα, οι όροι *I/O network* και *CLB network* ορίζονται ως εξής:

- *I/O network*: Δίκτυο μιας εφαρμογής που περιέχει τουλάχιστον ένα I/O pad.
- *CLB network*: Δίκτυο μιας εφαρμογής που δεν περιέχει I/O (περιέχει αποκλειστικά CLBs).

Όπως συμπεραίνεται από το σχήμα 4.9, το μέσο μήκος σύρματος για I/O δίκτυα είναι περίπου 8$\times$ περισσότερο σε σύγκριση με το αντίστοιχο των δικτύων CLB, ενώ από το Σχ. 4.8 είναι σαφές ότι κατανέμονται σε όλη την επιφάνεια της συσκευής FPGA. Κατά συνέπεια, νέες αρχιτεκτονικές λύσεις είναι απολύτως απαραίτητες για την αντιμετώπιση των προβλημάτων που δημιουργούνται από τα I/O που βρίσκονται στην περιφέρεια της FPGA συσκευής.

Στο παρόν ερευνητικό έργο, προτείνεται μια νέα αρχιτεκτονική 3-D, καθώς και το απαραίτητο λογισμικό για τη διερεύνηση και την αξιολόγηση της υλοποίησης εφαρμογών πάνω σε τέτοιες πλατφόρμες. Πιο συγκεκριμένα, με την ανάθεση σε διαφορετικά επίπεδα της λογικής και των I/O πόρων, επιτυγχάνεται μείωση μήκους των μονοπατιών του κυκλώματος . Πειραματικά αποτελέσματα αποδεικνύουν την αποτελεσματικότητα της εν λόγω επιλογής, δεδομένου ότι η προτεινόμενη αρχιτεκτονική ξεπερνά σε επίδοση τα συμβατικά 2-D FPGAs. Επιπλέον, η ανάγκη για partitioning της εφαρμογής ξεπερνιέται, μειώνοντας έτσι τον χρόνο εκτέλεσης και την πολυπλοκότητα της μεθοδολογίας.

Πίνακας 4.1: Ποιοτική σύγκριση μεταξύ των εργαλείων για 3-D πλατφόρμες FPGA.

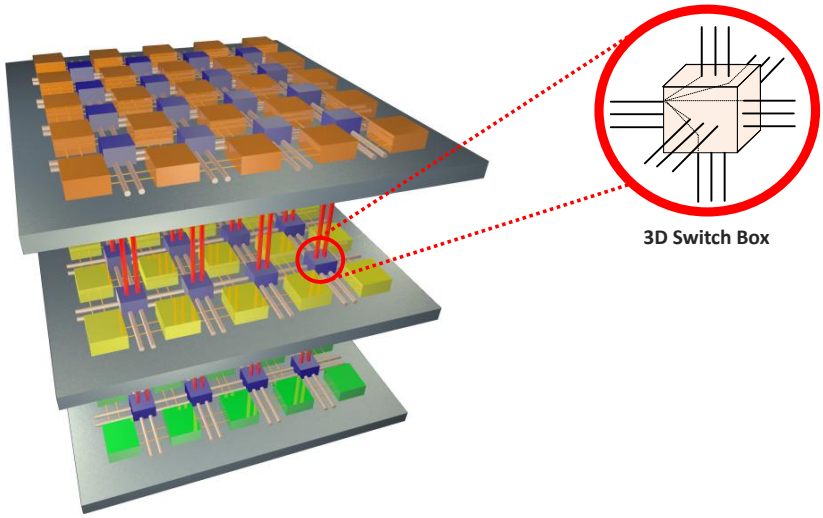
Λειτουργείες	3-D MEADER [58]	3-D NAROUTO [76]	VPR3D [22]	3D-Tree [77]
Ετερογενή στρώματα	OXI	NAI	OXI	OXI
Ετερογένεια μεταξύ στρωμάτων	OXI	NAI	OXI	OXI
Ετερογενής δρομολόγηση μεταξύ στρωμάτων	NAI	OXI	OXI	OXI
Memory/DSP blocks	OXI	NAI	NAI	OXI
3-D Τεχνολογία	WireBonding, Face-to-Face, TSV, SSIT	WireBonding, Face-to-Face, TSV, SSIT	SSIT	TSV
Εξερεύνηση σε επίπεδο αρχιτεκτονικής	NAI	NAI	NAI	NAI
Full-custom δρομολόγηση μεταξύ στρωμάτων	NAI	OXI	OXI	OXI
Partitioning	min-cut, max-cut	OXI need	min-cut	min-cut
Εκτίμηση Καθυστέρησης	NAI	NAI	NAI	NAI
Εκτίμηση ισχύος	NAI	NAI	NAI	NAI
GUI	NAI	NAI	NAI	NAI

Τα ιδιαίτερα χαρακτηριστικά, που αναφέρονται στην προηγούμενη παράγραφο και διαχωρίζουν την προτεινόμενη λύση από τα υπάρχοντα toolflows εμφανίζονται σε μια ποιοτική σύγκριση στον Πίνακα 4.1. Αυτά τα εργαλεία δεν υποστηρίζουν εμπορικές πλατφόρμες 3-D FPGA. Αντίθετα, χρησιμοποιούνται κυρίως για την εξερεύνηση σε επίπεδο αρχιτεκτονικής. Πιο συγκεκριμένα, κατά τη διάρκεια αυτής της φάσης, ένας αριθμός αρχιτεκτονικών παραμέτρων που αφορά τόσο τη λογική, όσο και τη δρομολόγηση της αρχιτεκτονικής αξιολογούνται με διάφορες μετρήσεις (π.χ. συχνότητα λειτουργίας, κατανάλωση ενέργειας, αξιοποίηση των πόρων, κλπ).

4.3 Μοντελοποίηση της Προτεινόμενης 3-D FPGA Αρχιτεκτονικής

Αυτή η ενότητα παρουσιάζει την προτεινόμενη αρχιτεκτονική για 3-D FPGAs, η οποία αναπτύχθηκε με στόχο να αντιμετωπιστεί το πρόβλημα των μεγάλου μήκους καλωδίων στα I/O δίκτυα. Η αρχιτεκτονική απεικονίζεται σχηματικά στο Σχ. 4.10, όπου η

λογική, η μνήμη και τα I/O μέσα σε αυτήν την αρχιτεκτονική έχουν εκχωρηθεί σε διαφορετικά στρώματα. Ο αριθμός των συνολικών στρωμάτων είναι παραμετροποιήσιμος, προκειμένου να καλύπτει όσο το δυνατόν καλύτερα τις απαιτήσεις των εφαρμογών. Το αρχιτεκτονικό πρότυπο που παρουσιάζεται σε αυτήν την ερευνητική εργασία είναι ορθογώνια ανεπτυγμένο ως προς τις υπάρχουσες 3-D FPGA προσεγγίσεις, όπου τα λογικά μπλοκ κατανέμονται μεταξύ των διαφορετικών στρωμάτων (π.χ. [78] [3] [79]).

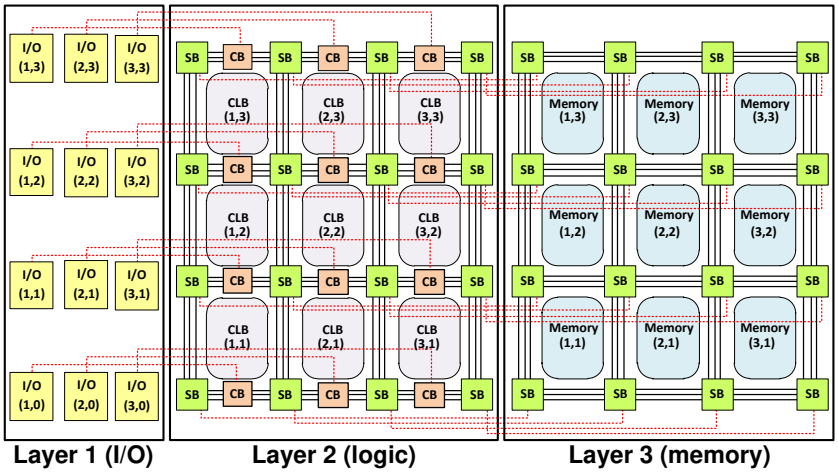


Σχήμα 4.10: Σχηματική απεικόνιση της προτεινόμενης αρχιτεκτονικής 3-D FPGA.

Η συνδεσιμότητα μεταξύ των στρωμάτων υλοποιείται μέσω κάθετων ευθυγραμμισμένων 3-D Switchbox (SBs). Για να είναι οι πόροι δρομολόγησης ενήμεροι σχετικά με την τρίτη διάσταση, έχουν κατάλληλα επεκταθεί με τον τρόπο που συζητήθηκε στο [80]. Πιο συγκεκριμένα, η συνδεσιμότητα μεταξύ I/O και λογικών στρώματων (Layers 1 και 2 στο Σχ. 4.10) υλοποιείται εντός 3-D Connection boxes (CB), ενώ τα 3-D SBs χρησιμοποιούνται για την συνδεσιμότητα μεταξύ των στρωμάτων που περιέχουν CLBs και μνήμες (Επίπεδα 2 και 3 στο σχήμα 4.10.) [80]. Προηγούμενες μελέτες έδειξαν την αποτελεσματικότητα του σχεδιασμού τέτοιων συστατικών [65]. Ωστόσο, δεδομένου ότι τα 3-D CBs επιβάλλουν τη χρήση TSVs, η προσεκτική χωροταξική τοποθέτηση τους είναι ένας κρίσιμος παράγοντας για την επίτευξη καλών επιδόσεων και αποτελεσματικές σε εμβαδόν λύσεις. Στην αρχιτεκτονική 3-D FPGA που παρουσιάζεται εδώ, έχουν εκχωρηθεί διαφορετικού τύπου μπλοκ σε κάθε 3-D στρώμα. Ο αριθμός των I/Os, καθώς και η χωρική τους θέση στο στρώμα τους, ανακτώνται από την εκτέλεση μιας εξερεύνησης επιπέδου αρχιτεκτονικής, έχοντας ως στόχο τη βελτιστοποίηση της καθυστέρησης, της κατανάλωσης ισχύος και το θερμικό στρες.

Το Σχ. 4.11 απεικονίζει σχηματικά το πρότυπο της προτεινόμενης αρχιτεκτονικής που αποτελείται από δύο στρώματα. Πιο συγκεκριμένα, το πρώτο από αυτά τα στρώματα περιέχει τα I/O, θεωρώντας ότι η λογικά μπλοκ (CLBs) έχουν ανατεθεί στο δεύτερο στρώμα. Τα μπλοκ μνήμης RAM είναι αποκλειστικά τοποθετημένα σε τρίτο στρώμα.

Το προτεινόμενο 3-D FPGA βασίζεται σε island-style, όπου τα μπλοκ λογικής διατάσσονται σε μία συστοιχία (τετράγωνο ή ορθογώνιο) και περιβάλλονται από οριζόντια και κάθετα κανάλια δρομολόγησης. Η επικοινωνία μεταξύ των πόρων που βρίσκονται σε διαφορετικά στρώματα μπορεί να γίνεται είτε μέσω TSVs, ή SSITs [2]. Παρόμοια με τις σχετικές εργασίες για το σχεδιασμό 3-D FPGAs [78] [3] [79] η διασύνδεση μεταξύ των στρωμάτων στη παρούσα εργασία πραγματοποιείται με TSVs. Αυτά είναι TSVs εφαρμόζονται στο εσωτερικό των SBs, τα οποία έχουν κατάλληλα επεκταθεί ώστε να λαμβάνουν υπόψη την τρίτη διάσταση [80]. Τέτοιου είδους συνδεσιμότητα παρέχει μονοπάτια δρομολόγησης (απεικονίζονται με διακεκομμένες γραμμές στο σχήμα 4.10.) μεταξύ των SBs και I/O pads που βρίσκονται χωροταξικά στις ίδιες (x, y) συντεταγμένες.



Σχήμα 4.11: Αρχιτεκτονικό πρότυπο της προτεινόμενης 3-D αρχιτεκτονικής με τρία στρώματα που διασυνδέονται μέσω TSVs (γραμμές με κόκκινες χρώμα).

Ο Πίνακας 4.2 συνοψίζει τις κύριες ιδιότητες της TSV τεχνολογίας που χρησιμοποιείται [7], ενώ σχετικά με την μοντελοποίηση των υπόλοιπων πόρων του υλικού (π.χ. δρομολόγηση καλωδίων, τρανζίστορ, κλπ) έχουν σχεδιαστεί σε τεχνολογία CMOS 90nm, ακολουθώντας μια προσέγγιση παρόμοια με αυτή που διατυπώθηκε στην σχετική βιβλιογραφία [5]. Η επιλογή των τιμών που χρησιμοποιούνται για τα αρχιτεκτονικά στοιχεία δεν επηρεάζουν το γενικότητα της προτεινόμενης λύσης.

Πίνακας 4.2: Χαρακτηριστικά των TSVs που χρησιμοποιήθηκαν [7].

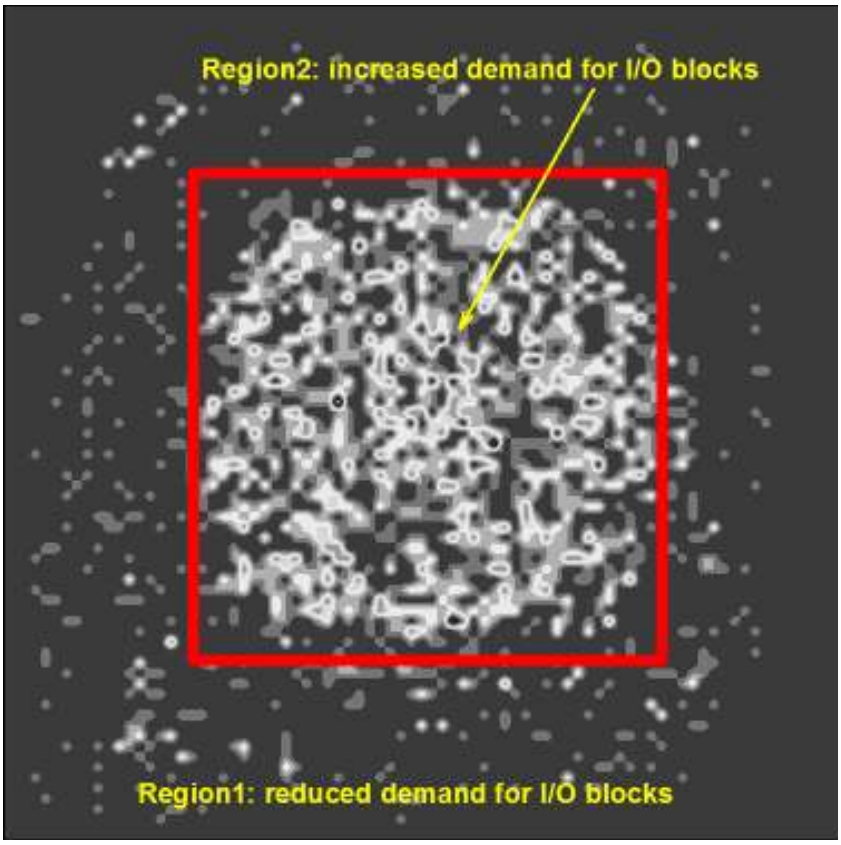
Property	Value
Length:	4-9 um
Diameter:	1.2 um
Minimum pitch (P_{TSV}):	4 um
Resistance:	0.35 Ω
Capacitance:	2.5 fF

Μια σημαντική διαφοροποίηση της εισαγόμενης προσέγγισης σε σύγκριση με τις υπάρχουσες αρχιτεκτονικές λύσεις έχει να κάνει με την ετερογένεια στην χωρική κατανομή

των I/O pads. Συγκεκριμένα, μέχρι τώρα, οι σχεδιαστές προτιμούν να σχεδιάσουν ομοιόμορφες πλατφόρμες, όπου τα I/Os έχουν εκχωρηθεί στην περιφέρεια του κάτω στρώματος.

Ακόμη και αν οι υπάρχοντες προσεγγίσεις οδηγούν σε μια πιο ομοιόμορφη αρχιτεκτονική, ωστόσο, όπως συζητήθηκε στην Ενότητα 2, είναι σπάνια αποτελεσματική λύση. Για αυτό το σκοπό, σε όλη αυτήν την ενότητα μια υλοποίηση της προτεινόμενης 3-D αρχιτεκτονικής μελετάται με στόχο να προσφέρει ακόμα πιο αποτελεσματική υλοποίηση εφαρμογών.

Πιο συγκεκριμένα, η καινοτομία της παρούσας υλοποίησης επηρεάζει την χωρική κατανομή των I/O με βάση τις πραγματικές ανάγκες για συνδεσιμότητα που τίθενται από την εφαρμογή. Για το σκοπό αυτό, αρχικά τα MCNC [81] benchmarks τοποθετούνται και δρομολογούνται σε ένα συμβατικό (δηλαδή παρόμοιο με εκείνο που απεικονίζεται στο Σχ. 4.11) 3-D FPGA και στη συνέχεια μελετάται η χωρική θέση των χρησιμοποιούμενων I/O pads. Τα αποτελέσματα αυτής της ανάλυσης συνοψίζονται στο Σχ. 4.12.



Σχήμα 4.12: Πυκνότητα των χρησιμοποιούμενων I/O pads στα MCNC benchmarks.

Κάθε σημείο (i, j) στο Σχ. 4.11 απεικονίζει το μέσο αριθμό χρησιμοποιούμενων TSVs στο εσωτερικό του αντίστοιχου SB που βρίσκεται χωρικά στην τοποθεσία (i, j). Με βάση το Σχήμα. 4.12, βγαίνει το συμπέρασμα ότι η ζήτηση για επικοινωνία μεταξύ στρωμάτων διαφοροποιείται μεταξύ δύο αυθαίρετων σημείων $(x_1, y1)$ και (x_2, y_2) της συσκευής,

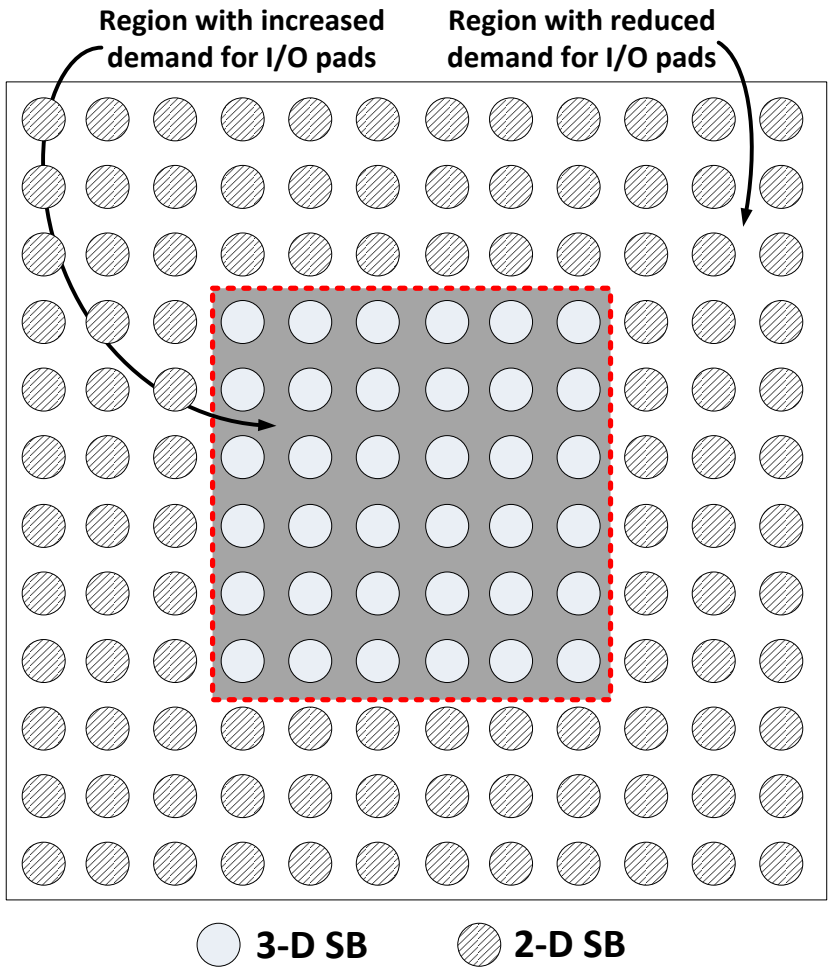
ακόμα και για τα 3-D SBs που τοποθετούνται σε γειτονικές περιοχές εντός του ίδιου στρώματος. Από την ανάλυσή μας, βρέθηκε ότι στη μέση του στρώματος το ποσοστό των χρησιμοποιούμενων TSVs είναι σημαντικά υψηλότερο, σε σύγκριση με την περιφέρεια. Αυτή η μη ομοιόμορφη χρησιμοποίηση των TSVs οφείλεται στην συμφόρηση δρομολόγησης που εμφανίζεται στο κέντρο ενός στρώματος FPGA και, ως εκ τούτου, πιο πολλοί πόροι δρομολόγησης πρέπει να κατασκευαστούν εντός της περιοχής αυτής [5] [82].

Τα αποτελέσματα από το Σχ. 4.12 υποδεικνύουν ότι οι περιοχές με περίπου σταθερή ζήτηση κάθετων συνδέσεων μπορούν να διακριθούν σε κάθε στρώση. Τα SBs ομαδοποιούνται ως εκ τούτου, σε δύο περιοχές με βάση το ποσοστό των χρησιμοποιούμενων κάθετων συνδέσεων της κάθε περιοχής. Πιο συγκεκριμένα, μέσα στην περιοχή $Region_1$ το ποσοστό των χρησιμοποιούμενων TSVs στα 3-D SBS είναι μικρότερο από 33%, ενώ η πυκνότητα των TSVs που χρησιμοποιούνται στην $Region_2$ είναι μεγαλύτερη από 33%. Η επιλογή δύο περιοχών αντιπροσωπεύει καλύτερα τη ζήτηση διαστρωματικής σύνδεσης .(π.χ. μικρή και υψηλή).

Για να αξιοποιηθεί καλύτερα αυτό το χαρακτηριστικό των 3-D FPGA αρχιτεκτονικών, μπορούμε να εφαρμόσουμε μια διαφορετική πυκνότητα διαστρωματικής συνδεσιμότητας σε κάθε (x, y) σημείο της 3-D αρχιτεκτονικής, ανάλογα με τις απαιτήσεις μιας εφαρμογής. Αυτή η προσέγγιση βελτιστοποίησης, όμως, οδηγεί σε έναν ASIC-like σχεδιασμό. Ως εκ τούτου, προτείνεται στην παρούσα εργασία μια τμηματικά ομογενής αρχιτεκτονική διασύνδεσης που αποτελείται από μερικές περιοχές με διαφορετικές πυκνότητες διαστρωματικής διασύνδεσης, παρόμοια με εκείνο που απεικονίζεται στο Σχ. 4.13.

Με την εξάλειψη σχεδόν της πλειοψηφίας των I/O pads από την περιφέρεια του κάτω στρώματος και, κατά συνέπεια, τα αντίστοιχες 3-D SBs, επιτυγχάνεται μια σημαντική βελτίωση όσον αφορά την απόδοση και το εμβαδόν του πυριτίου με σχεδόν αμελητέα υποβάθμιση της δυνατότητας δρομολόγησης της εκάστοτε εφαρμογής, όπως είχε ήδη συζητηθεί σε προηγούμενες δημοσιεύσεις [58].

Πιο συγκεκριμένα, οι δύο διαφορετικοί τύποι των SB χρησιμοποιούνται στο προτεινόμενο 3-D FPGA (βλέπε Σχ. 4.13). Ένα 2-D SB χρησιμοποιείται για τη σύνδεση μιας διαδρομής δρομολόγησης για καλώδια του ίδιου στρώματος. Εναλλακτικά, ένα 3-D SB υποστηρίζει επιπλέον συνδέσεις στην τρίτη διάσταση (κατώτερο στρώμα με το I / O pads). Για να διατυπωθούν αυτά τα δύο SBs, η ευελιξία του SB συμβολίζεται με F_S. Πιο συγκεκριμένα, αυτή η ευελιξία δίνει τον αριθμό κατευθύνσεων προς τις οποίες μπορεί να συνδεθεί κάθε εισερχόμενο καλώδιο. Ως εκ τούτου, ένα 2-D SB έχει F_S=3 (τρεις κατευθύνσεις μέσα στο ίδιο στρώμα), ενώ ένα 3-D SB παρέχει επίσης συνδεσιμότητα στον κατακόρυφο άξονα έχει F_S=5. Υποθέτοντας μια αρχιτεκτονική με ένα μήκος τμήματος του 1 tile, ένα 2-D SB σχηματίζεται από 6× τρανζίστορ, ενώ ένα 3-D SB απαιτεί 15× τρανζίστορ, όπου W_H και W_V δηλώνουν το πλάτος της ενδο και δια στρωματικής δρομολόγησης, αντίστοιχα [3] [58].



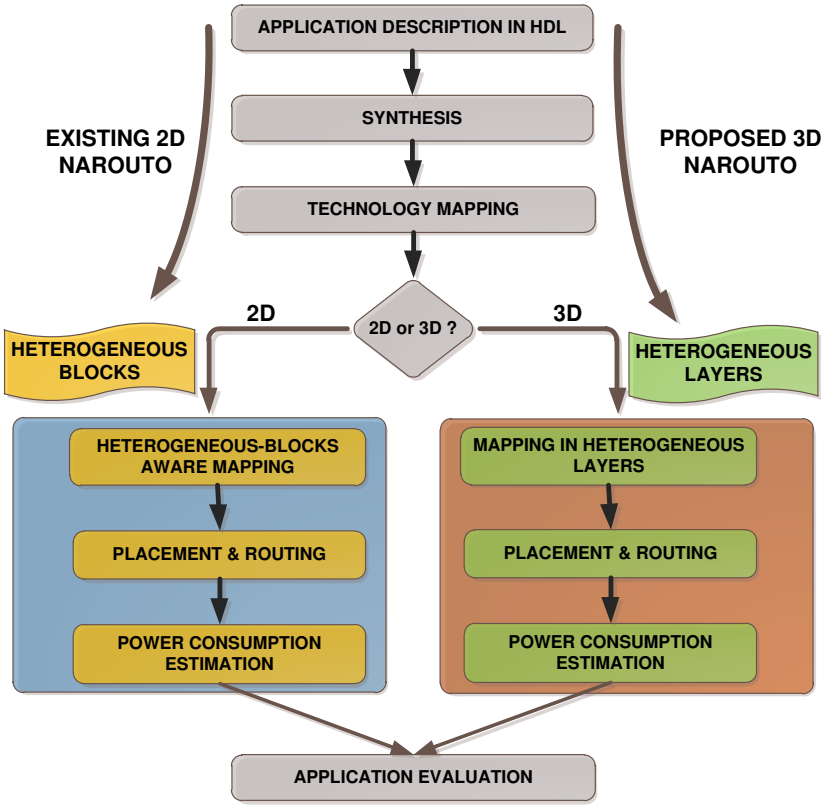
Σχήμα 4.13: Κατανομή των I/O pads.

Το συμπέρασμα σχετικά με τη χρήση λιγότερων I/O pads έχει ενσωματωθεί στην αρχιτεκτονική μας, έτσι ώστε να προσδιοριστεί η βέλτιστη χωρική θέση τους. Συγκεκριμένα, καθ 'όλη αυτήν την εργασία υποθέτουμε ότι το προτεινόμενο 3-D FPGA ενσωματώνει ακριβώς τον ίδιο αριθμό I/O pads σε σύγκριση με την αντίστοιχη 2-D αρχιτεκτονική [5]. Όσον αφορά τα συμβατικά 2-D FPGA με διαστάσεις $M \times N$ (M στήλες και σειρές από λογικά μπλοκ) και ένα I/O pad ανά λογικό μπλοκ, ο συνολικός αριθμός των pads είναι $2 \times (M + N)$. Από Από την άλλη πλευρά, εάν υποτεθεί ένα 3-D FPGA (παρόμοιο με εκείνο που απεικονίζεται στο Σχ 4.11.) τότε υπάρχουν $M \times N$ I/O pads για το ίδιο μέγεθος FPGA. Για λόγους πληρότητας, οι προτεινόμενες αρχιτεκτονικές λύσεις για 3-D FPGAs αξιολογούνται υποθέτοντας ακριβώς τον ίδιο αριθμό I/Os, σε σύγκριση με το αντίστοιχο 2-D FPGA. Προς αυτή την κατεύθυνση, ένας αριθμός I/O pads εξαλείφονται, που έχουν εκχωρηθεί σε περιοχή με μειωμένη ζήτηση για συνδεσιμότητα (π.χ. περιφέρεια της συσκευής).

4.4 Supporting Tool Framework

Το framework για την υποστήριξη της εξερεύνησης σε επίπεδο αρχιτεκτονικής, καθώς και την απεικόνιση της εφαρμογής σε 3-D FPGAs με ετερογενή στρώματα, παρουσιάζεται σε αυτήν την ενότητα. Το νέο Tool Framework ονομάζεται 3-D NAROUTO.

Το Σχ. 4.14 απεικονίζει μια αφηρημένη άποψη των framework 2-D και 3-D NAROUTO . Κατά τη διάρκεια της ανάπτυξης του 3-D NAROUTO, μόνο τα εργαλεία που χρειάζεται να λάβουν υπόψη την τρίτη διάσταση έχουν καταλλήλως επεκταθεί, ενώ τα υπόλοιπα εργαλεία (π.χ. σύνθεση και technology mapping) είναι πανομοιότυπα μεταξύ των ροών. Η κύρια διαφοροποίηση των NAROUTO frameworks, σε σύγκριση με παρόμοιες προσεγγίσεις βρίσκονται σε σχετικές αναφορές (π.χ. [78] [3] [79]) έχει να κάνει με την την υποστήριξη ετερογένειας, είτε μέσα σε κάθε στρώμα, καθώς και μεταξύ των διαφορετικών στρωμάτων.



Σχήμα 4.14: 2-D and 3-D NAROUTO framework.

Μετά τη σύνθεση και το technology mapping της εφαρμογής αυτή τοποθετείται και δρομολογείται (P&R). Οι αλγόριθμοι που χρησιμοποιήθηκαν κατά τη διάρκεια του P&R βήματος βασίζονται σε μια εκτεταμένη έκδοση του εργαλείου VPR [5], δεδομένου ότι πρέπει να είναι ενήμεροι σχετικά με την πρόσθετη ευελιξία που εισάγεται από την 3-D αρχιτεκτονική. Ένα σημαντικό βήμα σε αυτή τη διαδικασία επηρεάζει την αξιολόγηση

ενδιάμεσων λύσεων, καθώς και το P&R της εφαρμογής. Για αυτό το σκοπό, κατάλληλα μοντέλα χρησιμοποιήθηκαν τα οποία παρέχουν αποτελέσματα, μεταξύ άλλων, σχετικά με την καθυστέρηση και την κατανάλωση ενέργειας. Δεδομένου ότι η αποδοτικότητα των λύσεων ποσοτικοποιείται βάση αυτών των μοντέλων, η ακρίβειά τους είναι σημαντική. Για το σκοπό αυτό, το μοντέλο Elmore [83] και PowerModel [15] χρησιμοποιούνται για ποσοτικοποίηση της κρίσιμης καθυστέρησης και της κατανάλωσης ισχύος, αντίστοιχα. Πρόσθετες τεχνικές λεπτομέρειες σχετικά με τους χρησιμοποιούμενους αλγόριθμους αυτής της ενότητας μπορούν να βρεθούν σε [5] [84].

4.5 Πειραματικά Αποτελέσματα

Αυτή η ενότητα παρέχει μια σειρά από ποσοτικές συγκρίσεις που αποδεικνύουν την αποτελεσματικότητα της προτεινόμενης λύσης. Για το υπόλοιπο της εργασίας αυτής, η προτεινόμενη προσέγγιση του υλικού / λογισμικού αξιολογείται με την χρήση των MCNC benchmarks [81] και Altera QUIP [9] benchmark suite. Το mapping μιας εφαρμογής σε 2-D και 3-D FPGAs, γίνεται με τη χρήση του VPR [5] και το 3-D NAROUTO framework, αντίστοιχα, ενώ ο μέσος δείκτης χρησιμοποίησης της λογικής μπλοκ είναι περίπου 90%. Για λόγους πληρότητας, οι αρχιτεκτονικές (2-D και 3-D FPGAs) περιέχουν το ίδιο αριθμό λογικών (CLBs), I/O μπλοκ και μπλοκ μνήμης.

Προκειμένου να μελετηθούν όλες τις πιθανές αρχιτεκτονικές λύσεις, σε αυτή την ενότητα παρουσιάζονται τα αποτελέσματα για τρία εναλλακτικά σενάρια τα οποία συνοψίζονται ως ακολούθως:

- Μελέτη των επιπτώσεων του I/O bottleneck: Να αξιολογηθεί η αποτελεσματικότητα ενός 3-D FPGA, όπου η λογική και I/O μπλοκ εκχωρηθεί σε διαφορετικά στρώματα.
- Μελέτη των επιπτώσεων του stacking μνήμης σε λογική: Το 3-D FPGA αποτελείται από δύο στρώματα, όπου το πρώτο στρώμα περιέχει λογική και I/O μπλοκ, ενώ μνήμες έχουν ανατεθεί στο δεύτερο στρώμα.
- Μελέτη ενός συνδυασμού των προαναφερόμενων αρχιτεκτονικών: Αυτό το σενάριο περιλαμβάνει ένα 3-D FPGA, όπου η λογική, μνήμη και I/O μπλοκ έχουν ανατεθεί σε διαφορετικά στρώματα.

4.5.1 I/O bottleneck

Η πρώτη σειρά των πειραματικών αποτελεσμάτων ποσοτικοποιεί την αποτελεσματικότητα της ανάθεσης I/O μπλοκ σε ένα διαφορετικό στρώμα, σε σύγκριση με τους πόρους λογικής. Για το σκοπό αυτό, η 3-D αρχιτεκτονική συνθέτεται από δύο στρώματα,

ενώ η αξιολόγηση εκτελείται με την χρήση ενός καθιερωμένου benchmark set [81]. Κατά τη διάρκεια αυτής της μελέτης χρειάζονται benchmarks χωρίς μπλοκ μνήμης, διότι εισάγουν πρόσθετα (τεχνητή) προβλήματα στα δίκτυα δρομολόγησης.

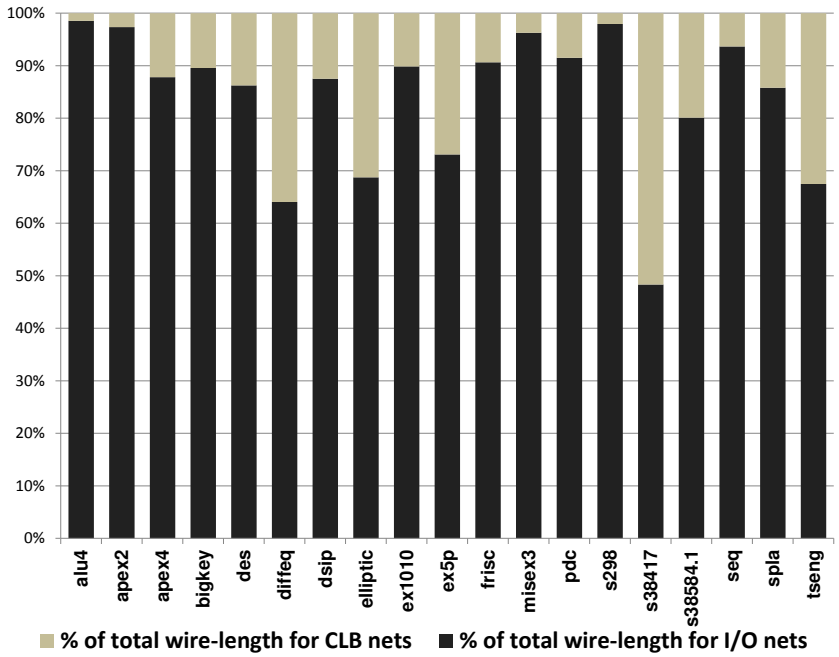
Αυτά τα benchmarks περιλαμβάνουν τόσο απαιτητικές εφαρμογές υπολογισμού καθώς και επικοινωνίας, όπως συνοψίζεται στον Πίνακα 4.3.

Πίνακας 4.3: Χαρακτηριστικά της σουΐτας benchmark.

Benchmark	**CLBs**	**Inputs**	**Outputs**	**Type**
alu4	1,592	14	8	Compute-bound
apex2	2,072	38	3	Compute-bound
apex4	1,299	9	19	Compute-bound
diffeq	1,875	229	197	Compute-bound
ex1010	4,682	256	245	Compute-bound
ex5p	1,162	64	39	Compute-bound
frisc	5,869	229	197	Compute-bound
misex3	1,477	131	114	Compute-bound
pdc	5,353	10	10	Compute-bound
s298	2,429	8	63	Compute-bound
s38417	8,108	20	116	Compute-bound
seq	1,911	14	14	Compute-bound
spla	3,865	16	40	Compute-bound
bigkey	2,039	4	6	I/O-bound
des	2,155	29	106	I/O-bound
dsip	1,619	38	304	I/O-bound
elliptic	4,183	41	35	I/O-bound
s38584	6,920	16	46	I/O-bound
tseng	1,401	52	122	I/O-bound
Average:	**3,158**	**64**	**88**	-

Προκειμένου να αποδειχθεί ότι η προτεινόμενη αρχιτεκτονική 3-D είναι ανώτερη σε σύγκριση με τα συμβατικά 2-D FPGAs, το Σχ 4.15 δίνει το μήκος σύρματος για δίκτυα που περιέχουν (ή όχι) τουλάχιστον ένα I/O. Πιο συγκεκριμένα, οι ανοιχτού χρώματος μπάρες απεικονίζουν το μήκος για δίκτυα που σχηματίζονται αποκλειστικά μεταξύ CLBs (ονομάζονται επίσης δίκτυα CLB), ενώ οι σκούρου χρώματος μπάρες αντιστοιχούν σε δίκτυα που έχουν τουλάχιστον ένα I/O pad (I/O δίκτυα). Δεδομένου ότι οι απόλυτες τιμές για το μήκος καλωδίων διαφέρουν πολύ μεταξύ των benchmark που μελετήθηκαν, οι τιμές απεικονίζονται με κανονικοποιημένο τρόπο. Συγκεκριμένα, μετά το επιτυχές P&R, το συνολικό μήκος καλωδίου ανά benchmark υπολογίζεται και στη συνέχεια, το μήκη των I/O και CLB δικτύων υπολογίζονται ως ποσοστό του συνολικού μήκους.

Με βάση τα αποτελέσματα που συνοψίζονται στο Σχ. 4.15, καταλήγουμε στο συμπέρασμα, ότι ο μέσος όρος μήκους για τα *I/O networks* είναι περίπου 92% του συνολικού μήκους για όλα τα δίκτυα, ενώ ο αντίστοιχος μέσος όρος μήκους για τα *CLB networks*



Σχήμα 4.15: Ποσοστό μήκους σύρματος για διαδρομές δρομολόγησης που περιέχουν (ή όχι) τουλάχιστον ένα μπλοκ I/O.

είναι μόλις 8%. Ότι μια τέτοια συμπεριφορά είναι κοινή για όλα τα benchmarks, ανεξάρτητα αν είναι compute-bound, ή I/O-bound. Το τελευταίο συμπέρασμα, όπως έχει ήδη συζητηθεί στην Ενότητα 2, υπογραμμίζει τη σημασία της ανάθεσης I/O pads αποκλειστικά σε ένα ειδικό στρώμα.

Ο πίνακας 4.4 ποσοτικοποιεί την αύξηση της απόδοσης σε σχέση με την συχνότητα λειτουργίας για τη χρήση της προτεινόμενης αρχιτεκτονικής 3-D, σε σύγκριση με ένα συμβατικό 2-D FPGA. Και οι δύο αρχιτεκτονικές 2-D και 3-D περιέχουν τον ίδιο αριθμό λογικών μπλοκ και I/O pads, ενώ ο αριθμός των καλωδίων δρομολόγησης ανά κανάλι είναι, επίσης, ίδιος.

Τα αποτελέσματα που συνοψίζονται στους πίνακες αυτούς δείχνουν ότι η χρήση της 3-D αρχιτεκτονικής οδηγεί σε μέση αύξηση της απόδοσης 1,36 × φορές. Αυτή η βελτίωση συμβαίνει κυρίως επειδή τα δίκτυα δρομολόγησης και πιο συγκεκριμένα τα κρίσιμα μονοπάτια, σε 3-D FPGAs είναι κατά πολύ μικρότερα σε σύγκριση με αυτά στις αντίστοιχες συσκευές 2-D. Επιπλέον, υπάρχουν κάποιες εφαρμογές που επιτυγχάνουν ακόμα μεγαλύτερη μείωση της καθυστέρησης (έως 2,72 × φορές) λόγω της καλύτερης απεικόνισης των εκτεταμένων *I/O* δικτύων που καλύπτουν ολόκληρη την αρχιτεκτονική.

Αξιολογήθηκε επίσης η κατανάλωση ισχύος για τις δύο εναλλακτικές αρχιτεκτονικές (2-D και 3-D). Τα αποτελέσματα αυτής της ανάλυσης συνοψίζονται στον πίνακα 4.5. Με βάση την παρούσα έρευνα, διαπιστώθηκε, ότι η προτεινόμενη 3-D FPGA αρχιτεκτονική παρέχει σχεδόν την ίδια κατανάλωση ενέργειας σε σύγκριση με τις αντίστοιχες συσκευές 2-D.

Πίνακας 4.4: Σύγκριση της μέγιστης συχνότητας λειτουργίας μεταξύ μιας 2-D και της προτεινόμενη 3-D FPGA αρχιτεκτονικής για υπολογιστικά απαιτητικές εφαρμογές.

Benchmark	**Maximum Operation Frequency (MHz)**		**Gain**
	3-D FPGA	**2-D FPGA**	(%)
alu4	16.35	13.32	1.23×
apex2	12.70	10.07	1.26×
apex4	15.88	13.54	1.17×
bigkey	28.05	10.30	2.72×
des	17.69	10.56	1.67×
diffeq	15.05	14.59	1.03×
dsip	28.50	16.12	1.77×
elliptic	8.16	7.91	1.03×
ex1010	8.80	6.53	1.35×
ex5p	16.88	14.50	1.16×
frisc	6.87	5.55	1.24×
misex3	17.76	13.58	1.31×
pdc	8.04	6.26	1.28×
s298	7.35	7.49	0.98×
s38417	8.37	8.34	0.99×
s38584.1	10.40	9.61	0.92×
seq	17.26	11.31	1.53×
spla	11.57	8.36	1.38×
tseng	16.46	16.71	0.98×
Average:	**14.92**	**11.01**	**1.36**×

4.5.2 Επιπτώσεις της ανάθεσης μνήμης και λογικής σε διαφορετικά στρώματα.

Μετά την ποσοτικοποίηση της αποτελεσματικότητας της βελτίωσης στο I/O bandwidth, μελετάται η επίδραση της εκχώρησης των μπλοκ μνήμης σε διαφορετικό στρώμα, από τους λογικούς πόρους. Για αυτό τον σκοπό, ένας αριθμός benchmarks από την QUIP suite [9] χρησιμοποιήθηκαν. Ο πίνακας 4.6 αξιολογεί τη μέγιστη συχνότητα λειτουργίας και την κατανάλωση ισχύος, όταν τα benchmarks αυτά απεικονίζονται σε 3-D FPGAs με δύο στρώματα (λογικής και μνήμης).

Τα πειραματικά αποτελέσματα από αυτήν την μελέτη έδειξαν ότι η προτεινόμενη 3-D λύση βελτιώνει την απόδοση των αρχιτεκτονικών, δεδομένου ότι οδηγεί σε υψηλότερο bandwidth μεταξύ μνήμης λογικής. Πιο συγκεκριμένα, υπάρχει μια μέση μείωση στην καθυστέρηση και στην ισχύ 29% και 10%, αντίστοιχα, σε σύγκριση με τα αντίστοιχα 2-D FPGAs. Αυτές οι βελτιώσεις συμβαίνουν κυρίως λόγω της μικρότερης διαδρομής των σημάτων που παρέχουν επικοινωνία μεταξύ της λογικής (CLBs) και των μπλοκ μνήμης. Τα κανάλια στον z άξονα είναι πιο πλατιά λόγω των απαιτήσεων των μνημών

Πίνακας 4.5: Σύγκριση της κατανάλωσης ισχύος μεταξύ της 2-D και της προτεινόμενης 3-D FPGA αρχιτεκτονικής.

Benchmark	**Power consumption (mWatt)**		**Gain**
	3-D FPGA	**2-D FPGA**	(%)
alu4	47.29	46.12	-2.47%
apex2	63.40	62.35	-1.66%
apex4	37.78	36.65	-2.99%
bigkey	22.79	22.89	0.44%
des	62.71	62.59	-0.19%
diffeq	35.89	35.89	0.00%
dsip	80.96	80.75	-0.26%
elliptic	46.02	45.18	-1.83%
ex1010	121.01	120.86	-0.12%
ex5p	44.38	43.89	-1.10%
frisc	130.61	129.74	-0.67%
misex3	55.84	51.67	-7.47%
pdc	84.18	83.46	-0.86%
s298	48.60	51.05	5.04%
s38417	76.06	80.19	5.43%
s38584	63.80	71.19	11.59%
seq	58.81	58.54	-0.45%
spla	235.58	232.47	-1.32%
tseng	25.75	25.93	0.71%
Average:	**70.61**	**70.60**	**0.10%**

Benchmark	**Max. Operation Frequency (MHz)**			**Power Consumption (mW)**		
	2-D FPGA	**3-D FPGA**	**Gain (%)**	**2-D FPGA**	**3-D FPGA**	**Gain (%)**
oc_aes_core_inv	101.58	131.96	29.91%	0.723	0.821	-13.48%
ata_ocidec	141.67	248.27	75.25	0.148	0.107	27.76%
os_blowfish	93.55	94.17	0.67%	0.250	0.221	11.75%
oc_hdlc	176.40	207.26	17.50%	0.182	0.167	8.22%
oc_minirisc	131.55	156.25	18.78%	0.066	0.057	13.63%
Average:	**128.95**	**167.58**	**28.42%**	**0.31**	**0.27**	**9.58%**

Πίνακας 4.6: Αξιολόγηση της βελτιστοποίησης στο bandwidth της μνήμης μεταξύ της 2-D και της προτεινόμενης 3-D FPGA αρχιτεκτονικής.

από ότι είναι σε άλλες 3-D FPGA υλοποιήσεις (π.χ. [78], [3], [58]) όπου τα διαφορετικά στρώματα έχουν μόνο μνήμες.

4.5.3 ΄Ενα 3D FPGA, με αποκλειστικά στρώματα για CLB, I/O και μνήμη

Στην υποενότητα αυτήν, αξιολογείται η υλοποίηση εφαρμογών σε 3-D FPGAs, όπου και τα I/O και οι μνήμες έχουν εκχωρηθεί σε διαφορετικά στρώματα, σε σχέση με τους λογικούς πόρους. Για αυτό το σκοπό, ο πίνακας 4.7 συνοψίζει τον αντίκτυπο της υλοποίησης των benchmarks σε μία αρχιτεκτονική με τρία στρώματα. Από τα αποτελέσματα που συνοψίζονται στον πίνακα 4.7 συμπεραίνεται ότι η προτεινόμενη αρχιτεκτονική επιτυγχάνει βελτίωση της απόδοσης κατά 30% με αμελητέα ποινή στην κατανάλωση ισχύος (αύξηση κατά 2,6%).

Benchmark	**Max. Operation Frequency (MHz)**			**Power Consumption (mW)**		
	2-D FPGA	**3-D FPGA**	**Gain (%)**	**2-D FPGA**	**3-D FPGA**	**Gain (%)**
oc_aes_core_inv	101.58	129.40	27.41%	0.723	0.783	-8.30%
ata_ocidec	141.67	243.49	71.87%	0.148	0.137	6.74%
os_blowfish	93.55	115.08	23.017%	0.250	0.274	-9.74%
oc_hdlc	176.40	198.53	12.55%	0.182	0.172	5.65%
oc_minirisc	131.55	152.58	16.01%	0.066	0.071	-7.37%
Average:	**128.95**	**167.82**	**30.16%**	**0.27**	**0.29**	**-2.61%**

Πίνακας 4.7: Αξιολόγηση της προτεινόμενης 3-D FPGA αρχιτεκτονικής.

4.6 Συμπεράσματα

Το έργο αυτό εισάγει μία νέα αρχιτεκτονική 3-D FPGA, όπου I/O και μπλοκ μνήμης ανατίθενται σε διαφορετικά στρώματα σε σύγκριση με τους πόρους λογική . Η νέα αρχιτεκτονική υποστηρίζεται από ένα λογισμικό πλαίσιο, που ονομάζεται 3-D NAROUTO. Τα πειραματικά αποτελέσματα αποδεικνύουν την αποτελεσματικότητα της προτεινόμενης λύσης υλικού/λογισμικού, καθώς επιτυγχάνεται η δρομολόγηση εφαρμογών με σημαντικά μικρότερα μήκη σύρματος, το οποίο με τη σειρά του οδηγεί σε βελτιώσεις στην καθυστέρηση και στην ισχύ. Πιο συγκεκριμένα, η μέση μείωση της καθυστέρησης και της ισχύος είναι 30% και 10%, αντίστοιχα, σε σύγκριση με συμβατικά 2-D FPGAs.

Κεφάλαιο 5

Just in Time απεικόνιση εφαρμογών σε FPGAs

5.1 Εισαγωγή

Ο χρόνος εκτέλεσης συνήθως είναι ένας πονοκέφαλος για τους σχεδιαστές που εκτελούν την απεικόνιση μιας εφαρμογής σε επαναδιαμορφούμενες αρχιτεκτονικές. Σε αυτή την εργασία προτείνεται μια μεθοδολογία, καθώς και τα CAD εργαλεία, με στόχο την γρήγορη υλοποίηση εφαρμογών σε επαναδιαμορφούμενες αρχιτεκτονικές με τη χρήση ενός Just-in-Time (JiT) πλαισίου μεταγλώττισης. Πειραματικά αποτελέσματα αποδεικνύουν την αποτελεσματικότητα του προτεινόμενου πλαισίου, καθώς ο χρόνος εκτέλεσης είναι σημαντικά μειωμένος σε σύγκριση με τις state-of-the-art προσεγγίσεις. Επιπλέον, οι παραγόμενες λύσεις επιτυγχάνουν υψηλότερες συχνότητες λειτουργίας, ενώ εμφανίζουν σημαντικά χαμηλότερο κατακερματισμό των πόρων υλικού.

Μέχρι τώρα, το έργο της απεικόνισης εφαρμογής σε FPGAs εφαρμόζεται κυρίως κατά τη διάρκεια του σχεδιασμού της εφαρμογής, δεδομένου ότι οι αλγόριθμοι CAD έχουν μεγάλους χρόνους εκτέλεσης. Μια άλλη πρόκληση για αυτούς τους αλγορίθμους έχει να κάνει με τον κατακερματισμό της συσκευής, καθώς εισάγει περιορισμούς στην απεικόνιση μελλοντικών εφαρμογών. Για την αντιμετώπιση αυτών των περιορισμών, προτείνεται μια καινοτόμα μεθοδολογία για την γρήγορη υλοποίηση εφαρμογών σε FPGAs. Ο στόχος αυτής της προσέγγισης είναι να μειωθεί σημαντικά ο χρόνος εκτέλεσης με την ελάχιστη πιθανή υποβάθμιση των επιδόσεων. Προς αυτή την κατεύθυνση και σε αντίθεση με άλλες σύγχρονες προσεγγίσεις που οι εφαρμογές προ-τοποθετούνται και προ-δρομολογούνται, προτείνεται η χρήση ενός Just-In-Time (JIT) πλαισίου απεικόνισης. Συγκεκριμένα, τα δεδομένα διαμόρφωσης του FPGA υπολογίζονται κατά το χρόνο εκτέλεσης έπειτα τοποθέτηση, δρομολόγηση και παραγωγή bitstream της εφαρμογής.

5.1.1 Τοποθέτηση εφαρμογής σε FPGA

Μετά την ολοκλήρωση του σταδίου partitioning, η netlist της εφαρμογής ανατίθεται σε μία συγκεκριμένη θέση στο FPGA μέσω της διαδικασίας της τοποθέτησης. Η τοποθέτηση είναι ένα ουσιαστικό βήμα στα εργαλεία EDA, αφού αφορά το πρόβλημα της κατανομής των πόρων (καθορίζονται οι ακριβείς θέσεις για τα διάφορα συστατικά του κυκλώματος εντός της περιοχής FPGA). Η τοποθέτηση εφαρμογών σε FPGA πλατφόρμες μπορεί να λάβει ώρες, ή ακόμη και ημέρες, ανάλογα με την πολυπλοκότητα αυτών των εφαρμογών, δεδομένου ότι η τοποθέτηση είναι από τις πιο χρονοβόρες διαδικασίες της απεικόνισης σε reconfigurable αρχιτεκτονικές. Το πρόβλημα αυτό γίνεται ακόμη χειρότερο σε 3-D πλατφόρμες, αφού οι εν λόγω πλατφόρμες περιλαμβάνουν περισσότερους πόρους σε σχέση με τα 2-D FPGAs.

Μια καλής ποιότητας τοποθέτηση είναι απαραίτητη για τη συνολική ποιότητα της υλοποίησης μιας εφαρμογής, δεδομένου ότι επηρεάζει μεταξύ άλλων διάφορες μετρικές σχεδιασμού, όπως την καθυστέρηση διασύνδεσης, τη συμφόρηση, το μήκος των καλωδίων, καθώς επίσης και την κατανάλωση ενέργειας. Ενώ αλγόριθμοι τοποθέτησης για τη βελτίωση αυτών των μετρικών έχουν ερευνηθεί διεξοδικά, πολύ λίγες από τις εργασίες έχουν δώσει σημασία για την ελαχιστοποίηση του χρόνου εκτέλεσης του εργαλείου.

Η πλειονότητα των υφιστάμενων τεχνικών για την τοποθέτηση του κυκλώματος συνοψίζονται ως εξής:

- Αλγόριθμοι που βασίζονται στο partitioning, όπου η netlist και η περιοχή του FPGA χωρίζονται σε μικρότερες υπο-netlist και υπο-περιοχές, αντίστοιχα, σύμφωνα με cut-based συναρτήσεις κόστους. Αυτή η διαδικασία επαναλαμβάνεται έως ότου κάθε υπο-netlist και υπο-περιοχή είναι αρκετά μικρές για να τοποθετηθούν αποτελεσματικά. Ένα παράδειγμα αυτής της προσέγγισης είναι η τοποθέτηση min-cut [85].

- Αναλυτικές τεχνικές που μοντελοποιούν το πρόβλημα της τοποθέτησης χρησιμοποιώντας μια συνάρτηση (κόστος), η οποία μπορεί να μεγιστοποιηθεί ή να ελαχιστοποιηθεί μέσω μαθηματικής ανάλυσης. Παραδείγματα αναλυτικών τεχνικών περιλαμβάνουν quadratic placement και force-directed placement [86].

- Στοχαστικοί αλγόριθμοι, οι οποίοι εκτελούν τυχαίες κινήσεις, προκειμένου να βελτιστοποιήσουν μια συνάρτηση κόστους. Ένα τυπικό παράδειγμα μιας τέτοιας προσέγγισης είναι η χρήση simulated annealing αλγορίθμου [5].

- Τεχνικές που βασίζονται σε εξελικτικούς αλγόριθμους. Οι προσεγγίσεις αυτές αποσκοπούν να γίνει μια πιο αποτελεσματική εξερεύνηση του χώρου λύσεων, ενώ ο παραλληλισμός τους μπορεί να αξιοποιηθεί από τις υποκείμενες αρχιτεκτονικές πολλαπλών πυρήνων για τη μείωση του χρόνου εκτέλεσης.

5.1.2 Δρομολόγηση εφαρμογής σε FPGA

Μετά την τοποθέτηση, ο αλγόριθμος δρομολόγησης καθορίζει τις ακριβείς διαδρομές για τα δίκτυα της εφαρμογής έτσι ώστε να υλοποιηθούν πάνω στο FPGA. Οι ακριβείς διαδρομές των δικτύων πρέπει να πληρούν τους περιορισμούς σχεδιασμού που παρέχονται από σχεδιαστή για να εξασφαλιστεί η λειτουργικότητα της εφαρμογής του. Ο πιο σημαντικός στόχος της δρομολόγησης είναι να ολοκληρωθούν όλες τις απαιτούμενες συνδέσεις (δηλαδή να επιτευχθεί 100% διασύνδεση) αλλιώς, το κύκλωμα δεν θα λειτουργήσει καλά και μπορεί ακόμη και να αποτύχει. Άλλοι στόχοι, όπως η μείωση του μήκους των καλωδίων δρομολόγησης και η εξασφάλιση ότι κάθε δίκτυο πληροί τους απαιτούμενους χρονικούς προϋπολογισμούς, έχουν γίνει απαραίτητοι για τους σύγχρονους δρομολογητές.

Η δρομολόγηση είναι τυπικά ένα πολύ πολύπλοκο πρόβλημα. Για να καταστεί διαχειρίσιμο, το πρόβλημα της δρομολόγησης συνήθως λύνεται με τη χρήση μιας προσέγγισης δύο σταδίων, της καθολικής δρομολόγησης που ακολουθείται από την λεπτομερή δρομολόγηση. Η καθολική δρομολόγηση διαχωρίζει αρχικά την περιοχή δρομολόγησης σε πλακίδια και αποφασίζει σε κάθε κομμάτι τις διαδρομές για όλα τα δίχτυα, ενώ προσπαθεί να βελτιστοποιήσει κάποια δεδομένη συνάρτηση κόστους (π.χ. συνολικό μήκος καλωδίου και χρονισμό κυκλώματος). Στη συνέχεια, καθοδηγούμενη από τα μονοπάτια που καθορίστηκαν στο καθολικό επίπεδο δρομολόγησης, η λεπτομερής δρομολόγηση εκχωρεί πραγματικά καλώδια και διόδους για τα δίκτυα της εφαρμογής.

Όπως είναι γνωστό, οι πόροι δρομολόγησης είναι ήδη προκατασκευασμένοι και πολύ περιορισμένοι στις πλατφόρμες FPGA. Για να μειωθεί ο χρόνος εκτέλεσης, η παγκόσμια δρομολόγηση προσπαθεί να μειώσει την καθυστέρηση διάδοσης για κάθε δίκτυο, ενώ ταυτόχρονα εξισορροπεί τη συμφόρηση καναλιών και ελαχιστοποιεί την χωρική πυκνότητα των καναλιών διασύνδεσης. Στην παγκόσμια δρομολόγηση οι ακριβείς θέσεις των καναλιών δεν έχουν επιλεγεί ακόμα. Αντίθετα, κατά τη διάρκεια της λεπτομερούς δρομολόγησης, κάθε δίκτυο έχει εκχωρηθεί σε ένα συγκεκριμένο τμήμα του καλωδίου μέσα σε ένα δεδομένο κανάλι. Αυτή η διαδικασία ολοκληρώνεται μόνο όταν κάθε δίκτυο έχει τοποθετηθεί σε ένα δεδομένο κανάλι του FPGA.

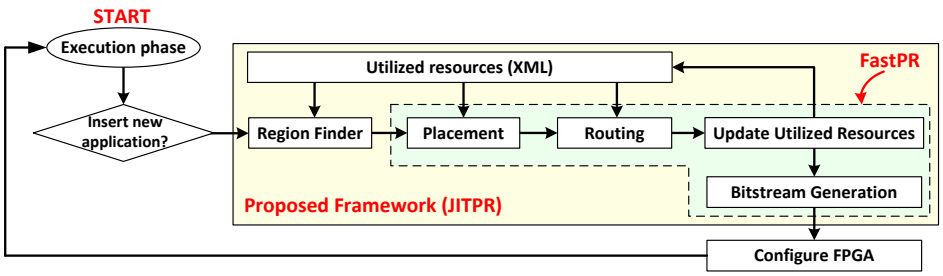
Ένα μικρότερο FPGA με ένα στενό πλάτος καναλιού είναι τυπικώς λιγότερο δαπανηρό και παρουσιάζει καλύτερη απόδοση από ένα μεγαλύτερο FPGA. Ως εκ τούτου, η λεπτομερής δρομολόγηση αποσκοπεί κυρίως στην ελαχιστοποίηση του συνολικού πλάτους καναλιού που απαιτείται για την διαδρομή όλων των δικτύων. Επιπλέον, λόγω της μεγάλης παρασιτικής χωρητικότητας και της αντίστασης των προγραμματιζόμενων διακοπτών, παίρνει σημαντικό χρόνο για να διαδοθεί ένα σήμα. Μία άλλη παράμετρος που επηρεάζει σημαντικά την αποτελεσματικότητα της απόδοσης των εφαρμογών είναι η έκταση των μονοπατιών της δρομολόγησης πάνω στην συσκευή. Συγκεκριμένα, σε περίπτωση που ένας δρομολογητής έχει να δρομολογήσει ένα δίκτυο που απλώνεται σε

ολόκληρο το FPGA, ο χρόνος εκτέλεσης της δρομολόγησης είναι μεγαλύτερος από αυτόν για ένα δίκτυο με μικρότερο μήκος.

Το υπόλοιπο του κεφαλαίου οργανώνεται ως εξής: Η ενότητα 2 περιγράφει το προτεινόμενο JiTPR πλαίσιο, ενώ οι αλγόριθμοι αναλύονται στην ενότητα 3. Τα πειραματικά αποτελέσματα που δείχνουν την αποτελεσματικότητα της προτεινόμενης λύσης βρίσκονται στην ενότητα 4. Τέλος, τα συμπεράσματα που συνοψίζονται στην ενότητα 5.

5.2 Προτεινόμενο λογισμικό πλαίσιο

Αυτή η ενότητα περιγράφει το προτεινόμενο πλαίσιο, που απεικονίζεται στο Σχήμα 5.1, για την υποστήριξη γρήγορης υλοποίησης εφαρμογών σε FPGA πλατφόρμες. Κάθε φορά που μια νέα εφαρμογή πρέπει να απεικονιστεί στην αρχιτεκτονική στόχου, η netlist της τροφοδοτείται στο προτεινόμενο JiTPR framework. Το πρώτο βήμα αφορά το floor plan της εφαρμογής με στόχο να βρεθεί η κατάλληλη περιοχή πάνω στο FPGA, όπου θα τεθεί η νέα εφαρμογή. Όπως θα αναλυθεί περαιτέρω αργότερα, αυτό είναι ένα κρίσιμο βήμα προκειμένου να διατηρηθεί στο ελάχιστο ο κατακερματισμός και για να μεγιστοποιηθεί η απόδοση. Στη συνέχεια, η netlist της εφαρμογής τοποθετείται και δρομολογείται (P&R) στην επιλεγμένη περιοχή. Εφόσον πρόσθετες εφαρμογές μπορεί να είναι ήδη απεικονισμένες στην αρχιτεκτονική, το λογισμικό πλαίσιο πρέπει να γνωρίζει τους διαθέσιμους (μη χρησιμοποιούμενους) πόρους της επιλεγμένης περιοχής. Για το σκοπό αυτό, το εργαλείο παίρνει ως είσοδο ένα αρχείο XML που περιγράφει τους κατεχόμενους πόρους (λογικής, I/O, ή δρομολόγησης) πάνω στη συσκευή FPGA, ενώ η λεπτομέρεια αυτής της περιγραφής είναι σε επίπεδο slice.



Σχήμα 5.1: Το προτεινόμενο πλαίσιο για την εκτέλεση γρήγορης απεικόνισης εφαρμογών σε συσκευές FPGA.

Σε αυτό το βήμα, όλες οι απαραίτητες πληροφορίες για τον υπολογισμό του αρχείου bitstream για την νέα εφαρμογή είναι διαθέσιμες. Ωστόσο, πριν από τη διαδικασία της διαμόρφωσης της συσκευής, το αρχείο XML συμπληρώνεται κατάλληλα προκειμένου να αντιπροσωπεύει την τρέχουσα κατάσταση των χρησιμοποιούμενων πόρων, λαμβάνοντας υπόψη και τους πόρους που απαιτούνται από τη νέα εφαρμογή. Τέλος, το αντίστοιχο αρχείο ρυθμίσεων δημιουργείται και το FPGA προγραμματίζεται με τη νέα

εφαρμογή. Πρόσθετες πληροφορίες σχετικά με την εφαρμογή αυτής της τεχνικής μπορούν να βρεθούν στο [34].

Από την άλλη πλευρά, όταν μια εφαρμογή πρέπει να αφαιρεθεί από την επαναδιαμορφούμενη συσκευή, το αρχείο XML ενημερώνεται καταλλήλως για την εξάλειψη αυτών των εγγραφών (επισημαίνονται οι πόροι αυτοί ως ελεύθεροι). Στη συνέχεια, τα αντίστοιχα slice διαμορφώνονται ως "άδεια". Διαφορετικές προσεγγίσεις θα μπορούσαν να χρησιμοποιηθούν για την πραγματοποίηση αυτού του βήματος (π.χ. προγραμματισμός των slice με ένα κενό αρχείο bitstream). Για την επιλογή στρατηγικής πρέπει να λάβει κανείς υπόψη και τους εγγενείς περιορισμούς που θέτει η επιλεγμένη συσκευή FPGA. Για παράδειγμα, αναφορικά με την πλατφόρμα, θα πρέπει να είναι σε θέση να παρέχει επαναδιαμόρφωση σε επίπεδο slice [34].

Το καθήκον της ταχείας υλοποίησης εφαρμογών με τη χρήση ενός πλαισίου JiT υποστηρίζεται από ένα εργαλείο που ονομάζεται NARUTO [87]. Ακόμα κι αν θα περίμενε κανείς ότι το P&R και η δημιουργία bitstream μιας εφαρμογής, θα εισήγαγε σημαντικές επιβαρύνσεις τόσο από την άποψη του χρόνου εκτέλεσης όσο και της ποιότητας των αποτελεσμάτων που παράγονται, ένα τέτοιο συμπέρασμα δεν προκύπτει με βάση τα πειραματικά αποτελέσματα που παρουσιάζονται στην ενότητα 4. Επιπλέον, το εισαγόμενο πλαίσιο JiT δημιουργεί τον ελάχιστο δυνατό κατακερματισμό, δεδομένου ότι δεν απαιτεί συνεχόμενη έκταση άδειων πόρων υλικού για την υλοποίηση της εφαρμογής.

5.3 Προτεινόμενη ροή εργαλείων

Αυτή η ενότητα περιγράφει με περισσότερες λεπτομέρειες τους αλγορίθμους για την υποστήριξη της ταχείας απεικόνισης εφαρμογής πάνω στην επιλεγμένη αρχιτεκτονική. Προηγούμενες αναλύσεις δείχνουν ότι αν οι αρχιτεκτονικές παράμετροι του FPGA είναι προκαθορισμένες, το στάδιο της τοποθέτησης είναι μακράν το πιο χρονοβόρο [37]. Αναλύονται εδώ τρεις ορθογώνιες προσεγγίσεις που ανατρέπουν αυτό το συμπέρασμα. Πιο συγκεκριμένα, αρχικά εισάγεται το εργαλείο RegionFinder, με στόχο να εντοπίσει την πιο κατάλληλη περιοχή του FPGA για την υλοποίηση της εφαρμογής. Στη συνέχεια, αναλύονται βελτιστοποιήσεις στον placer με στόχο να μειωθεί ο χρόνος εκτέλεσης, ενώ στο τελευταίο κομμάτι παρέχονται οι τροποποιήσεις του δρομολογητή για την εξισορρόπηση του χρόνου εκτέλεσης με την ποιότητα της τελικής λύσης.

5.3.1 JiT RegionFinder

Το πρώτο εργαλείο στο προτεινόμενο πλαίσιο JiTPR είναι το RegionFinder. Αυτό το εργαλείο εφαρμόζεται πριν το στάδιο της τοποθέτησης της εφαρμογής, προκειμένου να προσδιοριστεί η πλέον κατάλληλη χωρική τοποθεσία πάνω στη συσκευή, όταν η νέα εφαρμογή πρέπει να απεικονιστεί.

Σε αντίθεση με τις σχετικές προσεγγίσεις αντιμετώπισης του προβλήματος του floorplan, οι στόχοι της προτεινόμενης λύσης είναι δύο:

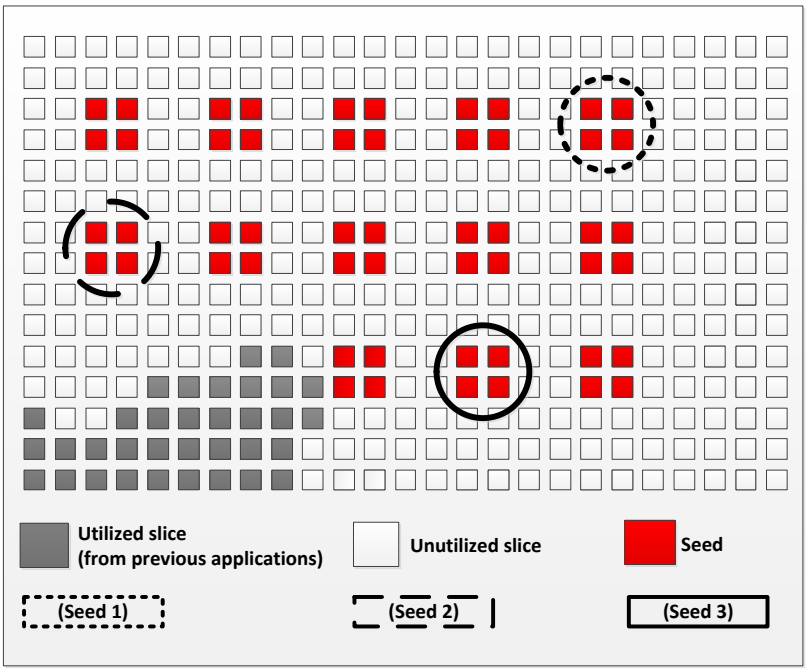
1. Να επιτύχει καλύτερη αξιοποίηση των διαθέσιμων πόρων και
2. να μειωθεί ο χρόνος εκτέλεσης του P&R της εφαρμογής.

Προς αυτή την κατεύθυνση, αντί να προσπαθεί να εντοπίσει τακτικές (συνήθως με ορθογώνιο ή τετράγωνο σχήμα) περιοχές συνεχόμενων ελεύθερων πόρων, παρόμοια με τις σχετικές προσεγγίσεις, το εργαλείο RegionFinder ενσωματώνει μία προχωρημένη ευριστική μεθοδολογία που στοχεύει να αξιολογήσει ταυτόχρονα πολλαπλές υποψήφιες περιοχές πάνω στην αρχιτεκτονική.

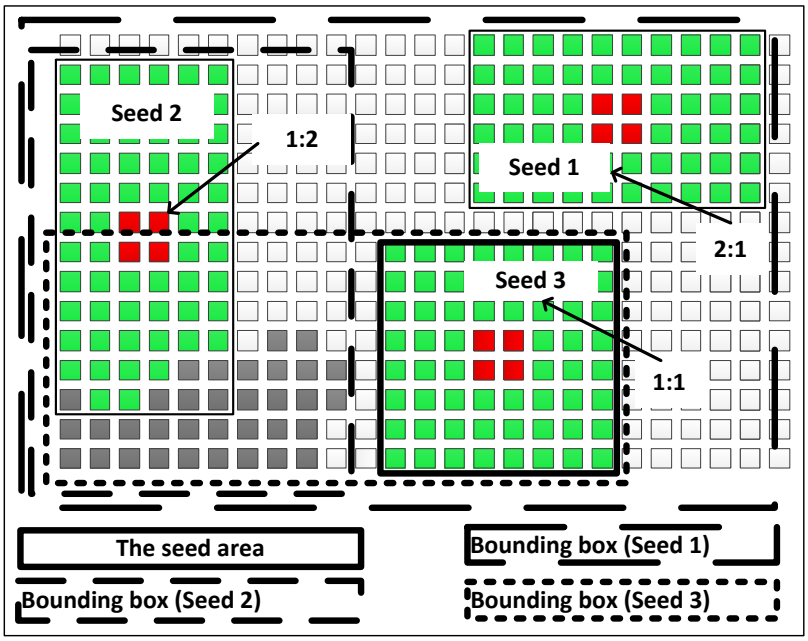
Οι είσοδοι στο εργαλείο RegionFinder είναι η netlist της εφαρμογής, καθώς και δύο XML αρχεία που περιγράφουν την αρχιτεκτονική του FPGA (π.χ. μέγεθος array, το πλάτος του καναλιού, κλπ) και τους διαθέσιμους πόρους (τόσο λογικής, όσο και δρομολόγησης) από τις εφαρμογές που έχουν ήδη απεικονιστεί πάνω στο FPGA, αντίστοιχα. Ομοίως, η έξοδος από αυτή την ανάλυση είναι μια περιοχή η οποία περιλαμβάνει μόνο την απαραίτητη ποσότητα πόρων του υλικού (π.χ. λογική και I/O μπλοκ), οδηγώντας σε χαμηλότερο ποσοστό κατακερματισμού, σε σύγκριση με τις σχετικές προσεγγίσεις. Ο αλγόριθμος, σε αντίθεση με τις σχετικές προσεγγίσεις, δεν εισάγει περιορισμούς για τις επερχόμενες εφαρμογές, δεδομένου ότι διατηρεί μόνο ότι πραγματικά χρειάζεται (για την υλοποίηση της εφαρμογής) από τους πόρους του υλικού, ενώ μπορεί επίσης να αξιοποιήσει περιοχές με ακανόνιστα σχήματα.

Η λειτουργικότητα του εργαλείου RegionFinder μπορεί να περιγραφεί ως εξής: Αρχικά, ο αλγόριθμος τοποθετεί έναν αριθμό ομοιόμορφα κατανεμημένων seeds σε όλο το FPGA, όπως παρουσιάζονται με κουτιά κόκκινου χρώματος στο σχήμα 5.2. Ο αριθμός των seeds προς τοποθέτηση, το μέγεθος τους, καθώς και η απόσταση μεταξύ δύο διαδοχικών seeds, καθορίζονται στο run-time, αφού η επιλογή τους επηρεάζεται από τη διαθεσιμότητα των πόρων υλικού πάνω στην αρχιτεκτονική και τις απαιτήσεις για επιδόσεις (περισσότερα seeds οδηγούν σε υψηλότερη απόδοση αλλά με αυξημένο υπολογιστικό κόστος).

Στη συνέχεια, κάθε ένα από τους σπόρους επεκτείνεται προς x και y με τις εξής τρεις αναλογίες (1:1, 1:2 και 2:1). Η επέκταση αυτή εφαρμόζεται επανειλημμένα έως ότου τα seeds να περιλαμβάνουν επαρκή ποσότητα πόρων υλικού για την υλοποίηση της εφαρμογής. Σημειώνεται ότι κατά τη διάρκεια αυτού του βήματος, το εργαλείο RegionFinder έχει επίγνωση των χρησιμοποιούμενων πόρων από τις προηγούμενες εφαρμογές. Η εικόνα 5.3 παρουσιάζει πώς το εργαλείο RegionFinder ανακτά τρεις υποψήφιες περιοχές (απεικονίζονται με διαφορετικά χρώματα) με την επέκταση των αντίστοιχων seeds με αναλογίες 1:1, 1:2 και 2:1, αντίστοιχα. Σε αυτό το σχήμα, τονίζεται επίσης το bounding box για τις υποψήφιες περιοχές.



Σχήμα 5.2: Παράδειγμα του RegionFinder, κατανομή seeds.



Σχήμα 5.3: Παράδειγμα του RegionFinder, επέκταση των seeds.

Η επιλογή της πλέον κατάλληλης περιοχής γίνεται με ποσοτικοποίηση της αποτελεσματικότητας εναλλακτικών seeds, με βάση μιας συνάρτησης κόστους που απεικονίζεται στην Εξίσωση 5.1.

$$Cost = a \times (UR_{Seed\_Area}) + (1 - a) \times (UR_{Bounding\_Box}) \tag{5.1}$$

, where:

$$UR_{Seed\_Area} = \left(\frac{Utilized\, CLBs}{Total\, CLBs\, at\, Seed\, Area} + \frac{Utilized\, IOs}{Total\, IOs\, at\, Seed\, Area} \right) \tag{5.2}$$

$$UR_{Bounding\_Box} = \left(\frac{Utilized\, CLBs\, at\, Bounding\, Box}{Total\, CLBs\, at\, Bounding\, Box} + \frac{Utilized\, IOs\, at\, Bounding\, Box}{Total\, IOs\, at\, Bounding\, Box} \right) \tag{5.3}$$

Ο συντελεστής στάθμισης a εξισορροπεί την προσπάθεια για βελτιστοποίηση είτε της τρέχουσας εφαρμογής είτε της επικείμενης, πάνω στην επαναδιαμορφούμενη αρχιτεκτονική. Ειδικότερα, κάθε φορά που το εργαλείο είναι ρυθμισμένο με μια υψηλή τιμή a, τότε, η υλοποίηση της εφαρμογής είναι όσο το δυνατόν πιο συμπαγής, οδηγώντας σε μεγιστοποίηση των επιδόσεων. Ωστόσο, υπάρχουν κάποιες κυρώσεις στο ποσοστό κατακερματισμού των πόρων. Από την άλλη πλευρά, η χαμηλότερη τιμή της παραμέτρου a αντιστοιχεί σε υλοποιήσεις με το ελάχιστο δυνατό bounding box, η οποίες με την σειρά τους βελτιώνουν τη δυνατότητα δρομολόγησης των μελλοντικών εφαρμογών που θα απεικονιστούν στην αρχιτεκτονική, αλλά επιβάλλει μία ελεγχόμενη ποινή, στην απόδοση της εφαρμογής. Το bounding box αντιστοιχεί στο ελάχιστο πλαίσιο που περιβάλλει όλους τους χρησιμοποιούμενους πόρους από ήδη υλοποιημένες εφαρμογές, καθώς και τους πόρους που ανήκουν στη περιοχή του seed. Κατά συνέπεια, με την κατάλληλη επιλογή της τιμής του a, είναι δυνατόν να δοθεί ένας αποδεκτός συμβιβασμός μεταξύ αυτών των ανταγωνιστικών σεναρίων υλοποίησης. Για τους σκοπούς της παρούσας εργασίας, η αξία του συντελεστή στάθμισης a ορίζεται σε 0,7.

Μια άλλη παράμετρος που πρέπει να μελετηθεί για το νέο εργαλείο επηρεάζει το χρόνο εκτέλεσης του. Η πολυπλοκότητα των RegionFinder είναι $O(n \times \sqrt{\frac{n}{m}})$, όπου n υποδηλώνει την τον αριθμό των slice που βρέθηκαν στην αρχιτεκτονική και m είναι ο αριθμός των slice που απαιτούνται για την υλοποίηση των εφαρμογών.

5.3.2 JiT Placer

Το επόμενο βήμα στο προτεινόμενο πλαίσιο περιλαμβάνει τον JiT placer, ο οποίος αναθέτει τα κομμάτια της εφαρμογής σε συγκεκριμένες θέσεις μέσα στην χωρική περιοχή που προήλθε από το RegionFinder εργαλείο, με τέτοιο τρόπο ώστε αυτά τα τμήματα να μπορούν να διασυνδεθούν επιτυχώς στο επόμενο βήμα (της δρομολόγησης) δεδομένων των διαθέσιμων πόρων δρομολόγησης. Για το σκοπό αυτό, ο placer ενσωματώνει

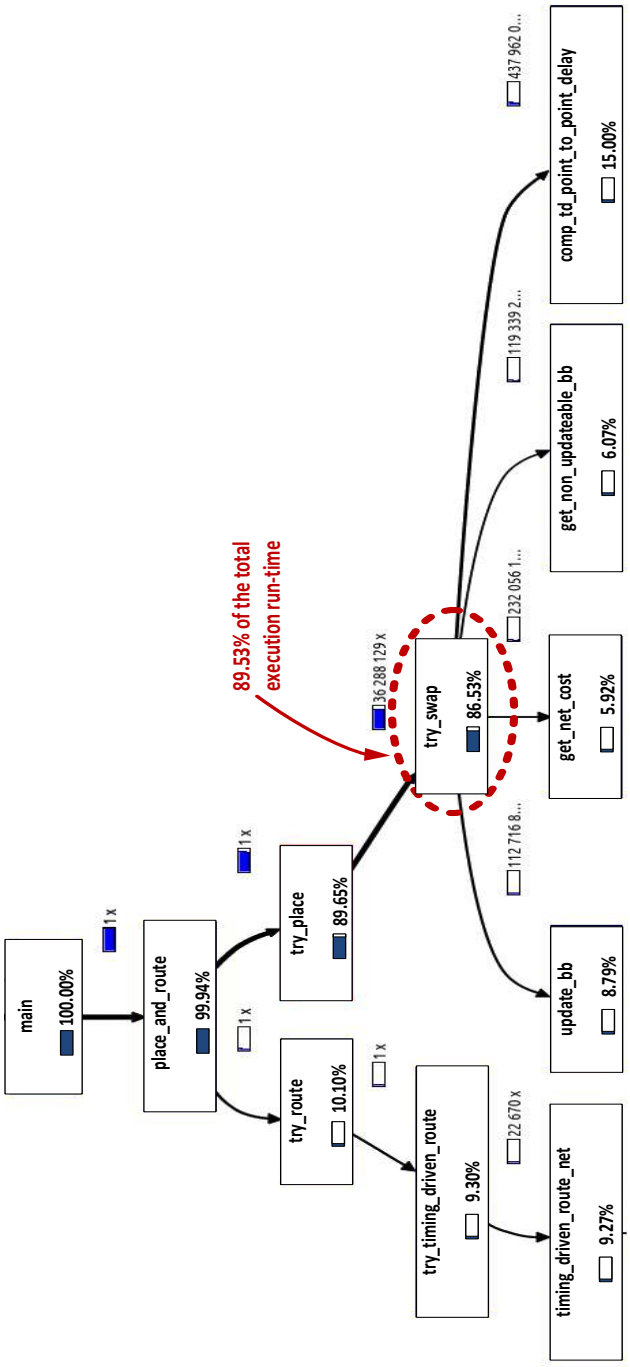
επίσης μια τεχνική για να εκτιμηθεί γρήγορα η δυνατότητα δρομολόγησης μέσα σε μία περιοχή [5].

Η λειτουργικότητα του placer βασίζεται σε μία γρήγορη simulated annealing προσέγγιση, παρόμοια με εκείνη που χρησιμοποιείται στο εργαλείο VPR [5]. Μια αρχική τοποθέτηση των μπλοκ σταδιακά βελτιστοποιείται από την εναλλαγή ζευγών μπλοκ, που έχει ως στόχο να βρεθεί μια ενδιάμεση τοποθέτηση με χαμηλότερο συνολικό κόστος. Άπληστη αποδοχή βελτίωσης κόστους συχνά οδηγεί σε ενδιάμεσες τοποθετήσεις που, ενώ τοπικά βέλτιστες, εξαρτώνται από την σειρά με την οποία ανταλλάσσονται τα μπλοκ και μπορεί να είναι μακριά από την ολικά βέλτιστη λύση. Οι αλγόριθμοι annealing χαρακτηρίζονται από το ότι αποδέχονται, όχι μόνο μεταθέσεις χαμηλότερου κόστους, αλλά και ένα ποσοστό μεταθέσεων με υψηλότερο κόστος για να αποφευχθεί η πρόωρη σύγκλιση σε τοπικά ελάχιστου κόστους λύσεις.

Η εικόνα 5.4 δίνει τον χρόνο εκτέλεσης για την τοποθέτηση του *alu4* benchmark με το εργαλείο VPR σε μια δεδομένη συσκευή FPGA. Όπως συνάγεται από αυτή την εικόνα, η "try_swap" λειτουργία είναι μακράν ο πιο χρονοβόρος πυρήνας, καθώς παίρνει σχεδόν το 89,5% του συνολικού χρόνου εκτέλεσης. Ακόμη και αν η εγγενής υπολογιστική πολυπλοκότητα της λειτουργίας αυτής είναι αρκετά χαμηλή, αυτή καλείται σχεδόν 36×10^6 φορές.

Όλοι οι δημόσια διαθέσιμοι placers που στηρίζονται στο simulated annealing εκτελούν $b \times N_{blocks}^R$ swaps ανά τιμή θερμοκρασίας, όπου N_{blocks} είναι ο αριθμός των μπλοκ (λογική και I/O) που χρησιμοποιούνται από την εφαρμογή, ενώ b=10 και $R = 4/3$. Στο [5], ο Betz υποδεικνύει ότι ο σταθερός συντελεστής του 10, μπορεί να κλιμακωθεί για να εξισορροπήσει τον χρόνο εκτέλεσης του placer με την ποιότητα, ενώ συμπληρωματικές προσεγγίσεις προς την ίδια κατεύθυνση περιλαμβάνουν την μείωση της θερμοκρασίας έναρξης και της αλλαγής του χρονοδιαγράμματος annealing [88].

Προς αυτή την κατεύθυνση, θα μπορούσαν να αρθούν οι επιβαρύνσεις του χρόνου εκτέλεσης κατά την τοποθέτηση, μειώνοντας την ποσότητα των φορών που η λειτουργία "try_swap" καλείται. Ωστόσο, σε μια τέτοια περίπτωση θα πρέπει να μελετηθούν οι επιπτώσεις αυτής της επιλογής σε σχέση με την υποβάθμιση των επιδόσεων. Πιο συγκεκριμένα, ο προτεινόμενος placer χρησιμοποιεί $b = 1.0$ και $R = 1.0$ (εκτελεί N_{blocks} swaps ανά τιμή θερμοκρασίας). Όσον αφορά το $alu4$ benchmark απεικονίζεται στο Σχήμα 5.4, μια τέτοια επιλογή οδηγεί σε μείωση του αριθμού της "try_swap" συνάρτησης 30×. Επιπλέον, με βάση τα αποτελέσματα που συνοψίζονται στην επόμενη ενότητα, όπως αναφέρθηκε προηγουμένως η μείωση των swaps ανά τιμή της θερμοκρασίας οδηγεί σε σημαντικά χαμηλότερη εκτέλεση χρόνου εκτέλεσης χωρίς να θυσιάζει την ποιότητα των παραγόμενων τοποθετήσεων.



Σχήμα 5.4: Αποτελέσματα σκιαγράφησης του $alu4$ benchmark με το VPR εργαλείο.

5.3.3 JiT Δρομολογητής Δημιουργία Bitstream

Εκτός από τον placer, το JiT πλαίσιο ενσωματώνει ένα δρομολογητή για τη δημιουργία μονοπατιών από τα net sources στα net destinations. Ο δρομολογητής που χρησιμοποιήθηκε είναι βασισμένος στον PathFinder negotiated congestion αλγόριθμο [83], που χρησιμοποιείται στο εργαλείο VPR, ωστόσο τροποποιήθηκε για να πετύχει ταχύτερους

χρόνους εκτέλεσης. Πιο συγκεκριμένα δύο παραμέτροι του αλγορίθμου του δρομολογητή τροποποιήθηκαν προς αυτή την κατεύθυνση:

- Το κόστος για την υπερ χρήση καναλιών έχει αλλάξει. Συγκεκριμένα, η προεπιλεγμένη τιμή στο αρχικό VPR (ίσο με 0,5) έχει αντικατασταθεί με μια υπερβολικά υψηλή τιμή (10,000). Η επιλογή αυτή αναγκάζει τον αλγόριθμο να δώσει αποδεκτά μονοπάτια δρομολόγησης από το πρώτο πέρασμα και, κατά συνέπεια, στις περισσότερες των περιπτώσεων, ο δρομολογητής δεν χρειάζεται να διαγράψει και να ξαναγράψει αυτά τα μονοπάτια
- Ο αλγόριθμος που ψάχνει για νόμιμα (αποδεκτά) μονοπάτια έχει περιορίσει το εύρος αναζήτησης στο bounding box (περιοχή που περιέχει $B_x \times B_y$ slices), σε αντίθεση με το VPR, όπου ένα net μπορεί να δρομολογηθεί σε μια περιοχή με $(B_x + 3) \times (B_y + 3)$ slices. Αυτή η επιλογή μειώνει τον χώρο εξερεύνησης της δρομολόγησης κατ $\frac{B_x \times B_y}{(B_x+3)\times(B_y+3)}$, όπου B_x και B_y είναι οι οριζόντιες και κάθετες διαστάσεις, αντίστοιχα του bounding box για ένα δεδομένο net.

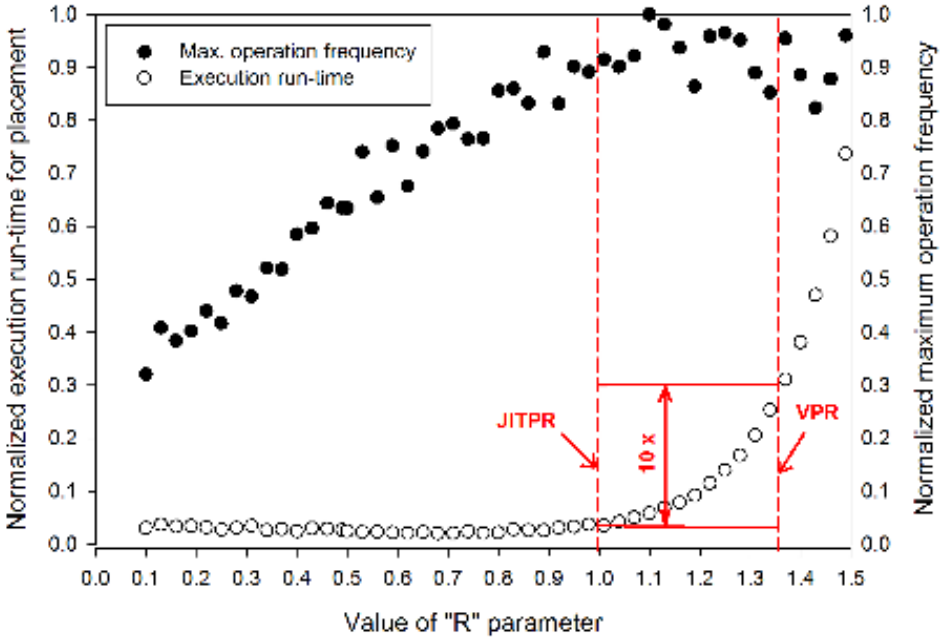
Με την ενσωμάτωση αυτών των διαφοροποιήσεων, οι αλγόριθμοι βελτιώνονται ως προς τον χρόνο εκτέλεσης, αλλά παρουσιάζουν μία αναμενόμενη ποινή στην απόδοση, λόγω της χαμηλότερης προσπάθειας για τη βελτιστοποίηση της διαδρομής δρομολόγησης (π.χ. χρησιμοποιούνται περισσότεροι πόροι διασύνδεσης). Μετά από το P&R της εφαρμογής, το σύνολο των απαραίτητων αρχείων εισόδου για τον υπολογισμό των δεδομένων διαμόρφωσης είναι διαθέσιμα. Ωστόσο, επειδή η δημιουργία του bitstream περιλαμβάνει επίσης μια σειρά από παραμέτρους της συσκευής που θα προγραμματιστεί, ο προγραμματισμός FPGA είναι πέρα από τα πεδία της εργασίας αυτής.

5.4 Πειραματικά αποτελέσματα

Αυτή η ενότητα παρέχει μια σειρά από συγκρίσεις που αποδεικνύουν την αποτελεσματικότητα του προτεινόμενου πλαισίου JiT, σε σύγκριση με σχετικές προσεγγίσεις. Εφόσον από τις πιο χρονοβόρες εργασίες είναι η απεικόνιση της εφαρμογής [37], εδώ δίνεται έμφαση στην ποσοτικοποίηση των δύο συμπληρωματικών εργαλείων, RegionFinder και FastPR που ασχολούνται με την τοποθέτηση εφαρμογών, δρομολόγηση και δημιουργία bitstream), ενάντια στο state-of-the-art εργαλείο VPR [5]. Δυστυχώς, δεν είναι δυνατό να δοθούν συγκρίσεις έναντι σε σχετικά εργαλεία που συζητούνται σε συναφείς εργασίες, δεδομένου ότι δεν είναι δημόσια διαθέσιμα. Για την ανάλυση αυτή, χρησιμοποιούνται τα 20 MCNC benchmarks [81] , τα οποία είναι ευρέως αποδεκτά στην FPGA κοινότητα. Όσον αφορά την αρχιτεκτονική, είναι ένα FPGA island-type που αποτελείται από 150 $\times$ 150 φέτες και 50 καλώδια ανά κανάλι δρομολόγησης.

5.4.1 JiTPR χρόνος εκτέλεσης και ποιότητα της απεικόνισης εφαρμογής

Η εικόνα 5.5 δείχνει την διακύμανση του χρόνου εκτέλεσης για την τοποθέτηση μιας εφαρμογής ως συνάρτηση της παραμέτρου "R" (συζητείται στο τμήμα 5.3.1). Ακόμα κι αν τα αποτελέσματα που συνοψίζονται στο σχήμα αυτό αφορούν το $alu4$ benchmark, είναι όμοια για το σύνολο των benchmarks. Με βάση αυτό το σχήμα, καταλήγουμε στο συμπέρασμα ότι η κλίση του χρόνου εκτέλεσης είναι σχεδόν μηδενική για τιμές R έως $1,0$, ενώ για μεγαλύτερες τιμές του R, υπάρχει μια εκθετική αύξηση. Συγκεκριμένα, ο προτεινόμενος placer (R = 1.0) επιτυγχάνει περίπου 10× πιο γρήγορη εκτέλεση σε σύγκριση με το εργαλείο VPR (όπου R = 1,33).



Σχήμα 5.5: Επίδραση του R στον χρόνο εκτέλεσης του placer και της Μεγιστης συχνότητας για το $alu4$ benchmark.

Ακόμα κι αν θα μπορούσε κανείς να υποθέσει ότι μια τέτοια επιτάχυνση έρχεται με σοβαρές επιβαρύνσεις στην απόδοση της εφαρμογής, αυτό δεν απεικονίζεται στο Σχήμα 5.5, όπου η μέγιστη συχνότητα λειτουργίας είναι σχεδόν σταθερή μεταξύ των δύο εναλλακτικών εργαλείων. Πιο συγκεκριμένα, με βάση την διερεύνηση, διαπιστώθηκε, ότι για τιμές R υψηλότερες από 0,9, υπάρχει μια επίδραση κορεσμού στην μέγιστη συχνότητα λειτουργίας, δεδομένου ότι η διάταξη τοποθετήσεως δεν οδηγεί σε υψηλότερες ποιοτικά λύσεις.

Προκειμένου να ποσοτικοποιήσουμε την επίδραση του προτεινόμενου πλαισίου στην απόδοση της εφαρμογής, ο πίνακας 5.1 παρέχει τον χρόνο εκτέλεσης για την τοποθέτηση, δρομολόγηση και την παραγωγή των δεδομένων διαμόρφωσης με τη χρήση του προτεινόμενου πλαισίου (FastPR), σε σύγκριση με το VPR [5]. Για λόγους πληρότητας, παρέχεται και ο χρόνος εκτέλεσης για το εργαλείο RegionFinder.

Πίνακας 5.1: Χρόνος εκτέλεσης (msec) για την διαδικασία της απεικόνισης εφαρμογής.

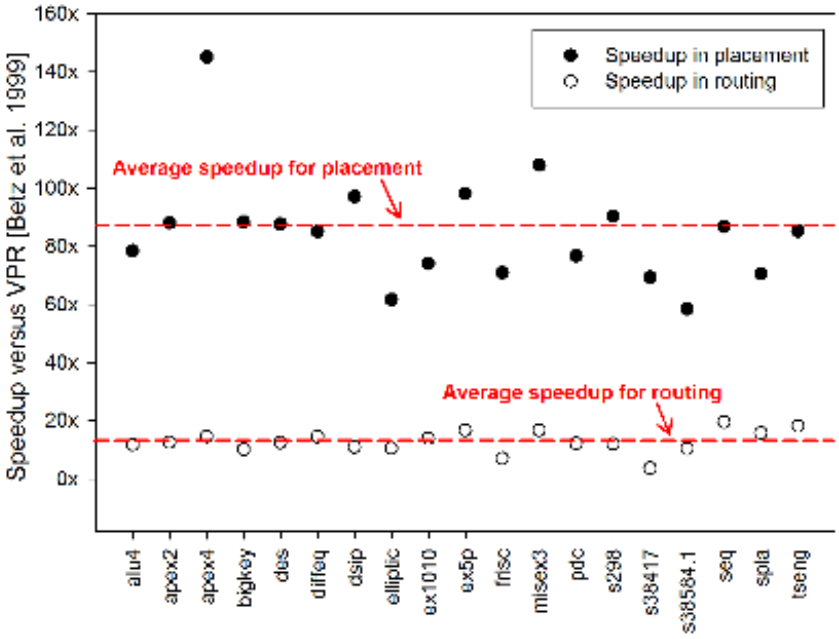
Benchmark	**Total Execution Time**				
	VPR	**JiTPR**			**Speedup**
		Region Finder	**FastPR**	**Total**	
alu4	54,647	35	1,006	1,041	52.49×
apex2	70,287	7	1,183	1,190	59.06×
apex4	49,987	7	612	619	80.75×
bigkey	78,898	8	1,837	1,845	42.76×
des	80,134	9	1,423	1,432	55.96×
diffeq	62,303	8	1,071	1,079	57.74×
dsip	61,171	9	1,188	1,197	51.10×
elliptic	148,379	12	3,816	3,828	38.76×
ex1010	174,274	13	3,504	3,517	49.55×
ex5p	47,559	7	688	695	68.43×
frisc	235,350	14	5,881	5,895	39.92×
misex3	52,441	8	748	756	69.37×
pdc	226,810	14	5,108	5,122	44.28×
s298	77,759	8	1,507	1,515	51.33×
s38417	337,944	16	7,362	7,378	45.80×
s38584.1	273,533	15	7,356	7,371	37.11×
seq	66,485	8	1,089	1,097	60.61×
spla	141,456	10	2,851	2,861	49.44×
tseng	50,928	7	817	824	61.81×
Average:	**120,544**	**11.32**	**2,581**	**2,592**	**53.49×**

Με βάση τα αποτελέσματα που συνοψίζονται στον πίνακα αυτόν, βγαίνει το συμπέρασμα ότι το προτεινόμενο πλαίσιο βελτιώνει τον χρόνο εκτέλεσης κατά 53.5× κατά μέσο όρο, σε σύγκριση με το εργαλείο VPR. Επιπλέον, στον πίνακα 5.2 ποσοτικοποιείται ο χρόνος εκτέλεσης ανά slice εφαρμογής. Προκειμένου να υπολογιστούν αυτές οι τιμές, ο συνολικός χρόνος εκτέλεσης ανά benchmark διαιρείται με τον αριθμό των χρησιμοποιούμενων slices. Από τα αποτελέσματα προκύπτει ότι το προτεινόμενο πλαίσιο JiTPR απαιτεί κατά μέσο όρο 0.8msec ανά slice, σε σύγκριση με τα 40.72msec που απαιτούνται από το VPR. Η βελτίωση στον χρόνο εκτέλεσης που απεικονίζεται στον πίνακα 5.1 προέρχεται κυρίως από την καλύτερη χειραγώγηση των διαθέσιμων πόρων με τη χρήση του RegionFinder, καθώς και τροποποιήσεις στον αλγόριθμο δρομολόγησης.

Η εικόνα 5.6 ποσοτικοποιεί την επιτάχυνση στον χρόνο εκτέλεσης για τον προτεινόμενο placer και router. Πιο συγκεκριμένα, ο κάθετος άξονας σε αυτό το σχήμα αναπαριστά (με κανονικοποιημένη τρόπο) τα οφέλη που επιτυγχάνονται με το JiTPR πλαίσιο, σε σύγκριση με τους αντίστοιχους αλγόριθμους στο εργαλείο VPR [5]. Με βάση τα αποτελέσματα, η λύση οδηγεί σε τοποθέτηση εφαρμογών κατά μέσο όρο κατά 85× γρηγορότερα, σε σύγκριση με το VPR, ενώ η αντίστοιχη τιμή για τη δρομολόγηση εφαρμογής είναι 14 ×. Αυτά τα αποτελέσματα δείχνουν την αποτελεσματικότητα του πλαισίου JiT να πραγματοποιήσει γρήγορα απεικόνιση εφαρμογής σε επαναδιαμορφούμενες αρχιτεκτονικές.

Πίνακας 5.2: Χρόνος εκτέλεσης (msec) ανά slice για την διαδικασία της απεικόνισης εφαρμογής.

Benchmark	**Execution Time per Slice**	
	VPR	**JiTPR**
alu4	35.39	0.67
apex2	36.61	0.62
apex4	38.78	0.48
bigkey	36.41	0.85
des	38.30	0.68
diffeq	38.94	0.67
dsip	34.06	0.67
elliptic	38.55	0.99
ex1010	37.74	0.76
ex5p	41.90	0.61
frisc	63.75	1.60
misex3	36.80	0.53
pdc	48.98	1.11
s298	39.71	0.77
s38417	51.67	1.13
s38584.1	40.28	1.09
seq	36.41	0.60
spla	37.70	0.76
tseng	41.71	0.67
Average:	**40.72**	**0.80**



Σχήμα 5.6: Επιτάχυνση χρόνου εκτέλεσης του προτεινόμενου placer και router συγκρινόμενοι με το VPR εργαλείο [5].

Στη συνέχεια, η απόδοση της λύσης αξιολογείται ως προς την Μέγιστη συχνότητα λειτουργίας και την κατανάλωση ενέργειας. Τα αποτελέσματα αυτής της ανάλυσης για τις δύο εναλλακτικές ροές εργαλείων συνοψίζονται στον πίνακα 5.3 και στον πίνακα 5.4. Με βάση τον εν λόγω πίνακα, το προτεινόμενο πλαίσιο οδηγεί κατά μέσο όρο σε 1,17× υψηλότερη συχνότητα λειτουργίας σε σύγκριση με το VPR, ενώ για ορισμένα benchmarks (π.χ. $DSIP$), αυτή η αύξηση κυμαίνεται έως και 1.91×. Εκτός από αυτό, από την ανάλυση βρέθηκε ότι το προτεινόμενο πλαίσιο επιτυγχάνει υψηλότερη αύξηση της συχνότητας για εφαρμογές I/O-bound, σε σύγκριση με compute-bound εφαρμογές [37]. Αυτό συμβαίνει κυρίως λόγω του RegionFinder, το οποίο καταβάλλει προσπάθεια να δημιουργήσει πιο συμπαγείς απεικονίσεις.

Η υψηλότερη συχνότητα λειτουργίας που επιτυγχάνεται με τη χρήση του προτεινόμενου πλαισίου επιβάλει υψηλότερη κατανάλωση ισχύος, όπως συνοψίζεται στον πίνακα 5.4. Συγκεκριμένα, η μέση κατανάλωση ισχύος για το JiTPR είναι περίπου 1,38× υψηλότερη σε σύγκριση με το εργαλείο VPR. Ωστόσο, αν λειτουργεί η αρχιτεκτονική με την συχνότητα λειτουργίας που ανακτάται από το εργαλείο VPR, τότε η ποινή στην κατανάλωση ισχύος του προτεινόμενου πλαισίου είναι μόνο 1,21×. Μια τέτοια ποινή είναι ανεκτή, αν λάβουμε υπόψη το σημαντικά χαμηλότερο χρόνο εκτέλεσης.

Πίνακας 5.3: Αξιολόγηση απόδοσης της εφαρμογής όταν απεικονίζεται με διαφορετικά πλαίσια.

Benchmark	**Max. Operation Freq. (MHz)**		
	VPR	**JiTPR (proposed)**	**Gain**
alu4	12.10	10.10	0.83×
apex2	6.75	7.91	1.17×
apex4	11.60	10.30	0.89×
bigkey	11.30	20.20	1.79×
des	6.54	11.40	1.74×
diffeq	11.40	13.40	1.18×
dsip	11.60	22.10	1.91×
elliptic	9.27	7.92	0.85×
ex1010	5.73	5.63	0.98×
ex5p	12.60	11.00	0.87×
frisc	6.11	7.05	1.15×
misex3	11.40	11.00	0.96×
pdc	5.05	5.29	1.05×
s298	6.85	6.87	1.00×
s38417	9.52	6.39	0.67×
s38584.1	8.00	10.20	1.28×
seq	7.08	9.41	1.33×
spla	6.50	6.60	1.02×
tseng	10.50	16.70	1.59×
Average:	**8.94**	**10.50**	**1.17×**

Πίνακας 5.4: Αξιολόγηση απόδοσης της εφαρμογής όταν απεικονίζεται με διαφορετικά πλαίσια.

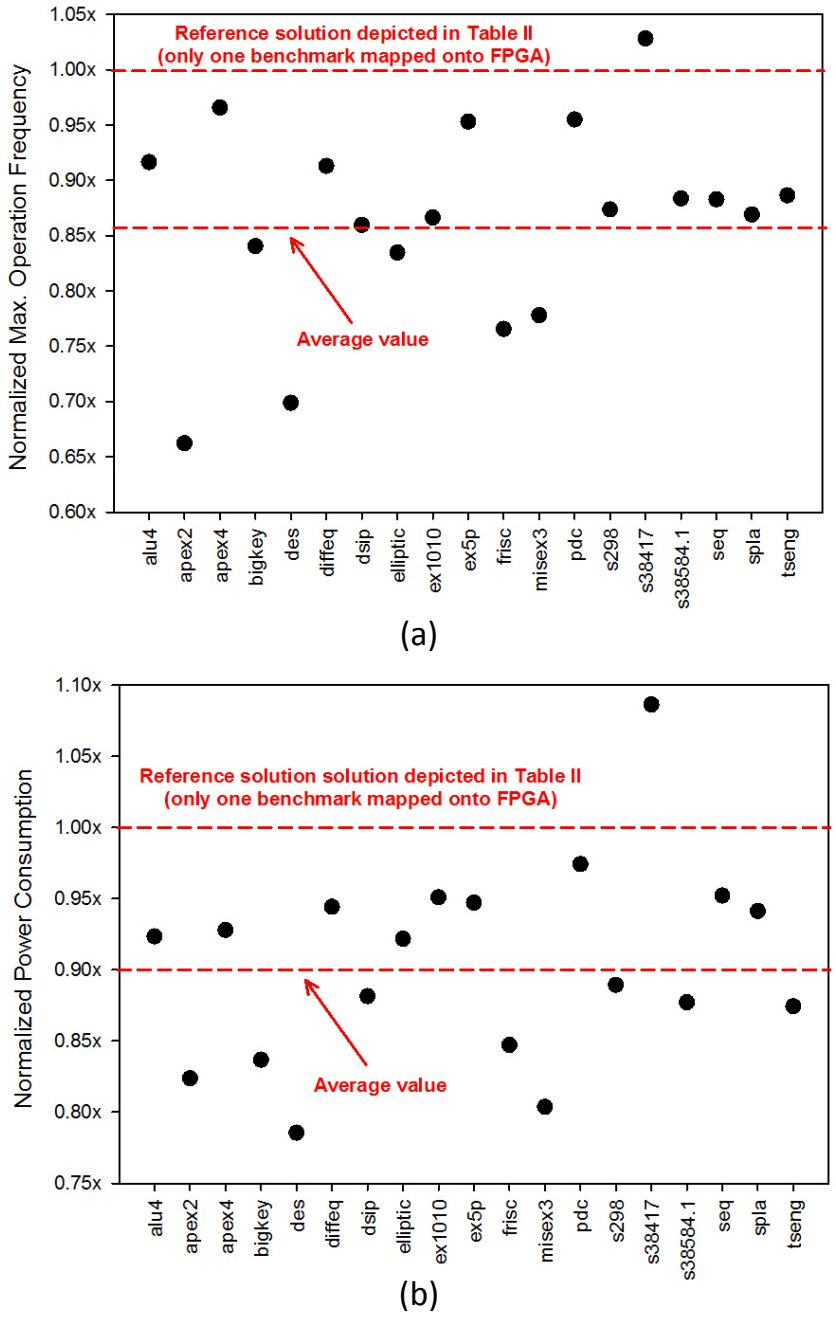
Benchmark	**Power Consumption (mWatt)**					
	Without Scaling			**With Scaling**		
	VPR	**JiTPR**	**Penalty**	**VPR**	**JiTPR**	**Penalty**
alu4	20.30	21.40	1.05×	16.94	21.40	1.26×
apex2	14.20	22.80	1.61×	14.20	19.46	1.37×
apex4	11.20	11.60	1.04×	9.94	11.60	1.17×
bigkey	27.70	42.34	1.53×	27.70	23.69	0.86×
des	19.30	38.21	1.98×	19.30	21.92	1.14×
diffeq	10.20	10.80	1.06×	10.20	9.19	0.90×
dsip	27.10	61.23	2.26×	27.10	32.14	1.19×
elliptic	19.00	20.90	1.10×	16.23	20.90	1.29×
ex1010	10.40	12.00	1.15×	10.22	12.00	1.17×
ex5p	11.60	11.40	0.98×	10.13	11.40	1.13×
frisc	11.30	14.32	1.27×	11.30	12.41	1.10×
misex3	16.80	22.00	1.31×	16.21	22.00	1.36×
pdc	19.20	25.30	1.32×	19.20	24.15	1.26 ×
s298	13.10	16.10	1.23×	13.10	16.05	1.23×
s38417	55.50	64.50	1.16×	37.25	64.50	1.73×
s38584.1	124.00	186.34	1.50×	124.00	146.15	1.18×
seq	13.30	24.90	1.87×	13.30	18.73	1.41×
spla	13.70	18.90	1.38×	13.70	18.61	1.36×
tseng	7.79	11.30	1.45×	7.79	7.10	0.91×
Average:	**23.46**	**33.49**	**1.38×**	**21.99**	**27.02**	**1.21×**

5.4.2 JiTPR την υποστήριξη μερικής επαναδιαμόρφωσης.

Η ανάλυση που έγινε μέχρι τώρα αφορά εφαρμογές που απεικονίζονται σε reconfigurable αρχιτεκτονικές ως stand-alone εφαρμογές (με την προϋπόθεση ότι καμία άλλη εφαρμογή δεν έχει απεικονιστεί επάνω στη συσκευή). Ωστόσο, το πλαίσιο είναι σε θέση να υποστηρίξει επίσης loading εφαρμογών και αφαίρεση πάνω στο FPGA, ακόμη και αν πρόσθετες εφαρμογές έχουν ήδη φορτωθεί επάνω στη συσκευή. Για το σκοπό αυτό, αξιολογείται τώρα η αποτελεσματικότητα της προτεινόμενης λύσης στο να χειριστεί εργασίες που σχετίζονται με τη μερική αναδιαμόρφωση (δυναμική εισαγωγή και την αφαίρεση των εφαρμογών).

Προκειμένου να αποφευχθούν τυχόν χρονικές παραβιάσεις στις εφαρμογές που απεικονίζονται πάνω στο FPGA, είναι δυνατόν να καθοριστούν προτεραιότητες στο εργαλείο RegionFinder. Οι προτεραιότητες αυτές περιγράφουν την επιθετικότητα κατά την υλοποίηση εφαρμογής για τη βελτίωση είτε της απόδοσης (δλδ. συχνότητα λειτουργίας, κατανάλωση ενέργειας), ή του χρόνου εκτέλεσης. Σε όλα τα πειράματα, θεωρείται ότι όλα τα κριτήρια έχουν το ίδιο βάρος.

Τα αποτελέσματα αυτής της ανάλυσης συνοψίζονται στην Εικόνα 5.7.Πιο συγκεκριμένα, τα Σχήματα 5.7 (α) και 5.7 (β) απεικονίζουν την μέγιστη συχνότητα λειτουργίας



Σχήμα 5.7: Αξιολόγηση του προτεινόμενου πλαισίου όταν πολλαπλές εφαρμογές απεικονίζονται στο FPGA ως προς: (a) μέγιστη συχνότητα λειτουργίας και (b) κατανάλωση ισχύος.

και την κατανάλωση ενέργειας, αντίστοιχα, όταν πολλαπλές εφαρμογές έχουν απεικονιστεί πάνω στο FPGA. Οι δύο όροι παρουσιάζονται με κανονικοποιημένο τρόπο πάνω από τα αντίστοιχα αποτελέσματα του πίνακα 5.3 (σχετικά με την stand-alone υλοποίηση εφαρμογής).

Με βάση τα αποτελέσματα που συνοψίζονται στα σχήματα αυτά, βγαίνει το συμπέρασμα ότι η μέση μέγιστη συχνότητα λειτουργίας και η μέση κατανάλωση ενέργειας που επιτυγχάνεται με την προτεινόμενη λύση είναι κατά 0.86 × και 0,90×, αντίστοιχα, χειρότερες σε σύγκριση με τις αντίστοιχες τιμές που παρουσιάστηκαν προηγουμένως στον πίνακα 5.3. Οι κυρώσεις αυτές συμβαίνουν κυρίως λόγω των δυσκολιών στην δρομολόγηση που τίθενται από τις εφαρμογές που έχουν ήδη απεικονιστεί πάνω στο FPGA. Ακόμη και αν τα αποτελέσματα αυτά δείχνουν μια υποβάθμιση των επιδόσεων, σε σύγκριση με το εργαλείο VPR, πρέπει να λάβουμε υπόψη το σημαντικά χαμηλότερο ποσοστό κατακερματισμού, το οποίο με τη σειρά του, επιτρέπει πρόσθετες εφαρμογές να απεικονιστούν πάνω στο FPGA.

5.5 Συμπεράσματα

Ένα νέο πλαίσιο λογισμικού που υποστηρίζει την εκτέλεση Just-In-Time compilation για εφαρμογές FPGA παρουσιάστηκε σε αυτό το κεφάλαιο. Αντί παρόμοιων προσεγγίσεων, οι οποίες ασχολούνται κυρίως με την προ-υπολογισμένα bitstream, το προτεινόμενο πλαίσιο JiT εκτελεί την απεικόνιση εφαρμογής κατά τον χρόνο εκτέλεσης, που οδηγεί σε αμελητέο κατακερματισμό των πόρων. Πειραματικά αποτελέσματα, δείχνουν ότι η προτεινόμενη λύση επιτυγχάνει μια μέση ταχύτητα στο P&R της εφαρμογής 53×, σε σύγκριση με το state-of-the-art εργαλείο VPR, ενώ για συγκεκριμένα benchmarks το κέρδος κυμαίνεται μέχρι και 80,75×. Επιπρόσθετα τα οφέλη αυτά έρχονται με κατά μέσο όρο 1,17× υψηλότερη συχνότητα λειτουργίας.

Κεφάλαιο 6

Απεικόνιση εφαρμογών σε FPGAs κατά το χρόνο εκτέλεσης σε περιβάλλον cloud

6.1 Εισαγωγή

Τα σύγχρονα συστήματα πληροφορικής συχνά απαιτούν περισσότερη υπολογιστική ισχύ λόγω της αυξημένης πολυπλοκότητας. Η συνολική ποσότητα των πληροφοριών που προστίθενται κάθε χρόνο στον παγκόσμιο ιστό αυξάνεται εκθετικά, κάτι το οποίο απαιτεί τεράστια data centers που απαιτούν με την σειρά τους τεράστια ποσότητα ενέργειας και φυσικούς πόρους.

Παραδείγματα εφαρμογών που απαιτούν τέτοια υπολογιστική ισχύ μπορεί να προέλθουν από πεδία που απαιτούν High Performance Computing (HPC), αλλά και εφαρμογές από τον τομέα των ενσωματωμένων συστημάτων μπορούν να ωφεληθούν από την αύξηση των υπολογιστικών πόρων. Παραδείγματα αποτελούν οι εφαρμογές για την μοντελοποίηση των καιρικών συνθηκών, κρυπτογράφησης, εξόρυξης big data, η μοντελοποίηση μορίων γονιδίων και πρωτεϊνών και στον τομέα της ασφάλειας που έχουμε έλεγχο των πακέτων δεδομένων.

Αρχικά η υπολογιστική αυτή ανάγκη αντιμετωπιζόταν τοπικά στις επιχειρήσεις με την επέκταση των hardware υποδομών τους, αλλά δημιουργήθηκαν νέα προβλήματα. Η διαχείριση των server farms, η κατανάλωση και κατανομή ενέργειας, η υποδομή ψύξης και οι χωρικές απαιτήσεις είναι μερικά μόνο από τα προβλήματα, οι εταιρείες αυτές αντιμετωπίζουν.

Στο μοντέλο cloud όλα αυτά τα προβλήματα μπορούν να αποφευχθούν, με την ενοικίαση χρόνου εκτέλεσης σε clusters υπολογιστών, που προσφέρονται από εξειδικευμένες εταιρείες. Αν και αυτό απαλλάσσει τις εταιρείες από κόστη μηχανοργάνωσης δεν μετριάζει τα προβλήματα που αναφέρθηκαν προηγουμένως.

Οι νέες στρατηγικές και μεθοδολογίες έχουν αναπτυχθεί στις μέρες μας για την αξιοποίηση των μαζικά παράλληλων και ετερογενών συστημάτων, όπως clusters, clouds, και φάρμες από CPUs, και/ή GPUs κλπ. Χαρακτηριστικό της τάσης αυτής είναι το γεγονός ότι εμπορικές επιχειρήσεις όπως η Amazon [28] (Amazon Elastic Compute Cloud) που ενοικιάζουν υπολογιστικούς πόρους, όχι μόνο στη βιομηχανία αλλά και σε καταναλωτές. Πιο συγκεκριμένα η Amazon προσφέρει μια απλή Queue Service (Amazon SQS), η οποία διαχειρίζεται τα μηνύματα και τις αιτήσεις, την Amazon SimpleDB η οποία είναι μια βάση δεδομένων για την αποθήκευση των ενδιάμεσων καταστάσεων και των δεδομένων, και το Amazon EC^2, όπου η πραγματική επεξεργασία λαμβάνει χώρα. Στο EC^2, ένας χρήστης μπορεί να αγοράσει πόρους υπολογιστικής ισχύος με τη μορφή εικονικών CPUs και EC^2 Compute Units με χρονοχρέωση ανά ώρα.

Υπάρχει μεγάλο ενδιαφέρον να χρησιμοποιηθούν πολλαπλά FPGAs ως επιταχυντές, πλατφόρμες προτυποποίησης, και μονάδες επεξεργασίας [29] [30] [31]. Αυτό οφείλεται στο ελκυστικό trade-off μεταξύ απόδοσης και κατανάλωσης ισχύος, μιας και επόμενη γενιά των συστημάτων HPC προβλέπεται να υπερβαίνει τα 10MW σε κατανάλωση ισχύος [32]. Από την άλλη πλευρά τα FPGAs χαρακτηρίζονται από μειωμένη παραγωγικότητα, δεδομένου ότι ασυμβατότητες μεταξύ των συσκευών και ροές εργαλείων απαγορεύουν την φορητότητα εφαρμογών και περιορίζουν το reusability.

Σε αυτήν την εργασία μια προτείνεται μια μεθοδολογία cloud και το λογισμικό πλαίσιο υποστήριξης με στόχο να καταστεί δυνατή η αποτελεσματική απεικόνιση πολλαπλών εφαρμογών κατά το χρόνο εκτέλεσης σε μία απλή ή πολλαπλή FPGAs. Ο χρήστης/σχεδιαστής παρέχει την netlist μιας εφαρμογής και αποστέλλει ένα αίτημα για reconfigurable πόρους. Το λογισμικό πλαίσιο ελέγχει τους διαθέσιμους πόρους από μια βάση δεδομένων, αναθέτει ένα τμήμα αυτών των πόρων και απεικονίζει την εφαρμογή. Αυτό είναι εφικτό με την δημιουργία στιγμιότυπων Virtual FPGA (V-FPGA) σε FPGAs και ταυτόχρονα, όταν ζητηθεί απεικόνιση εφαρμογών πάνω τους. Οι συνεισφορές της προτεινόμενης μεθοδολογίας συνοψίζονται ως εξής:

- Η φορητότητα μεταξύ διαφορετικών FPGA πλατφορμών γίνεται δυνατή με την χρήση Virtual FPGA πυρήνων που απεικονίζονται στα πραγματικά FPGAs.
- Γρήγορος χρόνος εκτέλεσης κατάλληλος λόγω του καινοτόμου εργαλείου, CoreMapper σε συνδυασμό με ένα πολύ επιθετικό P&R βήμα.
- Το λογισμικό πλαίσιο κλιμακώνεται με σκοπό να υποστηρίξει μεγάλο φόρτο από αιτήσεις εφαρμογών. Αυτό είναι εφικτό μέσω πολλαπλών ανεξάρτητων στιγμιοτύπων των εργαλείων που λειτουργούν με μια pipelined λογική.

Το υπόλοιπο αυτού του κεφαλαίου οργανώνεται ως εξής. Η ενότητα 2 περιγράφει την προτεινόμενη εικονική αρχιτεκτονική. Η ενότητα 3 παρέχει μια περιγραφή της μεθοδολογίας και του προτεινόμενου πλαισίου υποστήριξης. Πειραματικά αποτελέσματα που αποδεικνύουν την αποτελεσματικότητα της μεθοδολογίας συζητιούνται στην ενότητα 4. Τέλος τα συμπεράσματα συνοψίζονται στην ενότητα 5.

6.2 Virtual FPGA Αρχιτεκτονική

Ένας πυρήνας V-FPGA είναι το βασικό στοιχείο επεξεργασίας στο προτεινόμενο πλαίσιο. Όπως ένα εικονικό CPUs σε ένα υπολογιστικό cloud, όλοι οι πυρήνες V-FPGA είναι πανομοιότυποι, ομοιογενείς και πολλαπλά στιγμιότυπα μπορούν να απεικονιστούν σε ένα FPGA.

Έχει αναπτυχθεί σε VHDL με επεκτάσιμο και παραμετροποιήσιμο τρόπο χρησιμοποιώντας generics και automated GENERATE βρόχους και είναι ανεξάρτητο από την πλατφόρμα. Με αυτή την προσέγγιση μπορούν να υποστηριχτούν χαρακτηριστικά, όπως δυναμική και fine-grain partial reconfiguration για οποιαδήποτε πλατφόρμα FPGA, ακόμη και αν δεν υποστηρίζει αυτές τις λειτουργίες εγγενώς. Επιπλέον, η πλατφόρμα είναι retargetable, αφού απεικονισμένες εφαρμογές μπορούν να ξαναχρησιμοποιηθούν με τα ίδια bitstreams σε άλλες πλατφόρμες εφόσον ένα V-FPGA με τις σωστές παραμέτρους είναι δημιουργημένο.

Η αρχιτεκτονική του εικονικού FPGA είναι island based και αποτελείται κυρίως από CLBs (Configurable Logic Blocks), PSMs (Programmable Switch Matrices) και την IOBs (I / O Blocks). Τα CLBs και οι PSMs είναι τοποθετημένα σε ένα 2-διαστάσεων πλέγμα. Διαύλοι δρομολόγησης μεταξύ των PSMs περιβάλλουν τα CLBs και παρέχουν καλώδια δρομολόγησης που είναι προσβάσιμα από τα CLBs μέσω connection boxes. Τα IOBs βρίσκονται στα άκρα του εικονικού FPGA και συνδέουν σήματα από τα εξωτερικά κανάλια δρομολόγησης με τα εικονικά I/O.

Τα CLBs βασίζονται σε LUTs 4 εισόδων (Lookup πίνακες) και έτσι μπορούν να πραγματοποιήσουν οποιαδήποτε Boolean συνάρτηση με 4 μεταβλητές. Οι συναρτήσεις με περισσότερες μεταβλητές κατανέμονται σε διαφορετικά CLBs, ωστόσο, η αρχιτεκτονική μπορεί επίσης εύκολα να επεκταθεί και σε ομαδοποίηση των διαφόρων LUTs μέσα σε ένα CLB. Τα D-FlipFlops μέσα τα CLBs επιτρέπουν χρονισμένες και σειριακές διαδικασίες, ωστόσο, μπορούν να παρακαμφθούν με πολυπλέκτες για καθαρά συνδυαστικής λογικής κυκλώματα. Υπάρχουν δύο τύποι connection boxes που χρησιμοποιούνται για την παροχή των CLBs επιλεκτικής πρόσβασης στα καλώδια δρομολόγησης, ένα για την ανάγνωση και ένα για γράψιμο. Η υλοποίηση γίνεται με πολυπλέκτες. Τα ίδια connection boxes χρησιμοποιούνται επίσης για IOBs, τα οποία δίνουν πρόσβαση στα εξωτερικά κανάλια δρομολόγησης.

Τα PSMs μεταξύ των καναλιών δρομολόγησης αποτελούνται από πολυπλέκτες που μπορεί να διασυνδέσουν καλώδια από διαφορετικά αλλά παρακείμενα κανάλια δρομολόγησης. Το Wilton scheme χρησιμοποιείται για τη δομή διασύνδεσης καθώς παρέχει υψηλό routability [5].

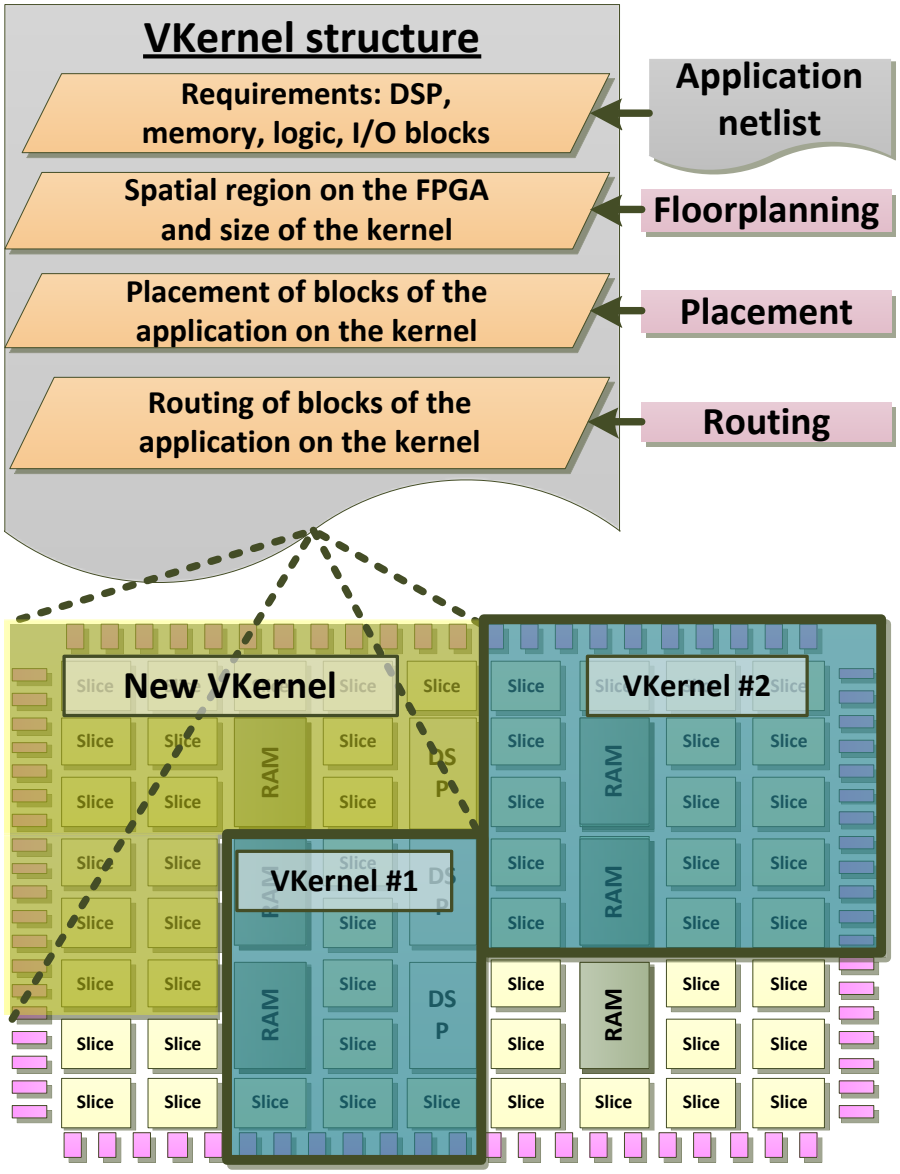
Όλοι οι τύποι των τριών στοιχείων - CLBs, PSMs και IOBs - είναι παραμετροποιήσιμοι από τα bitstreams. Ως εκ τούτου, περιέχουν μονάδες διαμόρφωσης που βασίζονται σε καταχωρητές ολίσθησης που μπορούν να φορτωθούν με τα αντίστοιχα bitstreams. Η διαμόρφωση γίνεται με έναν ημι-παράλληλο τρόπο που παρέχει ένα καλό trade-off ανάμεσα στην ταχύτητα και το εμβαδόν του διαύλου διαμόρφωσης. Όλα τα στοιχεία του ίδιου τύπου μέσα σε μια στήλη προγραμματίζονται σειριακά - οι μονάδες διαμόρφωσής τους είναι daisy chained - αλλά τα σήματα διαμόρφωσης όλων των στηλών μπορεί να οδηγηθούν παράλληλα την ίδια στιγμή. Αυτό σημαίνει ότι ο χρόνος διαμόρφωσης ολόκληρου του V-FPGA ισούται με το χρόνο διαμόρφωσης μίας στήλης. Αρχικά προκαθορισμένα bits στα bitstreams, ελέγχουν τους μηχανισμούς παράκαμψης στις μονάδες διαμόρφωσης, έτσι fine grain partial reconfiguration επιτυγχάνεται σε επίπεδο slice. Περισσότερες λεπτομέρειες σχετικά με την αρχιτεκτονική του V-FPGA και τους μηχανισμούς αναδιαμόρφωσης μπορούν να βρεθούν στο [34] και [89].

Όταν φτιάχνονται SoCs με αρκετούς πυρήνες V-FPGA, υπάρχει συνήθως ένα trade-off όσον αφορά το μέγεθος του κάθε πυρήνα. Φαίνεται σοφό να επιλεγεί ένα μέγεθος όπου η μεγαλύτερη εφαρμογή ταιριάζει ακριβώς. Ωστόσο, έχοντας ένα σύνολο από διαφορετικού μεγέθους εφαρμογές προς απεικόνιση, αυτό δεν είναι ο πιο αποτελεσματικός τρόπος επειδή για μικρές εφαρμογές μπορεί να υπάρχουν πολλά CLBs αναξιοποίητα. Γι αυτό τον λόγο η αρχιτεκτονική υποστηρίζει την τεχνική "CoreFusion" [89], μια μέθοδο που επιτρέπει να συγχωνευτούν διπλανοί πυρήνες σε ένα μεγαλύτερο πυρήνα όταν χρειάζεται. Η τεχνική CoreFusion όταν χρησιμοποιείται με πολλούς μικρού/μεσαίου μεγέθους πυρήνες και όχι λίγους μεγάλους πυρήνες παρέχει μεγαλύτερη ευελιξία στην κατανομή πόρων και βελτιώνει την συνολική αξιοποίηση και τον τον αριθμό των εφαρμογών που μπορούν να τρέχουν ταυτόχρονα.

6.3 Προτεινόμενη Μεθοδολογία

Αυτή η ενότητα περιγράφει την προτεινόμενη μεθοδολογία που απεικονίζεται στο Σχήμα 6.1. Όπως είναι σύνηθες σε ένα σύστημα cloud computing, δεν υπάρχει καμία γνώση σχετικά με τους πόρους που απαιτούνται από τις μελλοντικές εφαρμογές και οι διαθέσιμοι πόροι σε μία μελλοντική χρονική στιγμή. Κάθε φορά που μια νέα εφαρμογή ζητά πόρους, εισέρχεται σε ένα σύστημα ουράς ως εργασία απεικόνισης. Ένας scheduler διαχειρίζεται αυτήν την ουρά, αφαιρεί τις τελειωμένες εργασίες, ελέγχει για την διαθεσιμότητα των πόρων και αναλαμβάνει την ανάκτηση πληροφοριών από μια βάση δεδομένων. Το χρονοδιάγραμμα αποστέλλει σειριακά νέες εργασίες στο επόμενο εργαλείο του προτεινόμενου πλαισίου, το CoreMapper, το οποίο ενεργεί ως εκχωρητής πόρων.

Στο πεδίο του παρόντος ερευνητικού έργου ο scheduler υλοποιεί μια ουρά FIFO, αλλά έχει επίσης την ευελιξία να υποστηρίξει προτεραιότητες.



Σχήμα 6.1: Μια αφηρημένη θεώρηση της προτεινόμενης δομής ενός VKernel και της μεθοδολογίας για την παραγωγή των απαραίτητων πληροφοριών για την απεικόνιση εφαρμογής σε πυρήνες V-FPGA.

6.3.1 CoreMapper

Ένα από τα πιο κρίσιμα βήματα της μεθοδολογίας υποστηρίζεται από ένα νέο εργαλείο, που ονομάζεται CoreMapper. Ο CoreMapper επιλέγει μια περιοχή που αποτελείται από έναν ή περισσότερους πυρήνες V-FPGA όπου η νέα εφαρμογή θα απεικονιστεί, και

αυτή η πληροφορία στη συνέχεια παρέχεται στο βήμα τοποθέτησης και δρομολόγησης. Δεδομένου ότι ένας αριθμός από άλλες εφαρμογές μπορεί να είναι ήδη πάνω σε πυρήνες V-FPGA, ο CoreMapper έχει επίγνωση των διαθέσιμων (άδειων) εικονικών πόρων υλικού. Για το σκοπό αυτό, η τρέχουσα κατάσταση της αρχιτεκτονικής ανακτάται από μία βάση δεδομένων. Αυτή η βάση δεδομένων παρέχει πληροφορίες σχετικά με τους ελεύθερους πόρους σε κάθε πυρήνα του V-FPGA σε επίπεδο slice.

Ο Core Mapper υποθέτει αρχικά ότι όλοι οι πυρήνες V-FPGA αρχικοποιούνται σε μια πραγματική FPGA είναι μία ενοποιημένη reconfigurable περιοχή. Σε αυτόν τον ενιαίο χώρο όλες οι ελεύθερες περιοχές ελέγχονται με έναν γρήγορο τροποποιημένο αλγόριθμο flood-fill προκειμένου να καθοριστεί η διαθεσιμότητα των πόρων (λογικής και I/O μπλοκ). Οι αποδεκτές περιοχές στη συνέχεια αξιολογούνται με μία συνάρτηση κόστους, που γνωρίζει τους καταλυμένους πόρους μέσα σε κάθε επιμέρους πυρήνα V-FPGA. Αυτή η συνάρτηση κόστους λαμβάνει υπόψη τρεις παραμέτρους:

1. Το πόσο συμπαγής είναι η επιλεγμένη περιοχή.
2. Τον κατακερματισμό κάθε V-FPGA core.
3. Τον αριθμό V-FPGA πυρήνων που απαιτείται για την απεικόνιση της εφαρμογής.

Όσο πιο συμπαγής η επιλεγμένη περιοχή, τόσο μεγαλύτερη είναι η πιθανότητα ότι μπλοκ της εφαρμογής θα πρέπει να απεικονιστούν πιο κοντά με αποτέλεσμα την υψηλότερη απόδοση (δηλ. χαμηλότερη κατανάλωση ενέργειας και χαμηλή καθυστέρηση). Από την άλλη πλευρά μία τέτοια επιλογή εισάγει υψηλά ποσοστά κατακερματισμού των πόρων σε ένα σύστημα όπου οι εφαρμογές φορτώνονται και να αφαιρούνται δυναμικά κατά το χρόνο εκτέλεσης.

Μετά τον προσδιορισμό της περιοχής πάνω στους πυρήνες V-FPGA, ο CoreMapper περνάει την περιοχή αυτή, στην βάση δεδομένων πόρων. Αυτό αποτρέπει άλλες εφαρμογές από τη χρήση αυτών των "κλειδωμένων" πόρων. Η περιοχή παραμένει "κλειδωμένη" κατά τη διάρκεια του σταδίου τοποθέτησης και δρομολόγησης. Στη συνέχεια στο αρχείο XML οι πληροφορίες σχετικά με τους πόρους της ενημερώνονται με τους πραγματικά κατεχόμενους πόρους (δηλ. Logic Blocks, I/O μπλοκ και διαδρομές δρομολόγησης).

Η τεχνική που συζητήθηκε σε αυτήν την ενότητα εξασφαλίζει ότι το P&R βήμα κάθε εφαρμογής είναι ανεξάρτητο επιτρέποντας έτσι την παράλληλη εκτέλεση της P&R για διάφορες εφαρμογές. Αυτό το βήμα καταναλώνει το περισσότερο χρόνο, αλλά η παραλληλοποίηση αυξάνει την απόδοση σε μεγάλο βαθμό.

6.3.2 Τοποθέτηση και δρομολόγηση

Η λειτουργικότητα του προτεινόμενου placer βασίζεται σε ένα γρήγορο simulated annealing αλγόριθμο, παρόμοιο με εκείνο που χρησιμοποιείται στο εργαλείο VPR [5], που παρουσιάζεται στην δημοσίευση [37]. Η αρχική τοποθέτηση των λογικών μπλοκ βελτιστοποιείται από την εναλλαγή ζευγών μπλοκ με στόχο να βρεθούν ενδιάμεσες λύσεις που έχουν χαμηλότερο συνολικό κόστος. Άπληστη αποδοχή βελτίωσης του κόστους οδηγεί συχνά σε ενδιάμεσες τοποθετήσεις που είναι τοπικά βέλτιστες, αλλά μπορεί να είναι μακριά από την ολική βέλτιστη λύση. Για να αποφευχθεί αυτό, οι αλγόριθμοι annealing δέχονται ένα ποσοστό από κινήσεις που οδηγούν σε υψηλότερου κόστους λύσεις. Ο χρόνος εκτέλεσης του placer έχει βελτιωθεί σημαντικά λόγω:

i. Της μείωσης του αριθμού των κινήσεων σε σύγκριση με τον αρχικό VPR αλγόριθμο.

ii. Της επιλογής (μέσω του CoreMapper) μιας προκαθορισμένης περιοχής για την απεικόνιση της εφαρμογής προκειμένου να μειωθεί ο χώρος λύσεων κατά τη διάρκεια της τοποθέτησης.

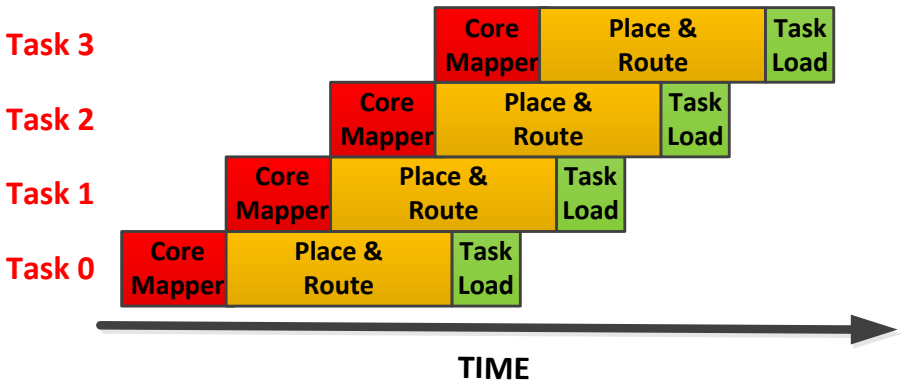
Εκτός από τον placer, το πλαίσιο ενσωματώνει έναν router που προσδιορίζει τη δρομολόγηση και τους διακόπτες που πρέπει να χρησιμοποιηθούν για να δημιουργήσουν συνδεδεμένα μονοπάτια από net sources σε net sinks για όλα τα δίκτυα του κυκλώματος. Ο αλγόριθμος δρομολόγησης βασίζεται στον PathFinder [5] negotiated congestion αλγόριθμο, τροποποιημένο για ταχύτερους χρόνους εκτέλεσης. Πιο συγκεκριμένα η επιτάχυνση του χρόνου εκτέλεσης επιτυγχάνεται με τον καθορισμό πολύ υψηλού κόστους για την αρχική υπερ-χρήση του καναλιού. Αυτό οδηγεί πολύ γρήγορα σε αποδεκτές λύσεις δρομολόγησης. Ωστόσο, έχει μια μικρή ποινή στον αριθμό καλωδίων δρομολόγησης που χρησιμοποιούνται για τη δρομολόγηση του κυκλώματος της εφαρμογής.

Μετά το στάδιο της τοποθέτησης και δρομολόγησης, όλες οι απαραίτητες πληροφορίες είναι διαθέσιμες. Πιο συγκεκριμένα η έξοδος αυτού του σταδίου είναι ένα μερικό αρχείο bitstream ανά V-FPGA το οποίο ρυθμίζει τα LUTs, τα PSMs και τα μπλοκ I/O στις στήλες όπως περιγράφεται στην ενότητα 6.2.

6.3.3 Κλιμάκωση

Η προτεινόμενη μεθοδολογία έχει αναπτυχθεί έτσι ώστε να είναι κλιμακούμενη. Διαφορετικά στιγμιότυπα του πλαισίου μπορούν να τρέξουν απομακρυσμένα και κατανεμημένα ως μέρος ενός περιβάλλοντος cloud στοχεύοντας διαφορετικές FPGA συσκευές. Για κάθε στιγμιότυπο του πλαισίου τα βήματα του CoreMapper, τοποθέτησης και δρομολόγησης μπορούν να γίνουν με έναν pipelined τρόπο, όπως φαίνεται στο Σχήμα 6.2.

Όπως αναφέρεται στο τμήμα 6.3.1 το εργαλείο CoreMapper "κλειδώνει" την περιοχή που θα πρέπει να απεικονιστεί η εφαρμογή. Έτσι, η τοποθέτηση και δρομολόγηση



Σχήμα 6.2: Pipelined εκτέλεση των προτεινόμενων εργαλείων

μπορεί να γίνουν ανεξάρτητα για πολλαπλές εφαρμογές. Όπως παρουσιάζεται επίσης στο σχήμα 6.2 μόνο το στάδιο του CoreMapper πρέπει να εκτελεστεί σειριακά για κάθε εφαρμογή. Αυτό είναι ένα σημαντικό πλεονέκτημα του πλαισίου, δεδομένου ότι το CoreMapper είναι το ταχύτερο βήμα στην ροή των εργαλείων. Μια πιο ενδελεχής ανάλυση των χρόνων εκτέλεσης μπορεί να βρεθεί στο τμήμα 6.4.

6.4 Πειραματικά Αποτελέσματα

Αυτή η ενότητα παρουσιάζει μια σειρά από πειραματικά αποτελέσματα και συγκρίσεις που αποδεικνύουν την αποδοτικότητα της προτεινόμενης λύσης. Για το σκοπό αυτό, χρησιμοποιούνται τα 20 μεγαλύτερα MCNC benchmarks [81], τα οποία είναι ευρέως αποδεκτά στην FPGA κοινότητα. Η πλατφόρμα δοκιμών αποτελείται από έναν επεξεργαστή Intel Xeon με 8 πυρήνες και 8GB μνήμης RAM.

6.4.1 Ανάλυση φορητότητας

Όπως αναφέρθηκε προηγουμένως ένας πυρήνας V-FPGA περιγράφεται VHDL και μπορεί να απεικονιστεί σε μια ποικιλία συσκευών FPGA από διαφορετικούς προμηθευτές. Για το πεδίο έρευνας του παρόντος έργου ένας μικρός πυρήνας V-FPGA απεικονίστηκε με επιτυχία, ο οποίος αποτελούταν από 8×8 CLBs σε Xilinx Virtex 7 (Xc7v2000t flg1925-2 συσκευή), Altera Stratix V (συσκευή 5SEEBF45I2), και Actel ProASIC3 (Συσκευή A3P15000) FPGAs, χρησιμοποιώντας τις προεπιλεγμένες παραμέτρους στα αντίστοιχα εργαλεία της Xilinx, Altera και Actel.

Ο πίνακας 6.1 παρουσιάζει τους πόρους που χρησιμοποιούνται από τον πυρήνα V-FPGA σε κάθε FPGA, την μέγιστη συχνότητα λειτουργίας που επιτυγχάνεται και τον αριθμό των πυρήνων που μπορούν να απεικονιστούν σε κάθε συσκευή. Με βάση τον πίνακα αυτό καταλήγουμε στο συμπέρασμα ότι η χρήση των πόρων και η απόδοση

ενός ενιαίου πυρήνα V-FPGA επηρεάζονται σημαντικά από την πλατφόρμα που απεικονίστηκε. Αυτό συμβαίνει κυρίως επειδή, η αρχιτεκτονική του κάθε FPGA διαφέρει και κάθε πλατφόρμα έχει τα δικά της εργαλεία για, technology mapping τοποθέτηση και δρομολόγηση. Για να επισημανθούν αυτές οι διαφορές, παρακάτω παρατίθενται πληροφορίες για τα λογικά μπλοκ του κάθε προμηθευτή.

Xilinx [90]

Το σειρά 7 Xilinx FPGAs, το configurable logic block (CLB) είναι ο κύριος λογικός πόρος για την υλοποίηση διαδοχικών αλλά και συνδυαστικών κυκλωμάτων. Κάθε CLB στοιχείο είναι συνδεδεμένο σε ένα πίνακα διακοπτών για την πρόσβαση στην γενική μήτρα δρομολόγησης. Ένα CLB αποτελείται από

- Look-up table (LUT) 6 εισόδων.
- Επιλογή για Dual LUT5 (5-input LUT).
- Ικανότητα για κατανεμημένη μνήμη και Shift Register Logic.
- Ειδική high-speed carry logic για αριθμητικές συναρτήσεις.
- Πλατείς πολυπλέκτες για αποδοτική αξιοποίησή τους.

Οι πίνακες Look-Up (LUTs) στην σειρά 7 FPGAs μπορεί να διαμορφωθεί είτε ως ένα 6-input LUT με μία έξοδο, ή ως δύο LUTs 5 εισόδων με ξεχωριστές εξόδους, αλλά κοινές διευθύνσεις ή λογικές εισόδους. Κάθε έξοδος LUT 5-εισόδου μπορεί προαιρετικά να περνά από ένα flip-flop. Τα 4 LUTs 6-εισόδων, τα οκτώ flip-flops τους, καθώς και οι πολυπλέκτες και η αριθμητική carry logic αποτελούν ένα slice, και δύο slices σχηματίζουν ένα CLB. Τέσσερα flip-flops ανά slice (μία ανά LUT) μπορούν προαιρετικά να ρυθμιστούν ως latches. Στην περίπτωση αυτή, τα υπόλοιπα τέσσερα flip-flops σε αυτό το slice πρέπει να παραμείνουν αχρησιμοποίητα. Περίπου τα δύο τρίτα των slice είναι SLICEL λογική slice και τα υπόλοιπα είναι SLICEM, τα οποία μπορούν επίσης να χρησιμοποιήσουν LUTs τους ως κατανεμημένη RAM 64-bit ή ως καταχωρητές ολίσθησης 32-bit (SRL32) ή ως δύο SRL16s.

Altera [60]
Το Logic Array Blocks (LAB) της Altera αποτελείται από βασικά δομικά στοιχεία που είναι γνωστά ως adaptive logic modules (ALMs), που μπορούν να ρυθμιστούν ώστε να εφαρμόσουν λογικές λειτουργίες, αριθμητικές λειτουργίες και τις λειτουργίες register. Τα LABs είναι παραμετροποιήσιμα μπλοκ λογικής που αποτελούνται από μια ομάδα λογικών πόρων. Κάθε LAB περιέχει ειδική λογική για την οδήγηση σημάτων ελέγχου για τα ALMs του.

Κάθε LAB μπορεί να οδηγήσει 30 ALMs μέσω γρήγορων τοπικών και απευθείας συνδέσεων. Η τοπική διασύνδεση μπορεί να οδηγήσει ALMs στο ίδιο LAB χρησιμοποιώντας

διασυνδέσεις στήλης και σειράς. Κάθε ALM περιλαμβάνει μια ποικιλία πόρων LUT που μπορούν να να διαιρεθούν μεταξύ δύο συνδυαστικών προσαρμοστικών LUTs (ALUTs) και τεσσάρων registers.

Με έως οκτώ εισόδους για τα δύο συνδυαστικά ALUTs, ένα ALM μπορεί να εφαρμόσει διάφορους συνδυασμούς των δύο λειτουργιών. Αυτή η προσαρμοστικότητα επιτρέπει σε ένα ALM να είναι εντελώς συμβατό με αρχιτεκτονικές LUT τεσσάρων εισόδων. Ένα ALM μπορεί επίσης να υλοποιήσει οποιαδήποτε συνάρτηση με έως και έξι εισόδους και ορισμένες συναρτήσεις επτά εισόδων.

Actel [91]

Ο πυρήνας ενός Actel FPGA αποτελείται από μια θάλασσα από VersaTiles. Κάθε VersaTile μπορεί να ρυθμιστεί ως συνάρτηση τριών εισόδων, ένα D-flip-flop (με ή χωρίς ενεργοποίηση), ή ένα latch με προγραμματισμό των κατάλληλων flash διακοπτικών διασυνδέσεων. Το ProASIC3 VersaTile υποστηρίζει τα εξής:

- Όλες τις ισοδύναμες συναρτήσεις τριών εισόδων.
- Latch με σήμα clear or set.
- D-flip-flop με σήμα clear or set.
- Enable D-flip-flop σήμα clear or set.

Αν και το V-FPGA έχει αναπτυχθεί για να είναι ανεξάρτητο από την πλατφόρμα, από τα πειράματα καταλήγουμε στο συμπέρασμα ότι η CLB αρχιτεκτονική της Altera είναι η πιο αποτελεσματική για να υποστηρίξει αυτό το εικονικό στρώμα.

Target FPGA	**Basic Logic Elements**	**Max Op. Freq.(MHz)**	**# of V-FPGAs that can fit**
Xilinx xc7v2000 tflg1925-2	7903(0.64%)	36	155
Altera 5SEEBF45I2	4032(1.12%)	267	88
Actel A3P15000	18556(38%)	21	2

Πίνακας 6.1: Αξιοποίηση πόρων και μέγιστη συχνότητα λειτουργίας ενός V-FPGA πυρήνα απεικονισμένου σε διάφορες συσκευές.

6.4.2 Ανάλυση χρόνου εκτέλεσης

Όσον αφορά τα υπόλοιπα πειραματικά αποτελέσματα, θεωρούμε μια σειρά από τέσσερις ταυτόσημους 75×75 πυρήνες V-FPGA, που συνιστούν ένα CLB πίνακα $150\times$, όπου κάθε κανάλι δρομολόγησης περιέχει 50 καλώδια. Το μέγεθος του κάθε πυρήνα V-FPGA, ο αριθμός των πυρήνων και το πλάτος του καναλιού, ορίζονται από ένα αρχείο XML που περιγράφει το εικονική αρχιτεκτονική.

Circuit	Proposed Solution		VPR	Gain
	CoreMapper	P&R		
alu4	9	459	13,458	28.76×
apex2	12	585	22,500	37.69×
apex4	7	317	10,873	33.56×
bigkey	10	1,041	26,599	25.31×
des	9	836	24,916	29.49×
diffeq	9	635	18,156	28.19×
dsip	8	694	16,779	23.90×
elliptic	12	2,403	7,1308	29.53×
es1010	13	1,862	81,575	43.51×
ex5p	10	413	9,533	22.54×
frisc	14	3,221	12,4421	38.46×
misex3	7	362	12,352	33.47×
pdc	13	5,007	11,3703	22.65×
s298	9	742	25,392	33.81×
s38417	24	5,723	222,966	38.80×
s38584.1	17	4,353	159,544	36.51×
seq	12	591	20,484	33.97×
spla	16	1,419	65,512	45.65×
tseng	9	412	12,155	28.87×
Average	**12**	**1,636**	**55,380**	**32**×

Πίνακας 6.2: Χρόνος εκτέλεσης (σε msec) της προτεινόμενης προσέγγισης και του VPR.

Ο χρόνος εκτέλεσης είναι μια κρίσιμη μετρική, δεδομένου ότι η προτεινόμενη μεθοδολογία έχει ως στόχο να παρέχει απεικόνιση εφαρμογών κατά τον χρόνο εκτέλεσης πάνω στην επιλεγμένη αρχιτεκτονική. Ο πίνακας 6.2 ποσοτικοποιεί το χρόνο εκτέλεσης για την τοποθέτηση και την λεπτομερή δρομολόγηση με την χρήση της προτεινόμενης προσέγγισης και το εργαλείο VPR [5]. Το εργαλείο VPR χρησιμοποιήθηκε, δεδομένου ότι είναι αποδεκτό στον ακαδημαϊκό χώρο και είναι αρκετά ευέλικτο ώστε να υποστηρίξει μία FPGA αρχιτεκτονική παρόμοια με την προτεινόμενη Virtual αρχιτεκτονική. Εφόσον η προτεινόμενη μεθοδολογία ψάχνει το σύνολο της εικονικής περιοχής του FPGA για μία κατάλληλη λύση, το VPR προσπαθεί να απεικονίσει κάθε εφαρμογή σε ένα 150×150 CLB πίνακα με πλάτος καναλιού 50. Με βάση τα αποτελέσματα που συνοψίζονται στον πίνακα αυτό, βγαίνει το συμπέρασμα ότι τα προτεινόμενα εργαλεία τοποθετούν και δρομολογούν μια εφαρμογή 32×, κατά μέσο όρο, ταχύτερα από ό,τι το αρχικό VPR [5]. Όπως αναφέρεται στο τμήμα 6.3 αυτό οφείλεται κυρίως στο ό,τι:

i. Ο CoreMapper επιλέγει μια προκαθορισμένη περιοχή για την απεικόνιση της εφαρμογής και

ii. Στον επιθετικό placer που παρουσιάστηκε στο [37].

6.4.3 Πολλαπλές εφαρμογές απεικονισμένες δυναμικά

Μέχρι τώρα, κάθε benchmark απεικονίζεται επάνω στην προτεινόμενη εικονική αρχιτεκτονική ως αυτόνομη εφαρμογή. Στο υπόλοιπο κεφαλαίου, η αποτελεσματικότητα της προτεινόμενης λύσης αξιολογείται ως προς τον χειρισμό πολλαπλών benchmark απεικονισμένων δυναμικά σε V-FPGA πυρήνες. Για το σκοπό αυτό, δημιουργήθηκε μια ουρά που αποτελείται από 200 κυκλώματα (από τα 20 benchmarks MCNC) που περιμένουν να απεικονιστούν δυναμικά πάνω στο εικονικό FPGA.

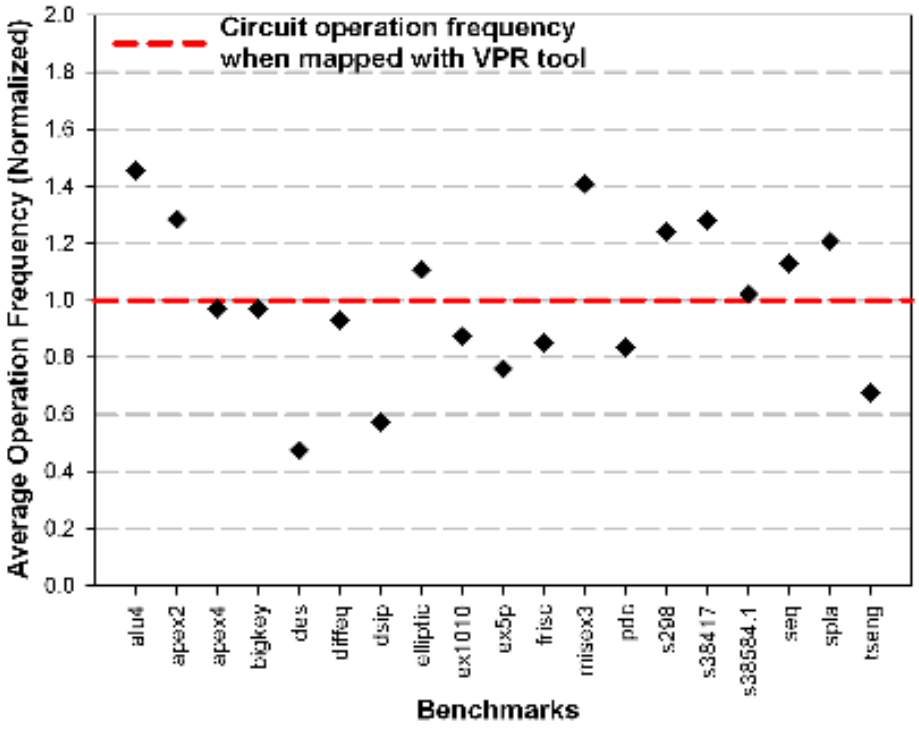
6.4.3.1 Ποιότητα της απεικόνισης των εφαρμογών

Η ευελιξία της επαναδιαμόρφωσης της προτεινόμενης αρχιτεκτονικής,σε επίπεδο slice και η ικανότητα να απεικονιστούν πολλές εφαρμογές σε πολλαπλούς πυρήνες V-FPGA επηρεάζει την μέγιστη συχνότητα λειτουργίας του κάθε κυκλώματος. Κάθε φορά που μια εφαρμογή απεικονίζεται πάνω στην εικονική αρχιτεκτονική, η αξιοποίηση των πόρων και η φυσική θέση των διαθέσιμων πόροι διαδραματίζουν σημαντικό ρόλο στην τελική λύση.

Κατά την πορεία του πειράματος κάθε benchmark απεικονίστηκε αρκετές φορές και η μέση μέγιστη συχνότητα απεικονίζεται στο Σχήμα 6.3. Ως λύση αναφοράς χρησιμοποιείται η απόδοση κάθε benchmark όταν είναι απεικονισμένο πάνω στο FPGA από το VPR ως αυτόνομη εφαρμογή και τα αποτελέσματα κανονικοποιήθηκαν ως προς αυτή την λύση. Αν και υπάρχουν διακυμάνσεις στα αποτελέσματα, κατά μέσο όρο σε σύγκριση με την λύση αναφοράς παρατηρείται μια διαφοροποίηση μόνο 0.26%, η οποία μπορεί να θεωρηθεί αμελητέα. Οι διακυμάνσεις αυτές οφείλονται στο γεγονός ότι ταυτόχρονα πολλαπλά benchmarks απεικονίζονται πάνω στον εικονικό FPGA και αυτό επηρεάζει σημαντικά την τοποθέτηση και δρομολόγηση του καθενός.

6.4.3.2 Ανάλυση κατακερματισμού

Δεδομένου ότι οι εφαρμογές απεικονίζονται και αφαιρούνται δυναμικά κατά το χρόνο εκτέλεσης, με την πάροδο του χρόνου αυτό εισάγει σημαντικό κατακερματισμό πόρων. Το Σχήμα 6.4 δείχνει το ποσοστό κατακερματισμού κατά τη διάρκεια του χρόνου για 4 V-FPGA πυρήνες. Για να υπολογίσουμε το ποσοστό κατακερματισμού χρησιμοποιήθηκε ο τύπος 6.1. Ιδανικά το μεγαλύτερο ελεύθερο μπλοκ θα πρέπει να περιέχει όλους τους ελεύθερους πόρους (κατακερματισμός 0%). Καθώς η μεγαλύτερη συνεχόμενη ελεύθερη περιοχή περιέχει όλο και λιγότερους από τους συνολικά ελεύθερους πόρους, αυτό σημαίνει ότι οι ελεύθεροι πόροι κατακερματίζονται. Φαίνεται από το Σχήμα 6.4, ότι τα προτεινόμενα εργαλεία επιτυγχάνουν ένα μέσο ποσοστό 24%, όταν απεικονίζονται



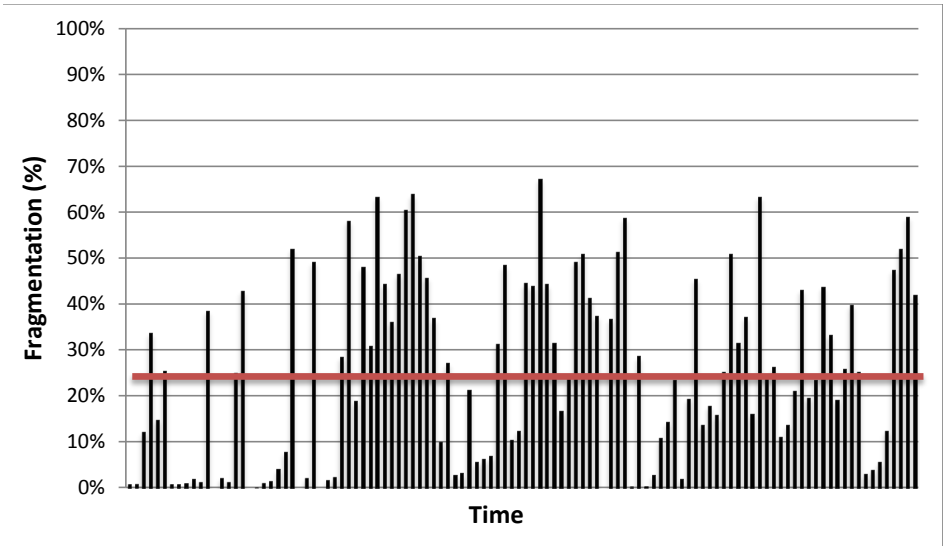
Σχήμα 6.3: Μέση συχνότητα λειτουργίας για πολλαπλές απεικονίσεις benchmarks με το πλαίσιο, κανονικοποιημένη ως προς τις λύσεις VPR.

πολλαπλές εφαρμογές. Αυτός είναι ένας λογικός συμβιβασμός αφού δεν υπάρχει υποβάθμιση στην η ποιότητα του κάθε benchmark, όπως αναφέρεται στην παράγραφο 6.4.3.1.

$$ResourceFragmentation = 1 - \frac{LargestFreeBlock}{TotalFreeResources} \tag{6.1}$$

Where LargestFreeBlock denotes the resources contained by the largest contiguous free area, and TotalFreeResources, denotes the total amount of free resources.

Στο Σχήμα 6.4, το μεγαλύτερο μέρος του χρόνου το ποσοστό κατακερματισμού είναι μικρότερο από 20% αλλά υπάρχουν κορυφές που φτάνουν έως και το 65%. Είναι σημαντικό να σημειωθεί ότι στο πεδία αυτού του ερευνητικού έργου η θέση της κάθε εφαρμογής μετά την απεικόνιση είναι σταθερή. Ο κατακερματισμός θα μπορούσε να βελτιωθεί σημαντικά εάν επιτραπεί κάποια επαναδιαμόρφωση, αλλά ο κύριος στόχος της εργασίας αυτής είναι αποκλειστικά η μείωση του χρόνου εκτέλεσης.



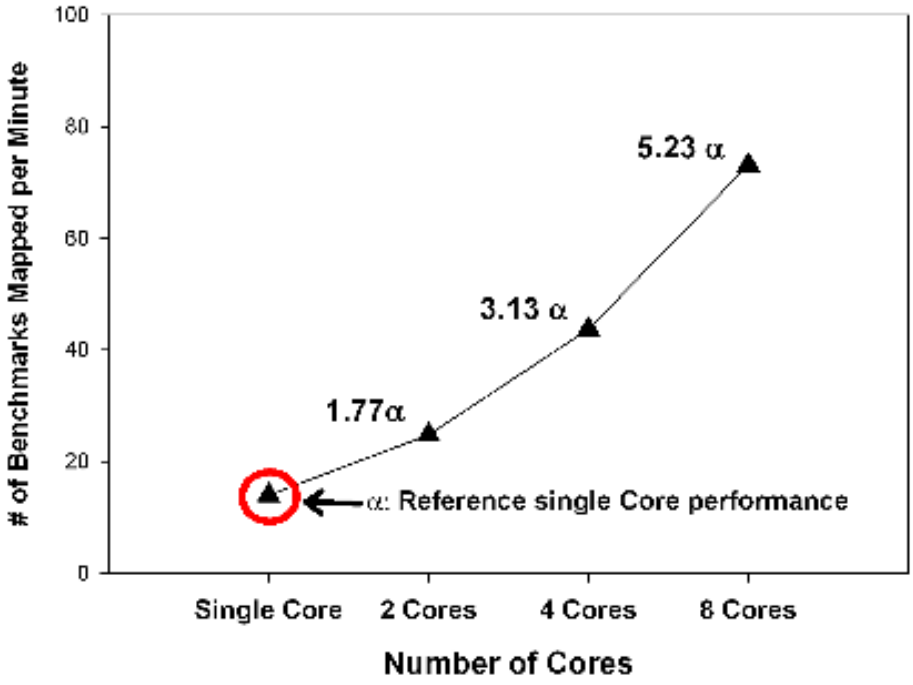
Σχήμα 6.4: Ποσοστό κατακερματισμού στους V-FPGA πυρήνες λόγω απεικόνισης πολλαπλών εφαρμογών.

6.4.3.3 Κλιμάκωση της προτεινόμενης λύσης

Στη συνέχεια, το πλαίσιο αξιολογήθηκε όταν εκτελείται σε ένα σύστημα πολλαπλών πυρήνων. Δύο, τέσσερις και οκτώ πυρήνες χρησιμοποιήθηκαν για να μετρηθεί πόσες εργασίες απεικόνισης μπορούν να εκτελεστούν ανά λεπτό. Για να είναι ακριβείς οι μετρήσεις του φόρτου που το λογισμικό πλαίσιο μπορεί να χειριστεί, ο χρόνος ξεκίνησε όταν το πρώτο benchmark εισήλθε στην ουρά και σταμάτησε όταν το τελευταίο benchmark φορτώθηκε στην εικονική αρχιτεκτονική. Ο χρόνος που ένα benchmark περιμένει να ελευθερωθούν πόροι δεν ελήφθη υπόψιν. Τα αποτελέσματα φαίνονται στο Σχήμα 6.5.

Είναι προφανές ότι η μεθοδολογία και το λογισμικό πλαίσιο μπορούν αποτελεσματικά να αυξήσουν το throughput με την προσθήκη των πόρων υλικού. Όταν εκτελείται σε έναν πυρήνα το πλαίσιο μπορεί να υποστηρίξει έως και 14 αιτήσεις για απεικόνιση εφαρμογών ανά λεπτό. Αυτή η δυνατότητα μπορεί να φτάσει έως και 25 αιτήσεις ανά λεπτό (1.77× περισσότερο) με δύο πυρήνες, 44 αιτήσεις ανά λεπτό (3.13× περισσότερο) με 4 πυρήνες και 73 αιτήσεις ανά λεπτό (5.23 × περισσότερο) με 8 πυρήνες.

Η κλιμάκωση είναι ομαλή και δεν δείχνει σημάδια κορεσμού. Η επικοινωνία μεταξύ των παράλληλων απεικονίσεων συνίσταται μόνο στο κοινό αρχείο με τους ελεύθερους πόρους. Αυτό το αρχείο απαιτεί κλείδωμα μόνο, όταν ένα στιγμιότυπο του CoreMapper βρίσκεται σε λειτουργία, προκειμένου να διατηρήσει κλειδωμένη την περιοχή. Στο τέλος της κάθε απεικόνισης, όταν αυτό το κοινό αρχείο ενημερώνεται δεν υπάρχει ανάγκη κλειδώματος, καθώς, κάθε εφαρμογή έχει πια τους δικούς της πόρους. Αυτό το γεγονός είναι απόδειξη ότι η προτεινόμενη ροή εργαλείων μπορεί να κλιμακωθεί αποτελεσματικά, αποτελώντας έτσι μία βιώσιμη λύση για να υποστηρίξει ένα ετερογενές cloud.



Σχήμα 6.5: Throughput του προτεινόμενου εργαλείου, όταν εκτελείται σε 1, 2, 4, 8 πυρήνες.

6.5 Συμπεράσματα

Μια νέα cloud μεθοδολογία για την υποστήριξη απεικόνισης πολλαπλών εφαρμογών παρουσιάστηκε σε αυτό το κεφάλαιο. Με βάση τα πειραματικά αποτελέσματα, η προτεινόμενη υλοποίηση P&R πραγματοποιείται 32× φορές, κατά μέσο όρο, πιο γρήγορα σε σύγκριση με την state-of-art προσέγγιση, χωρίς επιβάρυνση όσον αφορά στην μέγιστη συχνότητα λειτουργίας. Όταν πολλαπλές εφαρμογές φορτώνονται πάνω στο FPGA έχουμε κατά μέσο όρο 24% ποσοστό κατακερματισμού και η ροή των εργαλείων μπορεί να κλιμακωθεί για να υποστηρίξει έως και 73 εργασίες απεικόνισης εφαρμογών ανά λεπτό.

Κεφάλαιο 7

Απεικόνιση εφαρμογών σε FPGAs κατά το χρόνο εκτέλεσης σε ενσωματωμένα συστήματα

7.1 Εισαγωγή

Τα Field Programmable Gate Arrays (FPGAs) προσφέρουν μια χαμηλής ισχύος ευέλικτη εναλλακτική λύση ως επιταχυντές υλικού λόγω της εγγενούς παραλληλίας τους. Ο επαναπρογραμματισμός, αν και είναι ένα κρίσιμο χαρακτηριστικό τους, χρησιμοποιείται σχεδόν αποκλειστικά στο χρόνο σχεδίασης, λόγω των περιορισμών που επιβάλλονται από τα σύγχρονα εργαλεία CAD που απαιτούν ακόμα και ημέρες για να τρέξουν καθώς και δεκάδες GB μνήμης RAM. Στην παρούσα εργασία προτείνεται μια νέα μεθοδολογία και τα αντίστοιχα εργαλεία CAD για να χρησιμοποιηθούν δυναμικά τα FPGAs κατά το χρόνο εκτέλεσης. Το προτεινόμενο λογισμικό πλαίσιο επιτρέπει την αποτελεσματική απεικόνιση πολλαπλών εφαρμογών σε ετερογενείς FPGA πλατφόρμες.

Με τη χρήση δυναμικών εικονικών πυρήνων, βελτιστοποιήσεις στην μνήμη και προσαρμοσμένους κατανεμητές μνήμης, επιτυγχάνεται άρση των περιορισμών που επιβάλλονται από τα εργαλεία CAD, και παρέχεται ένα proof-of-concept ότι η απεικόνιση εφαρμογών σε FPGAs μπορεί να γίνει και κατά τον χρόνο εκτέλεσης ακόμα και σε ενσωματωμένα συστήματα. Πειραματικά αποτελέσματα αποδεικνύουν την αποτελεσματικότητα της προτεινόμενης λύσης, καθώς επιτεύχθει απεικόνιση 15× ταχύτερα κατά μέσο όρο σε σύγκριση με την state-of-art προσέγγιση, χωρίς υποβάθμιση των επιδόσεων και με 12× κατά μέσο όρο μειωμένη χρήση μνήμης. Επιπλέον παρέχονται επιλογές για την περαιτέρω βελτίωση της απόδοσης του toolflow ανταλλάσσοντας χρόνο εκτέλεσης με χρήση μνήμης, ώστε να ταιριάζει άριστα στους διαθέσιμους πόρους κάθε συστήματος.

7.1.1 Κίνητρα

Υπάρχοντα ηλεκτρονικά ευρείας κατανάλωσης υλοποιούν μια μεγάλη ποικιλία από πυρήνες, συνήθως με ποικίλες λειτουργίες, από συσκευές αναπαραγωγής πολυμέσων μέχρι τηλεπικοινωνιακές πλατφόρμες. Πρέπει να σημειωθεί είναι ότι σε πολλές περιπτώσεις αυτές οι λειτουργίες δεν είναι γνωστές κατά το χρόνο σχεδιασμό, δεδομένου ότι καθορίζονται από τον τελικό χρήστη. Αυτό το φαινόμενο συμβαίνει όχι μόνο στο τομέα των ενσωματωμένων συστημάτων (έξυπνα τηλέφωνα που χρησιμοποιούνται ως κονσόλες παιχνιδιών, έξυπνες τηλεοράσεις χρησιμοποιούνται ως προγράμματα περιήγησης στο Web, κλπ), αλλά και στον τομέα του High Performance Computing (HPC), δεδομένου ότι τα υπολογιστικά κέντρα πια δεν συνδέονται με συγκεκριμένες εφαρμογές. Σε όλες τις προηγούμενες περιπτώσεις, τα FPGAs μπορούν να εκπληρώσουν το ρόλο ενός ευέλικτου επιταχυντή υλικού που διαμορφώνεται ανάλογα με τις προτιμήσεις του χρήστη, λόγω της εγγενούς τους δυνατότητας να επαναπρογραμματιστούν.

Για να επιτευχθεί αυτό, θα πρέπει να υποστηριχτεί μια ταχύτερη και πιο ευέλικτη υλοποίηση εφαρμογών που δεν είναι αγκιστρωμένη εξ ολοκλήρου στην φάση του σχεδιασμού της εφαρμογής. Δεδομένου ότι σε πολλά σενάρια σήμερα ο τελικός χρήστης επιλέγει τη λειτουργικότητα και τη χρήση των πόρων επεξεργασίας που διαθέτει (είτε σε ενσωματωμένες είτε cloud-based εφαρμογές), υπάρχει ανάγκη πολλαπλοί επιταχυντές, που δεν είναι γνωστοί εκ των προτέρων, να συνυπάρξουν σε ένα FPGA.

Ενα τέτοιο σενάριο παρουσιάζει δυο προκλήσεις:

- Πως να απεικονιστούν πολλαπλά ανεξάρτητα κυκλώματα σε ένα FPGA.
- Πως να εκτελεστεί η απεικόνιση εφαρμογών στην πλατφόρμα του χρήστηα αφού δεν υπάρχει a priori γνώση αυτών κατά τον χρόνο σχεδιασμού.

Η πρώτη πρόκληση μεταφράζεται σε εξεύρεση κατάλληλης περιοχής πάνω στο FPGA για κάθε πυρήνα υλικού, χωρίς την εισαγωγή κατακερματισμού στους πόρους ή περιορισμούς σε μελλοντικούς πυρήνες που θα πρέπει να απεικονιστούν. Αυτές οι προκλήσεις έχουν εν μέρει αντιμετωπιστεί στο [37], όπου μια περιοχή καθορίζεται για την απεικόνιση κάθε πυρήνα λαμβάνοντας υπόψη πυρήνες που έχουν ήδη απεικονιστεί πάνω στο FPGA. Η εργασία αυτή επεκτάθηκε, εισάγοντας μια δυναμική δομή εικονικού πυρήνα, που ονομάζεται VKernel, η οποία υποστηρίζει σύγχρονες πολύπλοκες ετερογενείς αρχιτεκτονικές FPGA όπου αποτελούνται από προσαρμοστικής λογικής μπλοκ, μη ομοιόμορφη δρομολόγηση και ετερογενή DSP και μνήμη RAM μπλοκ.

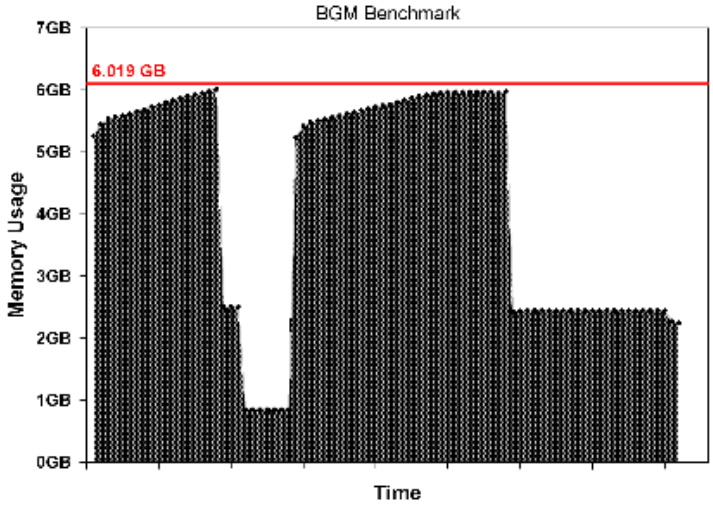
Η δεύτερη πρόκληση μεταφράζεται σε ελαχιστοποίηση του χρόνου εκτέλεσης και των υπολογιστικών πόρων των P&R βημάτων της απεικόνισης του κάθε πυρήνα. Αυτά τα βήματα πρέπει να είναι σε θέση να γίνονται κατά τον χρόνο εκτέλεσης στην πλατφόρμα του τελικού χρήστη. Σύγχρονες ροές εργαλείων FPGA τόσο στον ακαδημαϊκό χώρο όσο

και στη βιομηχανία [16], [6] απαιτούν σημαντική επεξεργαστική ισχύ, μνήμη και μεγάλους χρόνους εκτέλεσης, καθιστώντας την εκτέλεση τους κατάλληλη μόνο για σταθμούς εργασίας και servers, συνδέοντας έτσι το βήμα της απεικόνισης μιας εφαρμογής άρρηκτα με τον χρόνο σχεδιασμού της.

7.1.2 Ορισμός Προβλήματος

Η παραδοσιακή στρατηγική σχεδιασμού όταν χρησιμοποιούνται πλατφόρμες FPGA ως επιταχυντές, είναι να χρησιμοποιηθούν ένα, ή πολλαπλά, FPGA/s λόγω του εγγενούς παραλληλισμού τους και να επιταχύνουν μία εφαρμογή, ή έναν υπολογιστικό πυρήνα, μέρος της εφαρμογής. Παρά το γεγονός ότι αυτό το σενάριο είναι ιδανικό σε περιβάλλοντα HPC, σε ενσωματωμένα συστήματα τελικού χρήστης δεν υπάρχει ανάγκη για τέτοια, μαζικά παράλληλη, επιτάχυνση μιας μοναδικής εφαρμογής.

Όπως αναφέρθηκε προηγουμένως υπάρχει ανάγκη για ένα δυναμικό περιβάλλον που να υποστηρίζει και να επιταχύνει πολλαπλές εφαρμογές. Στα σημερινά εμπορικά ([16], [6]) και ακαδημαϊκά εργαλεία (VTR [14]), δεν υπάρχει άμεση στήριξη για τη απεικόνιση πολλαπλών ανεξάρτητων εφαρμογών πάνω σε ένα FPGA. Πιο συγκεκριμένα στο ακαδημαϊκό εργαλείο VTR, το οποίο είναι μια ώριμη ευρέως αποδεκτή ροή εργαλείων FPGA, ο μόνος τρόπος για να απεικονίσει κανείς πολλαπλές εφαρμογές είναι η συγχώνευση της περιγραφή τους σε επίπεδο HDL και η εκτέλεση της ροής των εργαλείων από την αρχή. Αυτό είναι αναποτελεσματικό και στον πραγματικό κόσμο το σενάριο αυτό θα μπορούσε να εκτελεστεί μόνο μία φορά, στο χρόνο σχεδίασης.



Σχήμα 7.1: Αποτύπωμα Μνήμης του VPR 7.0 εργαλείου κατά τη απεικόνιση του bgm benchmark.

Ένα σημαντικό, αλλά που συχνά παραβλέπεται μέρος των P&R βήματων είναι το αποτύπωμα μνήμης των εργαλείων. Οι μοντέρνες εφαρμογές και τα FPGAs ξεπερνάνε το

όριο των εκατομμυρίων πυλών λογικής και έτσι το κύριο bottleneck της ώρας εκτέλεσης των εργαλείων είναι η χρήση της μνήμης και όχι οι αλγόριθμοι που χρησιμοποιούνται σε κάθε βήμα. Για παράδειγμα ο πίνακας 7.1 δείχνει το αποτύπωμα της μνήμης του εργαλείου VPR 7.0 (το P& R εργαλείο του VTR) κατά την απεικόνιση των εφαρμογών σε ένα μεσαίου μεγέθους, μεσαίας κατηγορίας FPGA αποτελείται από 30.000 Logic Blocks. Η εικόνα 7.1 δείχνει το αποτύπωμα της μνήμης καθ όλη την εκτέλεση του εργαλείου VPR όταν απεικονίζεται το BGM benchmark. Γίνεται σαφές από αυτό το παράδειγμα ότι η χρήση της μνήμης μπορεί εύκολα να εμποδίσει την εκτέλεση καθώς επισκιάζει εύκολα τις αλγοριθμικές βελτιώσεις των εργαλείων.

Benchmark	**Mem Usage**	**Benchmark**	**Mem Usage**
arm_core	5.611 GB	mkDelayWorker32B	5.381 GB
bgm	6.019 GB	mkPktMerge	5.237 GB
blob_merge	5.368 GB	mkSMAdapter4B	5.251 GB
boundtop	5.281 GB	or1200	5.291 GB
ch_intrinsics	5.214 GB	raygentop	5.258 GB
diffeq1	5.225 GB	sha	5.270GB
diffeq2	5.214 GB	stereovision0	5.484 GB
LU8PEEng	5.907 GB	stereovision1	5.498 GB
stereovision2	6.006 GB		

Πίνακας 7.1: Αποτύπωμα Μνήμης του VPR 7.0 όταν απεικονίζει εφαρμογές σε ένα μεσαίου μεγέθους FPGA.

7.1.3 Συνεισφορά

Καθ ’όλη την παρούσα ερευνητική εργασία προτείνουμε μια νέα μεθοδολογία και το κατάλληλο toolflow για την εκτέλεση δυναμικής απεικόνισης πολλαπλών εφαρμογών πάνω σε ένα FPGA. Μια ετερογενής πλατφόρμα FPGA θεωρείται ένα σύνολο πόρων υλικού, συμπεριλαμβανομένων λογικών μπλοκ, μνήμης και DSP μπλοκ, όπου οι εφαρμογές μπορούν να απεικονιστούν ως δυναμικοί πυρήνες υλικού πάνω σε αυτούς τους πόρους. Κάθε ένας από αυτούς τους εικονικούς δυναμικούς πυρήνες, που ονομάζονται VKernels, μπορούν να υλοποιήσουν μόνο μία εφαρμογή, ενώ πολλαπλοί πυρήνες μπορούν να απεικονιστούν σε ένα ενιαίο FPGA. Προκειμένου να καταστεί εφικτό να απεικονιστούν αυτοί οι VKernels κατά το χρόνο εκτέλεσης, ακόμη και σε ενσωματωμένες πλατφόρμες έχει μειωθεί σημαντικά τόσο ο χρόνος εκτέλεσης όσο και το αποτύπωμα μνήμης, συνδυάζοντας επιθετικές βελτιστοποιήσεις στην χρήση μνήμης ταυτόχρονα με προσαρμοσμένους κατανεμητές μνήμης.

Τα βασικά χαρακτηριστικά της προτεινόμενης μεθοδολογίας, που ονομάζεται Het-JITPR, και του λογισμικού πλαισίου συνοψίζονται ως εξής:

– απεικόνιση πολλαπλών εικονικών πυρήνων υλικού (VKernels) σε μία ενιαία σύγχρονη FPGA αρχιτεκτονική που αποτελείται από προσαρμοστικά μπλοκ λογικής, μη ομοιόμορφη δρομολόγηση, DSP και μπλοκ μνήμης RAM.

– Οι απαιτήσεις επεξεργασίας έχουν μειωθεί σημαντικά τόσο μέσω αλγοριθμικών βελτιστοποιήσεων όσο και βελτιστοποιήσεων στην μνήμη.

– Απεικόνιση των δυναμικών πυρήνων στο FPGA, μέσω επιθετικού P&R, ακόμη και κατά τον χρόνο εκτέλεσης.

Το υπόλοιπο αυτού του κεφαλαίου είναι οργανωμένο ως εξής. Η ενότητα 2 παρουσιάζει τις σχετικές ερευνητικές εργασίες. Η ενότητα 3 παρέχει μια επισκόπηση της μεθοδολογίας για την αποτελεσματική απεικόνιση πυρήνων για FPGAs. Η ενότητα 4 παρέχει τις λεπτομέρειες των βελτιστοποιήσεων στα προτεινόμενα εργαλεία . Τα πειραματικά αποτελέσματα αναλύονται στην ενότητα 5. Τέλος, τα συμπεράσματα συνοψίζονται στην ενότητα 6.

7.2 Προτεινόμενη μεθοδολογία

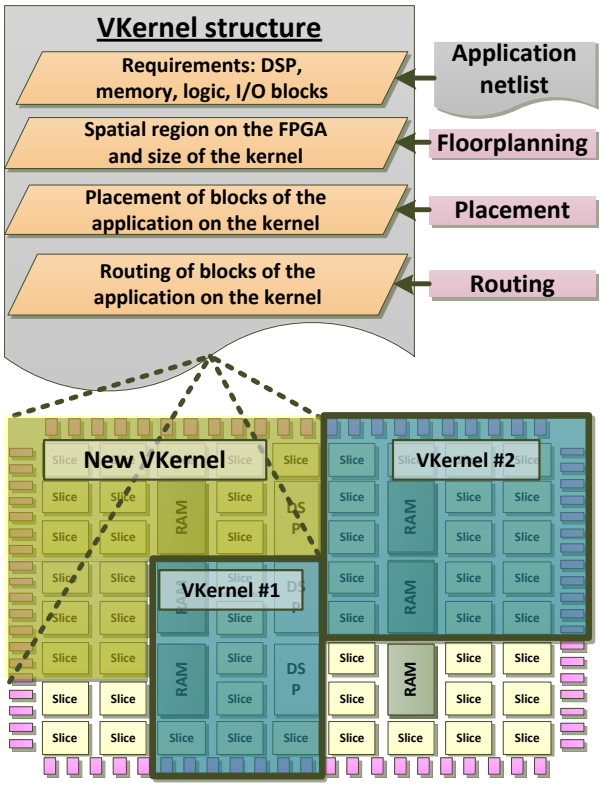
7.2.1 Δομή VKernel

Το επίκεντρο της προσέγγισής μας για τη αποτελεσματική απεικόνιση πολλαπλών πυρήνων υλικού είναι η χρήση των VKernel (Virtual πυρήνες). Για κάθε ξεχωριστή, ανεξάρτητη εφαρμογή που πρέπει να απεικονιστεί στο FPGA δημιουργείται ένα στιγμιότυπο ενός VKernel. Πολλαπλοί πυρήνες μπορούν να αναπτυχθούν στο FPGA την ίδια χρονική στιγμή. Αυτοί οι εικονικοί πυρήνες ενεργούν ως wrapper που ενσωματώνει τόσο την εφαρμογή όσο και το FPGA fabric που η εφαρμογή θα απεικονιστεί. Από την πλευρά της εφαρμογής, ο VKernel περιέχει πληροφορίες σχετικά με τις απαιτήσεις σε πόρους, περιορισμούς, και, τέλος, πώς η εφαρμογή τοποθετείται και δρομολογείται πάνω στον VKernel. Από την πλευρά της αρχιτεκτονικής FPGA ο VKernel χρησιμεύει ως ένα μικρότερο, ανεξάρτητο εικονικό FPGA, που συνδέεται άμεσα με τη φυσικούς reconfigurable πόρους.

Μια αφηρημένη αναπαράσταση της δομής τους, καθώς και τα βήματα, προκειμένου να απεικονιστεί μια εφαρμογή σε ένα VKernel, παρουσιάζονται στο Σχήμα 7.2. Το Σχήμα 7.3 αναπαριστά ποια βήματα της μεθοδολογίας μας αντιπροσωπεύονται από κάθε εργαλείο του προτεινόμενου πλαισίου, που ονομάζεται Het-JITPR.

7.2.2 Floorplanning / VKernel-Planner

Τα αρχικά βήματα της προτεινόμενης μεθοδολογίας είναι να σκιαγραφηθούν τα χαρακτηριστικά της εφαρμογής και το floorplanning. Κάθε φορά που μια νέα εφαρμογή πρέπει να απεικονιστεί πάνω στην αρχιτεκτονική, μια έκδοση της netlist της τροφοδοτείται ως είσοδος στα εργαλεία. Το πρώτο βήμα αφορά τον προσδιορισμό των πόρων



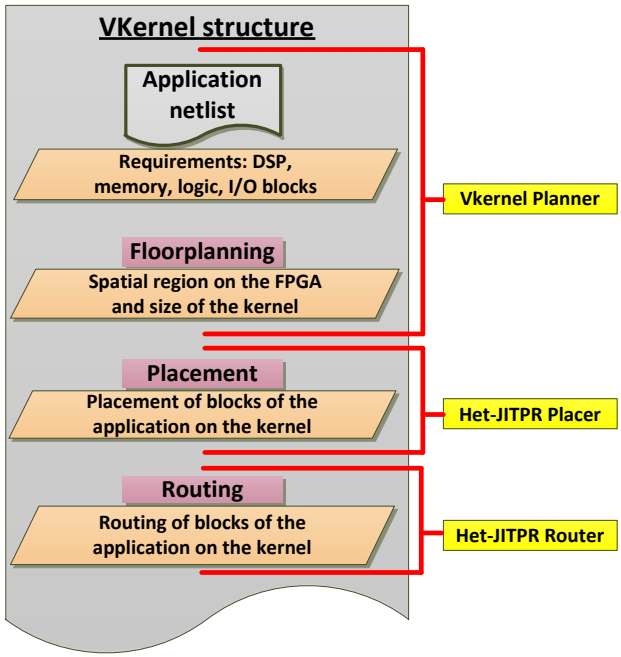
Σχήμα 7.2: Η δομή ενός VKernel και τα βήματα για να απεικονιστεί μια εφαρμογή πάνω σε αυτόν.

που απαιτούνται από την εφαρμογή, όπως λογικά μπλοκ, I/O μπλοκ, μνήμη και πυρήνες DSP. Μετά εκτελείται ένα βήμα floorplanning με στόχο να προσδιοριστεί η πλέον κατάλληλη περιοχή πάνω στο FPGA, όπου θα εφαρμοστεί ο νέος VKernel.

Και τα δύο αυτά βήματα υλοποιούνται από το προτεινόμενο εργαλείο VKernel-Planner. Αυτό το εργαλείο υπολογίζει αρχικά τους πόρους που απαιτούνται για έναν νέο VKernel και δημιουργεί ένα στιγμιότυπο αυτού. Στη συνέχεια, βρίσκει την πιο κατάλληλη χωρική τοποθεσία πάνω στην συσκευή, όπου ο νέος πυρήνας θα ανατεθεί. Δεδομένου ότι ο στόχος είναι η δυναμική απεικόνιση της εφαρμογής κατά το χρόνο εκτέλεσης, η επιλογή της χωρικής θέσης κάθε VKernel είναι το πιο κρίσιμο βήμα στη μεθοδολογία μας.

Ο αλγόριθμος που χρησιμοποιείται για στον VKernel-Planner έχει δύο στόχους: (i) την επίτευξη αποδοτικής αξιοποίησης των πόρων για κάθε VKernel, δεδομένου ότι πρέπει να απεικονίσουμε πολλαπλές εφαρμογές πάνω στο FPGA και (ii) την μείωση του χρόνου εκτέλεσης για το P&R των εφαρμογών, πάνω στην περιοχή του VKernel. Σε αυτό το βήμα τα εργαλεία είναι ενήμερα για τους κατειλημμένους πόρους του FPGA από προηγούμενους πυρήνες. Η περιοχή που θα επιλεχθεί, θα έχει τα ακόλουθα χαρακτηριστικά:

– Θα περιέχει αρκετούς ελεύθερους πόρους για τη νέα εφαρμογή/VKernel.



Σχήμα 7.3: Τα βήματα που εκτελούνται στο προτεινόμενο Het-JITPR framework και το λογισμικό που τα υλοποιεί.

- Μπορεί να έχει ακανόνιστο σχήμα, επιτρέποντας έτσι την καλύτερη αξιοποίηση FPGA.
- Θα είναι μια συνεχόμενη έκταση, επιτρέποντας έτσι μια αποτελεσματική απεικόνιση εφαρμογών να λάβει χώρα.
- Θα χρησιμοποιήσει ένα μικρό σύνολο των πόρων, προκειμένου να αποφευχθεί η εισαγωγή περιορισμών στην απεικόνιση μελλοντικών εφαρμογών/πυρήνων.

Όλες αυτές οι πληροφορίες είναι συσσωρευμένες στη δομή VKernel και έτσι τώρα ο νέος πυρήνας περιγράφεται από τον αριθμό των πόρων που θα καταλάβει (σύμφωνα με τις απαιτήσεις της εφαρμογής) και από την χωρική του θέση στην υποκείμενη FPGA πλατφόρμα (σύμφωνα πάντα με τους διαθέσιμους φυσικούς πόρους).

7.2.3 Τοποθέτηση και δρομολόγηση / Het-JITPR placer, router

Η netlist της εφαρμογής στην συνέχεια τοποθετείται και δρομολογείται (P&R) αυστηρά πάνω στους πόρους που διατίθενται από τον πυρήνα VKernel. Στα δύο αυτά βήματα τα εργαλεία αγνοούν το υπόλοιπο FPGA, έξω από την περιοχή του VKernel.

Ο προτεινόμενος placer βασίζεται σε ένα γρήγορο αλγόριθμο simulated annealing παρόμοιο με τον αλγόριθμο placer του toolflow του VTR [14], που υποστηρίζει επιπρόσθετα hard μπλοκ. Η αρχική τοποθέτηση μπλοκ βελτιστοποιείται από την εναλλαγή ζεύγους

μπλοκ, προκειμένου να βρεθεί μια τοποθέτηση που εμφανίζει χαμηλότερο συνολικό κόστος. Οι αλγόριθμοι annealing δέχονται ένα ποσοστό από κινήσεις που οδηγούν σε υψηλότερου κόστους λύσεις προκειμένου να αποφύγουν να παγιδευτούν σε τοπικά βέλτιστες λύσεις. Ο χρόνος εκτέλεσης του placer έχει βελτιωθεί σημαντικά με τη μείωση του αριθμού κινήσεων σε σχέση με τον αρχικό αλγόριθμο VTR όπως έχει ερευνηθεί στο παρελθόν στο [37]. Επιπλέον, δεδομένου ότι τοποθετούμε την εφαρμογή σε ένα VKernel που είναι ειδικά προσαρμοσμένο για τις ανάγκες αυτής της εφαρμογής, μειώνεται σημαντικά ο χώρος λύσεων της τοποθέτησης, επιτυγχάνοντας έτσι ακόμη ταχύτερους χρόνους εκτέλεσης.

Μετά το επιτυχές P&R οι κατειλημμένοι πόροι του πυρήνα VKernel συνδέονται άμεσα με τους αντίστοιχους πόρους του FPGA. Σε αυτό το βήμα, όλες οι απαραίτητες πληροφορίες για τον υπολογισμό του αρχείου bitstream για τη νέα εφαρμογή είναι διαθέσιμες.

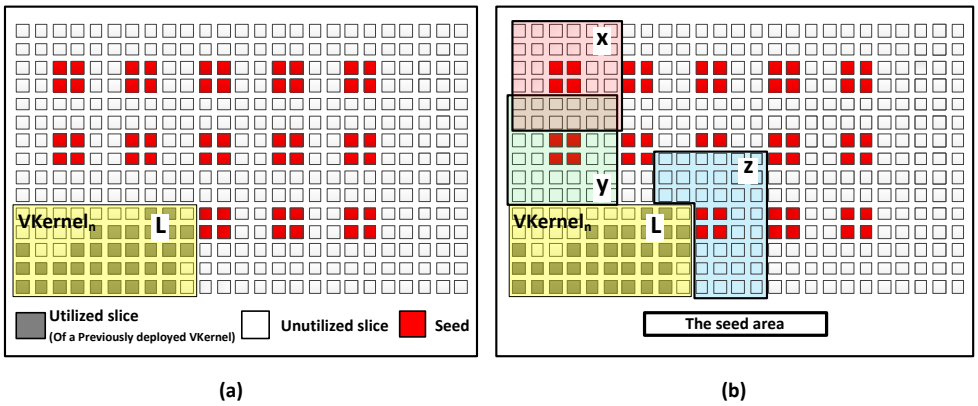
7.2.4 Διαγραφή των VKernels

Κάθε φορά που μια εφαρμογή πρέπει να εκφορτωθεί από την επαναδιαμορφούμενη συσκευή, οι πληροφορίες σχετικά με τους πόρους της ενημερώνονται κατάλληλα. Στη συνέχεια, τα αντίστοιχα slices μπορούν να διαμορφωθούν ως "κενά" αν υπάρχει ανάγκη από έναν καινούριο VKernel. Διαφορετικές προσεγγίσεις μπορούν να χρησιμοποιηθούν για την υλοποίηση αυτού του χαρακτηριστικού (π.χ. Προγραμματισμός αυτών των slices με ένα άδειο αρχείο bitstream), ενώ η επιλεγμένη προσέγγιση πρέπει να λαμβάνει υπόψη εγγενείς περιορισμούς που τίθενται από την υποκείμενη συσκευή FPGA.

7.3 Αλγοριθμικές βελτιώσεις και βελτιστοποιήσεις μνήμης

Ο κύριος στόχος αυτής της ερευνητικής εργασίας είναι να επιτρέψει τη απεικόνιση εφαρμογής σε FPGAs, όχι στατικά στην ώρα του σχεδιασμού, αλλά δυναμικά στον χρόνο εκτέλεσης, στην πλατφόρμα του τελικού χρήστη. Δεδομένου ότι η πλατφόρμα του τελικού χρήστη μπορεί να είναι ακόμη και ένα ενσωματωμένο σύστημα, μειώθηκε σημαντικά ο χρόνος εκτέλεσης και οι υπολογιστικοί πόροι του προτεινόμενου EDA εργαλείου ροής.

Ένα σημαντικό ποσοστό της μείωσης του χρόνου εκτέλεσης και του αποτυπώματος της μνήμης αποδίδεται στο νέο βήμα floorplanning που υλοποιείται από το προτεινόμενο εργαλείο VKernel-Planner. Λόγω της επιλογής μιας υποπεριοχής στο FPGA για κάθε VKernel εμείς επιθετικά μειώσαμε τόσο το χώρο αναζήτησης των P&R βημάτων, ενώ, επίσης, μειώθηκε η μνήμη που χρησιμοποιείται για την αναπαράσταση του FPGA κατά την εκτέλεση.



Σχήμα 7.4: Παράδειγμα της εφαρμογής του εργαλείου RegionFinder: (α) διανομή των seeds και (β) επέκταση των seeds στις αντίστοιχες περιοχές τους.

Προκειμένου να βελτιωθεί περαιτέρω τόσο ο χρόνος εκτέλεσης όσο και το αποτύπωμα μνήμης των εργαλείων απεικόνισης προτείνουμε επιπλέον βελτιστοποιήσεις στην διαχείριση μνήμης και έναν προσαρμοσμένο δυναμικό memory allocator. Οι βελτιστοποιήσεις έχουν στόχο τόσο την εκπροσώπηση της netlist της εφαρμογής όσο και της αρχιτεκτονικής του FPGA και είναι ανεξάρτητες από τους αλγόριθμους που χρησιμοποιούνται για την τοποθέτηση και δρομολόγηση.

7.3.1 VKernel-Planner

Το εργαλείο VKernel-Planner βασίζεται σε μια γρήγορη version ενός flood-fill αλγόριθμου που βρίσκει όχι μόνο τακτικές περιοχές (με ορθογώνιο ή τετράγωνο σχήμα), αλλά, κάθε περιοχή με ακανόνιστο σχήμα που αποτελείται από συνεχόμενους ανεκμετάλλευτους πόρους.

Η λειτουργικότητα του εργαλείου VKernel-Planner μπορεί να περιγραφεί ως εξής: Αρχικά, ο αλγόριθμος αναθέτει έναν αριθμό ομοιόμορφα κατανεμημένων seeds στο FPGA, όπως αυτές φαίνονται με κόκκινα κουτιά στο Σχήμα 7.4(a). Κάθε seed αντιπροσωπεύει μια πιθανή περιοχή για τον νέο VKernel. Ο αριθμός των seeds, το μέγεθός τους, καθώς επίσης και η απόσταση μεταξύ δύο συνεχόμενων seeds, υπολογίζονται κατά το χρόνο εκτέλεσης, δεδομένου ότι η επιλογή τους επηρεάζεται από τη διαθεσιμότητα των πόρων υλικού πάνω στην αρχιτεκτονική και τις απαιτήσεις των εφαρμογών (περισσότεροι seeds οδηγούν σε υψηλότερες επιδόσεις, αλλά με αυξημένο υπολογιστικό κόστος).

Κάθε ένας από τους seeds επεκτείνεται προς x και y κατευθύνσεις επανειλημμένα, με μια γρήγορη παραλλαγή του αλγορίθμου flood-fill μέχρι οι seeds να περιλαμβάνουν επαρκείς πόρους υλικού για τις ανάγκες του VKernel. Οι περιοχές που εκπροσωπούνται από κάθε seed μπορεί να είναι επικαλυπτόμενες μεταξύ τους, αλλά δεν μπορούν να επικαλύπτονται με τις περιοχές που έχουν ανατεθεί σε άλλο προηγούμενο VKernel. Αυτό παρουσιάζεται από ένα απλουστευμένο παράδειγμα στο Σχήμα 7.4(b), όπου οι

περιοχές X, Y και Z είναι πιθανές λύσεις για το νέο VKernel και η περιοχή L, έχει ανατεθεί σε προηγούμενο VKernel.

Το κόστος της λύσης του κάθε seed αξιολογείται από τον VKernel-Planner και έχει τρεις συνιστώσες, C_α, C_β και C_γ, που σχετίζονται με την αποτελεσματικότητα της κατανομής των πόρων.

- Η υπερβολική χρήση των πόρων εκπροσωπείται από την συνιστώσα C_α και επηρεάζει το συνολικό αριθμό των πυρήνων που μπορούν να απεικονιστούν πάνω στο FPGA.
- Η C_β συνιστώσα αντιπροσωπεύει το πόσο κοντά είναι οι πυρήνες που έχουν απεικονιστεί πάνω στο FPGA, και επηρεάζει τον κατακερματισμό των πόρων.
- Η κανονικότητα της υποψήφιας περιοχής εκπροσωπείται στο συστατικό C_γ και επηρεάζει την ποιότητα της απεικόνισης της εφαρμογής.

Είναι σημαντικό να αναφερθεί ότι ο όρος $Bounding\ Box$, που περιλαμβάνεται στις περισσότερες από τις ακόλουθες εξισώσεις είναι το ελάχιστο ορθογώνιο που περιέχει ένα δεδομένο σύνολο μπλοκ. Ορίζεται ως $[x_{min}, y_{min}] - [x_{max}, y_{max}]$ από τις συντεταγμένες του κάθε μπλοκ στο δεδομένο σύνολο.

Κάθε συνιστώσα του κόστους υπολογίζεται από τις ακόλουθες εξισώσεις:

1. Συνιστώσα α (Eq.7.1):

$$C_\alpha = \sum_{i=1}^{n} k_i * \frac{R_{req\ i}}{R_{free\ i}} \qquad with \sum_{i=1}^{n} k_i = 1 \tag{7.1}$$

$R_{req\ i}$: *is the number of resource type i blocks required by the VKernel*

$R_{free\ i}$: *is the number of free resource type i blocks inside the seed's area*

k_i : *is a weight factor representing how critical is resource type i*

Κάθε υποψήφια περιοχή μπορεί να περιλαμβάνει περισσότερους πόρους από αυτούς που πραγματικά απαιτείται από τον VKernel-Planner. Αυτό συμβαίνει κυρίως λόγω του περιορισμού για συνεχόμενες περιοχές. Σε ένα FPGA υπάρχουν διάφοροι τύποι πόρων όπως Logic μπλοκ, I/O pads, μπλοκ DSP, κλπ, το καθένα σε διαφορετικούς αριθμούς και χωρικές θέσεις στο εσωτερικό του FPGA. Με το C_α τιμωρείται το ποσό των επιπλέον πόρων, μέσω του $R_{req\ i}/R_{free\ i}$ και το k_i είναι ένα κανονικοποιημένο βάρος ανάλογα με το είδος των πόρων i. Τα επιπλέον DSP και τα μπλοκ μνήμης τιμωρούνται με τον υψηλότερο συντελεστή k, δεδομένου ότι ο αριθμός τους είναι περιορισμένος και η χωρική τοποθέτηση τους πάνω στη συσκευή FPGA είναι σταθερή. Πιο μικρές ποινές έχουν τα μη χρησιμοποιούμενα μπλοκ I/O, δεδομένου ότι επίσης έχουν σταθερές θέσεις συνήθως στην περιφέρεια του FPGA και τις χαμηλότερες ποινές τα λογικών μπλοκ. Δεδομένου ότι το

κόστος αυτό τιμωρεί την κατάχρηση πόρων, επηρεάζει άμεσα τον αριθμό των μελλοντικών πυρήνων που μπορούν να απεικονιστούν. Για να παρουσιαστεί καλύτερα αυτό το κόστος, το σχήμα 7.5(a), περιγράφει ένα παράδειγμα των δύο υποψήφιων περιοχών για το VKernel όπου $C_{\alpha\ I} > C_{\alpha\ II}$.

2. Συνιστώσα β (Eq.7.2):

$$C_\beta = 1 - \frac{Free\ blocks\ at\ Utilized\ Bounding\ Box}{Total\ blocks\ at\ Utilized\ Bounding\ Box} \tag{7.2}$$

Utilized Bounding Box ονομάζεται το Bounding Box που περιέχει την υποψήφια περιοχή για το νέο VKernel και κάθε ήδη απεικονισμένο VKernel. Το C_β αντιπροσωπεύει το πόσο κοντά είναι το νέο VKernel στα προηγούμενα VKernels. Αυτό επηρεάζει την ποιότητα της απεικόνισης για μελλοντικές εφαρμογές, δεδομένου ότι εισάγει υψηλό κατακερματισμό στους πόρους των μελλοντικών VKernel. Για να παρουσιαστεί καλύτερα αυτό το κόστος, στο σχήμα 7.5(b), παρουσιάζεται ένα παράδειγμα δύο υποψήφιων περιοχών με τα αντίστοιχά τους Utilized Bounding Boxes όπου $C_{\beta\ I} > C_{\beta\ II}$ και η γκρίζα ζώνη είναι δεσμευμένη από το ήδη υπάρχοντα VKernels.

3. Συνιστώσα γ (Eq.7.3):

$$C_\gamma = \left(1 - \frac{|nx - ny|}{nx + ny}\right) + \left(\frac{Free\ blocks\ at\ Seed\ Bounding\ Box}{Total\ blocks\ at\ Seed\ Bounding\ Box}\right) \tag{7.3}$$

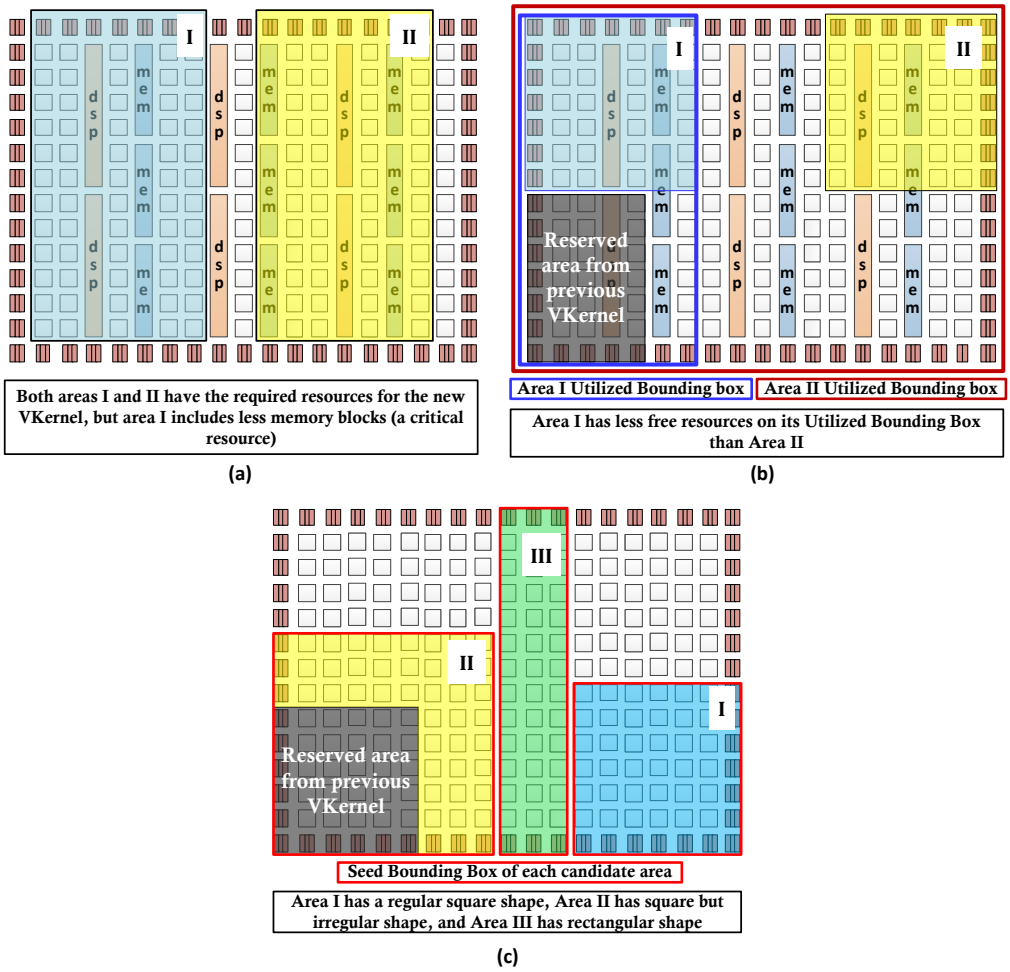
nx : *the width of the seed's area*

ny : *the height of the seed's area*

Seed Bounding Box ονομάζεται το Bounding Box που περιέχει τα στοιχεία της περιοχής του seed. Το C_γ αντιπροσωπεύει την καθετότητα $(1 - |nx - ny|/(nx + ny))$ και την κανονικότητα $(\frac{Free\ blocks\ at\ Seed\ Bounding\ Box}{Total\ blocks\ at\ Seed\ Bounding\ Box})$ της περιοχής του seed. Πολύ ακανόνιστα σχήματα επηρεάζουν με αρνητικό τρόπο την ποιότητα της P&R λύσης της εφαρμογής, έτσι προτιμάται η περιοχή που προορίζεται για ένα VKernel να είναι τακτική. Αυτή η συνιστώσα απεικονίζεται στο σχήμα 7.5(c) όπου $C_{\gamma\ I} > C_{\gamma\ II}$, $C_{\gamma\ I} > C_{\gamma\ III}$ και η γκρίζα ζώνη είναι δεσμευμένη από το ήδη υπάρχοντα VKernels.

Μετά τον υπολογισμό των C_α, C_β και C_γ για κάθε seed ο VKernel-Planner κρατά τις λύσεις (seeds) που σχηματίζουν το μέτωπο Pareto στο τρισδιάστατο χώρο που δημιουργείται από αυτές τις συνιστώσες του κόστους. Από το σύνολο αυτών των seeds ο VKernel-Planner επιλέγει πια πτυχή του κόστους θέλει να μεγιστοποιήσει, μέσω παραμέτρων που καθορίζονται από τον χρήστη.

Ο χρόνος εκτέλεσης γενικά του VMKernel-Planner είναι αμελητέος δεδομένου ότι η πολυπλοκότητα του είναι $O(n \times \sqrt{\frac{n}{m}})$, όπου n υποδηλώνει τον αριθμό των slices που βρέθηκαν στην αρχιτεκτονική και m είναι ο αριθμός των slices που απαιτούνται για την υλοποίηση της εφαρμογής.



Σχήμα 7.5: Παράδειγμα από τα τρία στοιχεία κόστους που χρησιμοποιούνται για την αξιολόγηση των λύσεων για VMKernel-Planner. (a)C_α Eq 7.1 where $C_{\alpha\ I} > C_{\alpha\ II}$, (b) C_β Eq 7.2 όπου $C_{\beta\ I} > C_{\beta\ II}$, (c) C_γ Eq 7.3 όπου $C_{\gamma\ I} > C_{\gamma\ II}$ και $C_{\gamma\ I} > C_{\gamma\ III}$.

7.3.2 Προτεινόμενες βελτιστοποιήσεις μνήμης

Οι εφαρμογές και οι πλατφόρμες FPGA σήμερα ξεπερνούν τα 400.000 στοιχεία λογικής και έτσι η χρήση μνήμης από τα εργαλεία EDA έχει γίνει ένας σημαντικός περιορισμός, τόσο όσο αναφορά τον χρόνο εκτέλεσης όσο και στις απαιτήσεις μνήμης. Η χρήση της μνήμης επηρεάζει το χρόνο εκτέλεσης με διάφορους τρόπους. Το πιο βασικό πρόβλημα είναι ότι οι λειτουργίες μνήμης (Load/Store) παίρνουν πολύ περισσότερο χρόνο σε σύγκριση με άλλες εντολές CPU. Το δεύτερο σημαντικότερο πρόβλημα είναι οι μεταφορές μνήμης μεταξύ cache, RAM και σκληρού δίσκου, όπου κάθε επίπεδο είναι μία τάξη μεγέθους πιο αργό από την προηγούμενο. Αυτό επιδεινώνεται από το γεγονός ότι ιδιαίτερα ακανόνιστες δυναμικές δομές, όπως γράφοι και σωροί χρησιμοποιούνται συχνά από τα εργαλεία EDA.

Οι προτεινόμενες βελτιστοποιήσεις μνήμης αφορούν τόσο την αρχιτεκτονική FPGA όσο και την netlist της εφαρμογής που θα απεικονιστεί. Για το υπόλοιπο της ερευνητικής εργασίας οι βελτιστοποιήσεις που ασχολούνται με την netlist της εφαρμογής είναι οι *Level 1* βελτιστοποιήσεις και αυτές που ασχολούνται με την αρχιτεκτονική του FPGA είναι οι *Level 2* βελτιστοποιήσεις.

7.3.2.1 *Level 1* βελτιστοποιήσεις

Η είσοδος του χρήστη στο toolflow είναι η εφαρμογή που θέλουμε να απεικονίσουμε πάνω στο FPGA σε .net, netlist αρχείο. Αυτό το αρχείο αντιπροσωπεύει ιεραρχικά το κύκλωμα της εφαρμογής, εκφράζοντας την σε πολύπλοκα μπλοκ που σχηματίζουν την αρχιτεκτονική FPGA, όπως Logic Blocks, I/O pads, μνήμη και μονάδες DSP. Για παράδειγμα, ένα Block Logic στο αρχείο .net θα περιέχει το όνομα του μπλοκ, όλες τις εισόδους και εξόδους, τις συνδέσεις και την εσωτερική αρχιτεκτονική σχετικά με το πώς τα LUTs, τα latches, τα flip-flops, κλπ διασυνδέονται στο μπλοκ. Φυσικά αυτή η λογική Block πρέπει να αντιστοιχεί στην αρχιτεκτονική του FPGA, όσον αφορά τον αριθμό των I/O pins, τον αριθμό των LUTs, latches, και τους τρόπους διασύνδεσης.

Στην σκιαγράφηση της χρήσης μνήμης του VPR 7,0 ανακαλύφθηκε ότι ένα σημαντικό μέρος της μνήμης δαπανάται για την εσωτερική αναπαράσταση αυτών των πολύπλοκων μπλοκ. Ένας ανεξάρτητος γράφος δρομολόγησης περιγράφει αυτή τη διασύνδεση σε κάθε μπλοκ. Η χρήση μνήμης σε σύγκριση με τον αριθμό μπλοκ του κάθε benchmark φαίνεται στον Πίνακα 7.2.

Όταν ένα πολύπλοκο μπλοκ διαβάζεται από την netlist, η εσωτερική του διασύνδεση επεξεργάζεται και ο αντίστοιχος γράφος δρομολόγησης δημιουργείται. Αυτός ο γράφος παραμένει στην μνήμη καθ όλη την διάρκεια της εκτέλεσης του P&R. Στις *Level 1* βελτιστοποιήσεις προστέθηκε ένα βήμα μετά την επεξεργασία της netlist και μετά την δημιουργία του γράφου. Σε αυτό το βήμα αποθηκεύονται σε ένα global διάνυσμα μόνο οι πληροφορίες σχετικά με τα pins του μπλοκ, οι συνδέσεις τους, εάν έχουν, ή η ισοδυναμία τους, αλλά όχι οι εσωτερικές διασυνδέσεις. Μετά από αυτό το βήμα απελευθερώνεται ο γράφος πριν από την ανάγνωση του επόμενου πολύπλοκου μπλοκ. Με την τροποποίηση αυτή επιτυγχάνουμε δύο πράγματα:

1. Η μνήμη που χρησιμοποιείται από το διάνυσμα είναι πολύ μικρότερη από αυτή που χρησιμοποιείται από τον γράφο δεδομένου ότι δεν περιλαμβάνονται οι εσωτερικές συνδέσεις.
2. Με τη χρήση ενός διανύσματος επιτυγχάνουμε καλύτερη χωρική συσχέτιση μνήμης από ότι με τον γράφο.

Benchmark	**Total # of blocks**	**Mem used for netlist representation**
arm_core	1469	228.67 MB
bgm	3107	491.86 MB
blob_merge	678	94.59 MB
boundtop	700	41.82 MB
ch_intrinsics	266	9.23 MB
diffeq1	300	10.61 MB
diffeq2	193	8.75 MB
LU8PEEng	2296	393.10 MB
mkDelayWorker32B	1549	114.49 MB
mkPktMerge	497	19.42 MB
mkSMAdapter4B	570	33.43 MB
or1200	1039	47.61 MB
raygentop	722	32.68 MB
sha	284	36.68 MB
stereovision0	1261	158.44 MB
stereovision1	1223	166.87 MB
stereovision2	2932	464.18 MB

Πίνακας 7.2: Μνήμη που χρησιμοποιείται για την αναπαράσταση της netlist κάθε benchmark στο VPR.

Παρά το γεγονός ότι υπάρχει μια αποτελεσματική μείωση στο αποτύπωμα μνήμης, αυτή η λύση έχει ένα μειονέκτημα. Ορισμένες σύγχρονες προσεγγίσεις για την τοποθέτηση και δρομολόγηση μερικές φορές αλλάζουν την εσωτερική απεικόνιση των μπλοκ σε μια προσπάθεια για περαιτέρω βελτιστοποίηση της ποιότητας της απεικόνισης. Αυτό αυξάνει το χρόνο εκτέλεσης λόγω της επέκτασης του χώρου αναζήτησης. Με τις *Level 1* βελτιστοποιήσεις απενεργοποιούμε αυτήν την επιλογή, δεδομένου έχουμε απορρίψει τις εσωτερικές πληροφορίες σύνδεσης μπλοκ για τη διάρκεια της τοποθέτησης και δρομολόγησης. Από την άλλη πλευρά ο στόχος είναι η γρήγορη απεικόνιση που οδηγεί σε μια αρκετά καλή λύση, οπότε η επιλογή αυτή καθίσταται άνευ σημασίας.

7.3.2.2 *Level 2* βελτιστοποιήσεις

Κατά τη διάρκεια της φάσης δρομολόγησης τα μπλοκ του κυκλώματος διασυνδέονται μέσω των πόρων του FPGA. Για το λόγο αυτό ανεξάρτητα από τον αλγόριθμο δρομολόγησης που θα χρησιμοποιηθεί, τα εργαλεία EDA χρειάζονται την εκπροσώπηση όλων των πιθανών μονοπατιών διασύνδεσης. Ο πιο συνηθισμένος τρόπος είναι να χρησιμοποιηθεί ένας γράφος πόρων δρομολόγησης (rr γράφος), που έχει ως κόμβους κάθε pin και κάθε κομμάτι υποδομής της δρομολόγησης FPGA, που οδηγεί σε εκατοντάδες χιλιάδες κόμβους.

Συγκεκριμένα στο VPR 7.0 ένα μεγάλο ποσοστό του χρόνου εκτέλεσης δαπανάται χτίζοντας τον rr γράφο. Το ποσοστό αυτό έχει μειωθεί σημαντικά χρησιμοποιώντας το

προτεινόμενο εργαλείο VKernel-Planner, λόγω κατάλληλης επιλογής μιας υποπεριοχής για τον VKernel. Επίσης, το μεγαλύτερο ποσοστό της χρήσης μνήμης στο VPR 7.0 παρατηρείται κατά την κατασκευή του γράφου rr.

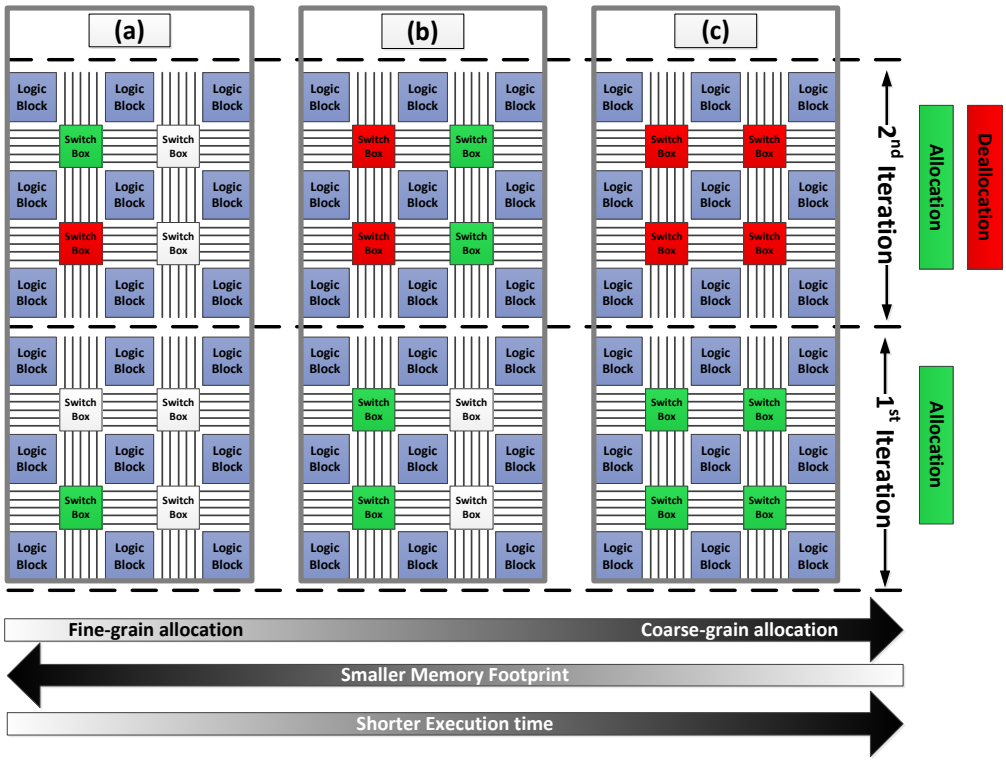
Κάθε switchbox σε μία reconfigurable αρχιτεκτονική συνδέει τα οριζόντια και τα κάθετα κανάλια στην διασταύρωση τους. Ανάλογα με την αρχιτεκτονική switchbox κάθε καλώδιο από κάθε πλευρά (επάνω, κάτω, αριστερά, δεξιά) μπορεί να συνδεθεί σε $3 \times N$ καλώδια όπου είναι το πλάτος του καναλιού. Το VPR πρώτα χτίζει ένα μεγάλο πίνακα που περιέχει όλες τις συνδέσεις switchbox προκειμένου να καθορίσει αργότερα τις άκρες των καλωδίων που περνούν μέσα από κάθε switchbox. Αυτό γίνεται πιο πολύπλοκο όταν έχουμε μη ομοιόμορφη αρχιτεκτονική δρομολόγησης, όπου τα καλώδια μπορούν να έχουν μεταβλητό μήκος. Για παράδειγμα, σε ένα FPGA με μέγεθος array 200×200 (σύγχρονο FPGA μεσαίου μεγέθους) και πλάτος καναλιού 200 το μέγεθος της μνήμης αυτού του πίνακα είναι 1,775.4 MB ή 1,73 GB.

Προκειμένου να μειωθεί το αποτύπωμα μνήμης, άλλαξε η κατανομή αυτού του πίνακα από στατική σε δυναμική. Οι πληροφορίες αυτές απαιτούνται κάθε φορά που ένα οριζόντιο ή κατακόρυφο κανάλι χτίζεται (στο γράφο rr) στη θέση x, y. Για να ελαχιστοποιηθεί η κατανομή μνήμης, φορτώνεται και υπολογίζεται ο πίνακας σε κάθε χ στήλη και στη συνέχεια γίνεται deallocated. Αυτό δίνει το καλύτερο trade-off μεταξύ χρόνου εκτέλεσης και μέγεθους αποτυπώματος μνήμης. Το σχήμα 7.6 δείχνει τρία σενάρια κατανομής από fine-grain σε coarse grain, όπου παρουσιάζονται σε δύο επαναλήψεις η κατανομή και η ανακατανομή των switchboxes. Κατά τη μετακίνηση προς μια coarse grain λύση ο χρόνος εκτέλεσης μειώνεται και το αποτύπωμα της μνήμης αυξάνεται. Η προτεινόμενη λύση φαίνεται στο Σχήμα 7.6(b) ενώ η λύση του VPR είναι η 7.6(c).

7.3.2.3 Καθολικές βελτιστοποιήσεις μνήμης

Προκειμένου να αξιολογηθούν με ακρίβεια οι λύσεις τοποθέτησης, το VPR πριν από την τοποθέτηση κατασκευάζει έναν πίνακα που συσχετίζει τις καθυστερήσεις δρομολόγησης με την απόσταση των μπλοκ. Για το λόγο αυτό, ο δρομολογητής χτίζει τον γράφο rr του FPGA και όλες τις διαδρομές μπλοκ που έχουν διαφορετικές αποστάσεις μεταξύ τους. Αυτό το βήμα έχει αντικατασταθεί με ένα κωδικοποιημένο αρχείο σε μορφή bit που περιέχει αυτές τις πληροφορίες. Το μέγεθος αυτού του αρχείου είναι μερικά kBytes, ακόμη και για μεγάλα FPGAs. Ο προτεινόμενος placer διαβάζει αυτό το αρχείο και δημιουργεί τον πίνακα χωρίς την ανάγκη για rr γράφο και δρομολόγηση.

Τέλος η μείωση του αποτυπώματος μνήμης έχει επιτευχθεί από την αναδιάρθρωση και την αλλαγή των εσωτερικών τύπων στις διαφορετικές δομές γράφου.



Σχήμα 7.6: Ένα απλουστευμένο παράδειγμα των διαφορετικών αναθέσεων μνήμης switchbox για τις 2 πρώτες επαναλήψεις, (a) η ανάθεση finest-grain, (b) η προτεινόμενη λύση, που δίνει ένα καλό trade-off μεταξύ του χρόνου εκτέλεσης και του αποτυπώματος μνήμης, (c) η ταχύτερη αλλά πιο ακριβή σε μνήμη λύση του VPR.

7.3.3 Προτεινόμενος δυναμικός κατανεμητής μνήμης

Έχουμε αναπτύξει την προτεινόμενη λύση βασιζόμενοι σε δύο σύγχρονους δυναμικούς κατανεμητές μνήμης, τον jemalloc [92] και τον Lockless [93]. Ο Jemalloc είναι η προεπιλεγμένη επιλογή στο FreeBSD και στον Firefox, ενώ ο Lockless είναι ένας κατανεμητής που χρησιμοποιεί πολλές βελτιστοποιήσεις που υποστηρίζουν τα σύγχρονα λειτουργικά συστήματα. Και οι δύο εστιάζουν στην παροχή μίας lock-free στρατηγικής κατά τη διάρκεια της ταυτόχρονης δυναμικής διαχείρισης μνήμης στην ίδια εφαρμογή. Αυτό επιτυγχάνεται με την ανάθεση περιοχών μνήμης για κάθε πυρήνα CPU. Αυτές οι περιοχές της μνήμης είναι ανεξάρτητες η μία από την άλλη και ο συγχρονισμός αποφεύγεται δεδομένου ότι κάθε νήμα που ζητά allocation ή deallocation υποβάλλει το αίτημα του σε διαφορετική περιοχή μνήμης.

Ένα πλεονέκτημα των δύο αυτών κατανεμητών προέρχεται από το γεγονός ότι ελαχιστοποιούν την επιστροφή μνήμης στο σύστημα. Όταν η εφαρμογή απελευθερώνει τμήματα μνήμης, ο επιλεγμένος κατανεμητής μνήμης είναι πιο απρόθυμος να τα επιστρέψει στο σύστημα σε σχέση με τον προεπιλεγμένο κατανεμητή στα περισσότερα συστήματα Linux, τον glibc. Αν το σύστημα ξεμείνει από μνήμη, η εικονική μνήμη του

πυρήνα του Linux είναι αρκετά έξυπνη για να αναθέσει σε περιοχή swap διευθύνσεις μνήμης διεργασιών που δεν εκτελούνται εκείνη την στιγμή.

Οι κρυφές μνήμες σε κάθε νήμα διαδραματίζουν σημαντικό ρόλο στον jemalloc. Ο κύριος στόχος τους είναι να μειωθεί ακόμη περισσότερο ο αριθμός των γεγονότων συγχρονισμού. Κάθε νήμα διατηρεί ένα χώρο προσωρινής αποθήκευσης των αντικειμένων έως ένα ορισμένο μέγεθος (32 KiB συνήθως). Κατά τη διάρκεια μιας εφαρμογής allocation και πριν το νήμα προσπελάσει τις περιοχές μνήμης, ελέγχει πρώτα για ένα αποθηκευμένο διαθέσιμο αντικείμενο. Allocation μέσω μιας κρυφής μνήμης του νήματος δεν απαιτεί κανένα απολύτως κλείδωμα. Όπως έχει παρατηρηθεί, κάποια μικρά μεγέθη είναι αρκετά δημοφιλή και ως εκ τούτου μπορούμε να βελτιστοποιήσουμε τον σχεδιασμό του κατανεμητή ως προς αυτά τα μεγέθη για να μεγιστοποιήσουμε την ταχύτητα του. Σε κάθε περίπτωση, οι κρυφές μνήμες του κάθε νήματος θα πρέπει να προσφέρουν ταχύτητα χωρίς να επιβαρύνουν τον κατακερματισμό της μνήμης.

Ο Lockless έχει μια διαφορετική προσέγγιση για την διαχείριση των μικρών μεγεθών, χρησιμοποιώντας έναν slab κατανεμητή. Slabs είναι περιοχές μνήμη των 64 KiB συν 128 bytes για τη διατήρηση των απαραίτητων μεταδεδομένων. Υπάρχουν πολλά slabs, ένα για κάθε μέγεθος έως και 512 bytes με βήμα των 16 bytes. Ένα πιθανό μειονέκτημα σε σχέση με τον jemalloc είναι ότι τα slabs αποτελούν αντικείμενο διαπραγμάτευσης μεταξύ παράλληλων νημάτων και ενδέχεται να υπόκεινται σε γενικό συγχρονισμό. Ωστόσο, τόσο Het-JITPR όσο και το VPR χρησιμοποιεί ένα ενιαίο νήμα, οπότε δεν υπάρχει ιδιαίτερη επιβράδυνση από τη χρήση του κατανεμητή slab.

7.4 Πειραματικά αποτελέσματα

Αυτή η ενότητα παρέχει μια σειρά από πειραματικά αποτελέσματα και οι συγκρίσεις που αποδεικνύουν η αποδοτικότητα της προτεινόμενης λύσης. Για το σκοπό αυτό, χρησιμοποιούμε 17 κοινά hardware benchmarks. Ένας αριθμός από αυτά κάνουν χρήση των πολλαπλασιαστών και των μνημών και τα υπόλοιπα απαιτούν μόνο λογικά μπλοκ, προκειμένου να υλοποιηθούν σε FPGA. Ο πίνακας 7.3 συνοψίζει τα χαρακτηριστικά αυτών των benchmark.

Για πειραματισμό, χρησιμοποιήθηκε μια πλατφόρμα FPGA που αποτελείται από μία συστοιχία 200×200 λογικών μπλοκ, καθώς και μια σειρά από ενσωματωμένους πολλαπλασιαστές και μνημες, όπως στην Altera Stratix IV αρχιτεκτονική. Η απεικόνιση των εφαρμογών πραγματοποιήθηκε με το προτεινόμενο toolflow που εκτελείται σε έναν διακομιστή με Intel Xeon επεξεργαστή και 64GB μνήμης RAM. Αυτό το σύστημα επιλέχθηκε επειδή η συγκρινόμενη επιλογή, το VPR 7.0, P&R εργαλείο του VTR [14], δεν μπορούσε να τρέξει σε ένα ενσωματωμένο σύστημα λόγω του μεγάλου χρόνου εκτέλεσης και της μεγάλης ποσότητας μνήμης που απαιτείται όπως θα καταστεί εμφανές και από το παρόν τμήμα της εργασίας.

Πίνακας 7.3: Χαρακτηριστικά των benchmark.

Benchmark	LUTs	Inputs	Outputs	Memories	Multipiers
arm_core	13697	133	179	40	0
bgm	30782	257	32	0	11
blob_merge	6018	36	100	0	0
boundtop	3037	275	192	1	0
ch_intrinsics	425	99	130	1	0
diffeq1	485	162	96	0	5
diffeq2	322	66	96	0	5
LU8PEEng	21739	114	102	45	8
mkDelayWorker32B	5631	511	553	43	0
mkPktMerge	228	311	156	15	0
mkSMAdapter4B	1977	195	205	5	0
or1200	3053	385	394	2	1
raygentop	2147	239	305	1	7
sha	2277	38	36	0	0
stereovision0	11472	157	197	0	0
stereovision1	10287	133	145	0	38
stereovision2	29768	149	182	0	213

7.4.1 Ποιότητα απεικόνισης ενός κυκλώματος στο FPGA

Εδώ παρουσιάζεται μία ποσοτικοποίηση της απόδοσης της παραγόμενης λύσης όσον αφορά τη μέγιστη συχνότητα λειτουργίας. Τα αποτελέσματα αυτής της ανάλυσης για τις δύο εναλλακτικές ροές συνοψίζονται στον Πίνακα 7.4. Με βάση τον πίνακα αυτό, το προτεινόμενο πλαίσιο έχει ένα κέρδος στην μέγιστη συχνότητα λειτουργίας σε σύγκριση με το VPR, 1.26 $\times$ κατά μέσο όρο. Αν και μειώθηκε σημαντικά ο χρόνος εκτέλεσης, η ποιότητα της λύσης μας είναι καλύτερη από το toolflow benchmark (VPR). Αυτό συμβαίνει επειδή οι εφαρμογές απεικονίζονται σε προσαρμοσμένους VKernels μειώνοντας σημαντικά τον χώρο λύσεων της τοποθέτησης.

Για την καλύτερη κατανόηση του παραπάνω σημείου δίνονται τρία παραδείγματα από τον πίνακα 7.4: ch_intrinsics, mkPktMerge και LU8PEEng. Ο placer χρησιμοποιεί έναν simulated annealing αλγόριθμο όπου μειώθηκε σημαντικά ο αριθμός των τυχαίων κινήσεων σε σύγκριση με το VPR [14] εργαλείο. Η αρχιτεκτονική FPGA που χρησιμοποιείται αποτελείται από περίπου 40.000 slices του κάθε τύπου. Με το προτεινόμενο toolflow που χρησιμοποιεί VKernels αριθμός των slices είναι ειδικά προσαρμοσμένος σε κάθε εφαρμογή. Έτσι έχουμε:

ch_intrinsics: Στο VPR [14], η τοποθέτηση αξιολογεί συνολικά 225.986 λύσεις, ενώ κάθε μπλοκ έχει 40.000 πιθανές θέσεις. Στον VKernel ο placer αξιολογεί 33.782 λύσεις, αλλά κάθε μπλοκ έχει 465 πιθανές θέσεις.

mkPktMerge: Στο VPR [14], η τοποθέτηση αξιολογεί συνολικά 523.852 λύσεις, ενώ κάθε μπλοκ έχει 40.000 πιθανές θέσεις. Στον VKernel ο placer αξιολογεί 64.610 λύσεις, αλλά κάθε μπλοκ έχει 1.180 πιθανές θέσεις.

LU8PEEng: Στο VPR [14], η τοποθέτηση αξιολογεί συνολικά 3.908.545 λύσεις, ενώ κάθε μπλοκ έχει 40.000 πιθανές θέσεις. Στον VKernel ο placer αξιολογεί 298.480 λύσεις, αλλά κάθε μπλοκ έχει 3.080 πιθανές θέσεις.

Αυτό εξηγεί επίσης γιατί στα τρία μεγαλύτερα benchmark (BGM, stereovision2 και LU8PEEng), βλέπουμε μια μικρή μείωση στη συχνότητα λειτουργίας σε αντίθεση με τα περισσότερα benchmark. Καθώς οι εφαρμογές αυξάνουν σε μέγεθος, προσεγγίζοντας το μέγεθος του FPGA τα κέρδη που προσφέρονται από τον VKernel μειώνονται.

Πίνακας 7.4: Σύγκριση μεταξύ του προτεινόμενου toolflow Het-JITPR, και του VTR κατά την απεικόνιση ενός benchmark, όσον αφορά τη μέγιστη συχνότητα λειτουργίας.

Benchmark	**Max Op. Freq (MHz)**		**Gain**
	VPR [14]	**Het-JITPR**	
arm_core	52.716	53.215	1.009×
bgm	39.938	36.514	0.914×
blob_merge	90.913	89.820	0.988×
boundtop	124.247	146.075	1.176×
ch_intrinsics	118.986	256.546	2.156×
diffeq1	40.905	45.932	1.123×
diffeq2	50.841	57.959	1.140×
LU8PEEng	8.5423	8.4703	0.992×
mkDelayWorker32B	95.567	128.85	1.348×
mkPktMerge	141.55	220.384	1.557×
mkSMAdapter4B	113.544	162.929	1.435×
or1200	59.423	66.332	1.116×
raygentop	107.758	192.735	1.789×
sha	70.7102	72.062	1.019×
stereovision0	172.764	218.32	1.264×
stereovision1	99.393	154.015	1.550×
stereovision2	57.356	53.450	0.932×
Average	**-**	**-**	**1.265×**

7.4.2 Ανάλυση του χρόνου εκτέλεσης

Ο χρόνος εκτέλεσης είναι μία κρίσιμη μετρική στην απεικόνιση μιας εφαρμογής, στην πλατφόρμα του τελικού χρήστη. Για κάθε βελτιστοποίηση που προτείνεται, χρησιμοποιείται το εργαλείο VPR για σύγκριση καθώς είναι ευρέως αποδεκτό στον ακαδημαϊκό χώρο και αρκετά ευέλικτο ώστε να υποστηρίζει hard blocks όπως μνήμες και πολλαπλασιαστές. Δεδομένου ότι το VPR δεν υποστηρίζει πολλαπλά benchmark, η σύγκριση από άποψη χρόνου εκτέλεσης γίνεται για τη απεικόνιση ενός benchmark.

Ο πίνακας 7.5 δείχνει τα οφέλη που επιτυγχάνονται στον χρόνο εκτέλεσης από τις βελτιστοποιήσεις στην μνήμη στο VPR. Αν και η τελική εφαρμογή του προτεινόμενου πλαισίου χρησιμοποιεί τη δική της εκδοχή (Het-JITPR) του εργαλείου P&R δείχνεται εδώ η επίδραση της προτεινόμενης βελτιστοποίησης στο αυθεντικό VPR.

Στην *Mem opts Level 1* στήλη έχουμε ενεργοποιημένες μόνο βελτιστοποιήσεις στο επίπεδο της εφαρμογής και στην *Mem opts Level 1 &2* στήλη έχουμε ενεργοποιήσει και βελτιστοποιήσεις στο επίπεδο αρχιτεκτονικής. Ο κύριος λόγος για την επιτάχυνση και στα δύο σενάρια είναι: i) οι καθολικές βελτιστοποιήσεις που μειώνουν το αρχικό επεξεργαστικό χρόνο και ii) η μείωση του αποτυπώματος της μνήμης που οδηγεί σε γρηγορότερους χρόνους εύρεσης λύσεων.

Στο σενάριο όπου οι βελτιστοποιήσεις στο επίπεδο της αρχιτεκτονικής είναι επίσης ενεργοποιημένες, βλέπουμε μια αύξηση του χρόνου εκτέλεσης. Αυτό είναι αναμενόμενο δεδομένου ότι χρησιμοποιείται μια finer grain ανάθεση μνήμης από το VPR και αυτό οδηγεί στην παρατηρούμενη ∼8% αύξηση κατά μέσο του χρόνου εκτέλεσης.

Πίνακας 7.5: Σύγκριση μεταξύ του αρχικού εργαλείου VPR και του τροποποιημένου VPR με Level 1 και Level 1&2 βελτιστοποιήσεις μνήμης κατά τη απεικόνιση μίας εφαρμογής, όσον αφορά τον χρόνο εκτέλεσης σε δευτερόλεπτα.

Benchmark	**Execution time (sec)**				
	VPR	**Mem opts Level 1**	**Gain**	**Mem opts Level 1&2**	**Gain**
arm_core	176.03	98.60	56.01%	111.00	63.06%
bgm	216.18	148.85	68.85%	163.51	75.64%
blob_merge	146.81	70.40	47.95%	83.28	56.73%
boundtop	143.41	66.74	46.54%	79.57	55.48%
ch_intrinsics	140.02	63.33	45.23%	76.10	54.35%
diffeq1	141.62	64.71	45.69%	77.44	54.68%
diffeq2	140.79	64.05	45.49%	76.75	54.51%
LU8PEEng	196.81	131.84	66.99%	140.54	71.41%
mkDelayWorker32B	159.81	79.93	50.02%	92.66	57.98%
mkPktMerge	142.28	68.25	47.97%	80.73	56.74%
mkSMAdapter4B	143.37	66.95	46.70%	79.02	55.12%
or1200	144.98	69.01	47.60%	81.51	56.22%
raygentop	142.94	66.69	46.66%	79.29	55.47%
sha	141.78	65.21	45.99%	78.00	55.01%
stereovision0	150.59	74.64	49.57%	87.29	57.97%
stereovision1	158.96	82.54	51.93%	95.24	59.91%
stereovision2	229.27	163.67	71.39%	175.18	76.41%
Average	**-**	**-**	**51.80%**	**-**	**59.81%**

Στον Πίνακα 7.6 παρουσιάζουμε τα αποτελέσματα διαφόρων κατανεμητών μνήμης στο προτεινόμενο Het-JITPR. Σε αυτόν τον πίνακα έχουμε ενεργοποιημένες όλες τις βελτιστοποιήσεις της μνήμης εκτός από τις αλγοριθμικές βελτιστοποιήσεις. Να σημειωθεί ότι ο χρόνος εκτέλεσης είναι $\sim 16\times$ γρηγορότερος σε σύγκριση με το αρχικό VPR.

Από τον παραπάνω πίνακα μπορούμε να δούμε ότι το αποτέλεσμα των κατανεμητών μνήμης είναι σταθερό κατά μέσο όρο, προσφέροντας μια μείωση του χρόνου εκτέλεσης, περίπου 5% για στον jemalloc και 10% στον Lockless. Ο λόγος πίσω από αυτό είναι διττός: (α) η στρατηγική allocation ταιριάζει καλύτερα στα μοτίβα χρήσης της μνήμης του toolflow Het-JITPR και (β) οι κατανεμητές παρέχουν απλούστερους και ως εκ τούτου ταχύτερους μηχανισμούς κλειδώματος από την malloc της glibc. Όταν τα μεγέθη που ζητούνται για κατανομή μνήμης είναι περιορισμένα και χωρίς μεγάλες διακυμάνσεις, οι κατανεμητές αποδίδουν πολύ καλύτερα από ό,τι η προεπιλεγμένη glibc. Επιπλέον, οι δύο αυτοί κατανεμητές προσπαθούν να βελτιώσουν την ταχύτητα εκτέλεσης τους μετακινώντας μερικές από τις εργασίες που πρέπει να γίνουν, στο βήμα του deallocation. Μερικές φορές δεν αποδεσμεύουν καν την μνήμη που χρησιμοποιήθηκε από την εφαρμογή. Όλες αυτές οι στρατηγικές, καθώς και το απλό pattern της μνήμη της εφαρμογής οδηγούν σε μια ταχύτερη εκτέλεση όταν χρησιμοποιούνται αυτοί οι προσαρμοσμένοι κατανεμητές.

Πίνακας 7.6: Σύγκριση μεταξύ διαφορετικών κατανεμητών στο εργαλείο Het-JITPR, με βελτιστοποιήσεις μνήμης Level 1&2 από την άποψη του χρόνου εκτέλεσης σε δευτερόλεπτα.

Benchmark	**Execution time (sec)**				
	Het-JITPR with Mem opts 1&2				
	glibc-malloc	**jemalloc**	**Gain**	**Lockless**	**Gain**
arm_core	21.54	19.67	91.32%	19.06	88.49%
bgm	38.6	34.18	88.55%	32.82	85.03%
blob_merge	5.02	4.57	91.04%	4.26	84.86%
boundtop	3.05	2.95	96.72%	2.74	89.84%
ch_intrinsics	1.2	1.17	97.50%	1.12	93.33%
diffeq1	1.45	1.4	96.55%	1.35	93.10%
diffeq2	1.11	1.09	98.20%	1.07	96.40%
LU8PEEng	33.91	31.36	92.48%	30.65	90.39%
mkDelayWorker32B	14.59	13.87	95.07%	13.1	89.79%
mkPktMerge	3.1	3.06	98.71%	2.89	93.23%
mkSMAdapter4B	2.44	2.34	95.90%	2.18	89.34%
or1200	5.83	5.65	96.91%	5.25	90.05%
raygentop	3.44	3.32	96.51%	3.09	89.83%
sha	2.05	1.97	96.10%	1.84	89.76%
stereovision0	10.86	9.9	91.16%	9.44	86.92%
stereovision1	12.69	11.59	91.33%	11.19	88.18%
stereovision2	53.18	48.66	91.50%	46.99	88.36%
Average	**-**	**94.44%**	**-**	**89.82%**	**-**

7.4.3 Αποτύπωμα μνήμης

Η χρήση της μνήμης και του αποτυπώματος είναι ένα υπολογιστικό εμπόδιο για τα FPGA CAD εργαλεία. Προκειμένου να απεικονιστούν οι εφαρμογές σε FPGAs κατά το

χρόνο εκτέλεσης στην πλατφόρμα του τελικού χρήστη, το αποτύπωμα πρέπει να μειωθεί σημαντικά.

Στον Πίνακα 7.7 παρουσιάζουμε τα κέρδη που επιτυγχάνονται με τις προτεινόμενες βελτιστοποιήσεις μνήμης στον αρχικό VPR [14] κώδικα. Μπορούμε να δούμε ότι με Level 1 βελτιστοποιήσεις (σε επίπεδο εφαρμογής) επετεύχθη μείωση περίπου 10% στο μέγιστο αποτύπωμα μνήμης και επιπλέον 30% με το Level 2 (επίπεδο αρχιτεκτονικής) συνολικού ύψους 40% μείωση κατά μέσο όρο.

Είναι προφανές ότι, παρά τη μεγάλη διακύμανση στα μεγέθη benchmark εμείς δεν παρατηρούμε τέτοιες διακυμάνσεις στις μειώσεις της μνήμης. Αυτό επιβεβαιώνει τα αποτελέσματα της σκιαγράφησης που δείξανε ότι η αρχιτεκτονική του FPGA είναι ο κύριος ένοχος για την κατανάλωση μνήμης στην διαδικασία της απεικόνισης μιας εφαρμογής. Αυτό εξηγεί επίσης γιατί με τις Level 2 βελτιστοποιήσεις της μνήμης, μπορούμε να επιτύχουμε $3\times$ μείωση σε σύγκριση με τις Level 1 βελτιστοποιήσεις.

Όλες οι βελτιστοποιήσεις της μνήμης έχουν αναπτυχθεί προσεκτικά, έτσι ώστε να μην επηρεάζεται η ποιότητα της τελικής λύσης, που σημαίνει ότι η εφαρμογή τοποθετείται και δρομολογείται πάνω στο FPGA με ακριβώς τον ίδιο τρόπο.

Πίνακας 7.7: Σύγκριση μεταξύ του αρχικού VPR και του τροποποιημένου VPR με Level 1 και Level 1&2 βελτιστοποιήσεις μνήμης κατά τη απεικόνιση ενός benchmark, από την άποψη του αποτυπώματος μνήμης σε MB.

Benchmark	**Memory footprint (MB)**				
	VPR	**Mem opts Level 1**	**Gain**	**Mem opts Level 1&2**	**Gain**
arm_core	5746	5139	89.45%	3417	59.47%
bgm	6163	5236	84.95%	3509	56.94%
blob_merge	5497	5078	92.38%	3361	61.14%
boundtop	5408	5061	93.58%	3337	61.71%
ch_intrinsics	5339	5045	94.50%	3325	62.27%
diffeq1	5350	5046	94.32%	3327	62.18%
diffeq2	5339	5031	94.23%	3327	62.31%
LU8PEEng	6049	5189	85.78%	3468	57.34%
mkDelayWorker32B	5510	5083	92.25%	3372	61.20%
mkPktMerge	5363	5048	94.14%	3330	62.10%
mkSMAdapter4B	5377	5057	94.04%	3338	62.08%
or1200	5418	5063	93.44%	3343	61.71%
raygentop	5384	5058	93.93%	3340	62.04%
sha	5396	5049	93.57%	3338	61.86%
stereovision0	5616	5122	91.21%	3402	60.58%
stereovision1	5630	5129	91.11%	3410	60.57%
stereovision2	6150	5267	85.65%	3534	57.46%
Average	**-**	**-**	**91.68%**	**-**	**60.76%**

Στον Πίνακα 7.8 παρουσιάζουμε τα αποτελέσματα των διαφόρων κατανεμητών μνήμης στο προτεινόμενο Het-JITPR σε σχέση με το αποτύπωμα μνήμης, όπου έχουμε ενεργοποιήσει όλες τις βελτιστοποιήσεις μνήμης εκτός από τις αλγοριθμικές βελτιστοποιήσεις. Σε σύγκριση με το αρχικό VPR, το Het-JITPR επιτυγχάνει $\sim 12\times$ μείωση στην χρήση μνήμης.

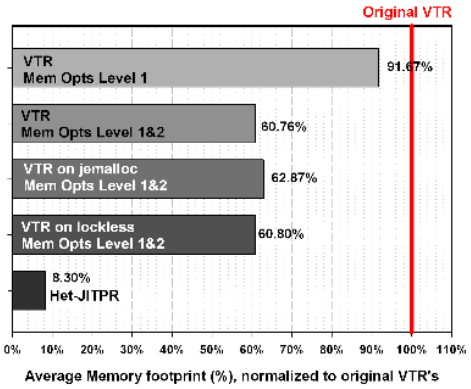
Μπορούμε να παρατηρήσουμε ότι και οι δύο, jemalloc και Lockless, κατανεμητές σε γενικές γραμμές, οδηγούν σε μία αύξηση του αποτυπώματος μνήμης σε σύγκριση με τον κατανεμητή μνήμης της glibc. Στον πίνακα 7.8 το αποτύπωμα της μνήμης έχει αυξηθεί 12% και 27% κατά μέσο όρο αντίστοιχα. Και οι δύο κατανεμητές είναι πιο επιθετικοί από ό,τι ο αρχικός και δεν επιστρέφουν πάντα την deallocated μνήμη στο σύστημα. Αντ αυτού προτιμούν να κρατούν την διαχείρισή της για μελλοντικά allocations που θα μπορούσε να ζητήσει το Het-JITPR. Αυτό εξαλείφει την πιθανότητα μιας κλήσης συστήματος για επιπλέον μνήμη σε μια μελλοντική εφαρμογή, αλλά με κόστος τον επιπλέον κατακερματισμό της μνήμης κατά την εκτέλεση της εφαρμογής. Ακόμα και αν τα αποτελέσματα αυτά παρουσιάζουν αύξηση στο αποτύπωμα μνήμης σε σύγκριση με εκείνα που χρησιμοποιούν την glibc-malloc, το αποτύπωμα μνήμης είναι πάντα μικρότερο σε σχέση με το αρχικό VPR, χωρίς Level 1&2 βελτιστοποιήσεις της μνήμης.

Πίνακας 7.8: Σύγκριση μεταξύ διαφορετικών κατανεμητών στο Het-JITPR, με βελτιστοποιήσεις μνήμης επιπέδου 1&2 , σε σχέση με το αποτύπωμα μνήμης σε MB.

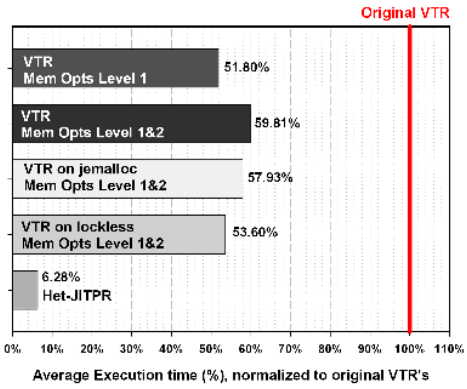
Benchmark	**Memory footprint (MB)**				
	Het-JITPR with Mem opts 1&2				
	glibc-malloc	**jemalloc**	**Gain**	**Lockless**	**Gain**
arm_core	278	319	114.92%	337	121.54%
bgm	569	633	111.28%	581	102.13%
blob_merge	151	148	98.30%	217	143.67%
boundtop	136	116	85.35%	200	147.03%
ch_intrinsics	122	133	108.84%	150	122.89%
diffeq1	122	131	106.84%	154	126.13%
diffeq2	121	127	104.42%	129	106.16%
LU8PEEng	459	492	107.04%	466	101.37%
mkDelayWorker32B	301	401	133.13%	405	134.51%
mkPktMerge	125	133	106.13%	177	141.75%
mkSMAdapter4B	132	151	114.32%	168	127.19%
or1200	168	235	140.28%	250	149.01%
raygentop	133	155	115.86%	208	155.94%
sha	132	151	114.14%	162	122.71%
stereovision0	198	262	132.42%	283	143.16%
stereovision1	225	251	111.26%	298	132.09%
stereovision2	824	824	99.95%	777	94.30%
Average	**-**	**-**	**112.03%**	**-**	**127.74%**

7.4.4 Σύνοψη

Τα σχήματα 7.7α , 7.7α , 7.7β και 7.7β συνοψίζουν τα οφέλη και τους συμβιβασμούς που επιτυγχάνονται με την προτεινόμενη λύση.



(α) Τα κέρδη στο αποτύπωμα μνήμης για το VPR με διάφορες επιλογές.



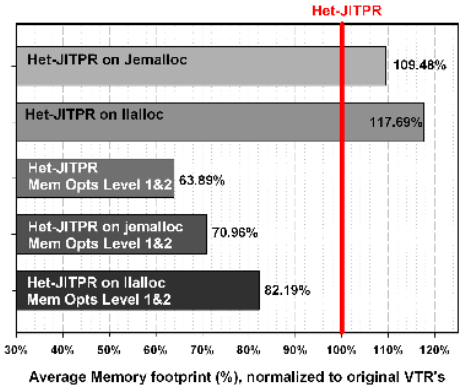
(β) Τα κέρδη σε χρόνο εκτέλεσης στο αρχικό VPR με διάφορες επιλογές.

Στην Εικόνα 7.7α παρουσιάζεται το αποτέλεσμα της βελτιστοποίησης μνήμης και των διαφορετικών κατανεμητών σχετικά με το αποτύπωμα μνήμης για το αρχικό VPR. Επιτυγχάνουμε από 10% έως 40% μείωση του αποτυπώματος μνήμης με τις βελτιστοποιήσεις της μνήμης, και τους προσαρμοσμένους κατανεμητές. Όταν προσθέσουμε και τις αλγοριθμικές βελτιστοποιήσεις (Het-JITPR) φτάνουμε μείωση ως και 90%.

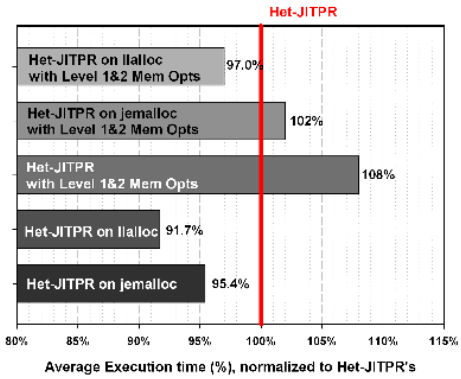
Στην Εικόνα 7.7β παρουσιάζεται η επίδραση της βελτιστοποίησης μνήμης και των διαφορετικών κατανεμητών από άποψη χρόνου εκτέλεσης στο αρχικό VPR. Ως αποτέλεσμα της βελτιστοποίησης της μνήμης και των κατανεμητών μνήμης βλέπουμε μια επιτάχυνση από 40% έως 50% στον χρόνο εκτέλεσης. Αν προσθέσουμε τις αλγοριθμικές βελτιστοποιήσεις με τη χρήση του VKernel, μπορούμε να επιτύχουμε μια επιτάχυνση της τάξης των 15× κατά μέσο όρο σε σύγκριση με το αρχικό VPR. Παρατηρούμε επίσης μια μικρή αύξηση που προκαλείται από τους προσαρμοσμένους κατανεμητές.

Η εικόνα 7.7α δείχνει το αποτέλεσμα των βελτιστοποιήσεων μνήμης, όταν οι αλγοριθμικές βελτιστοποιήσεις είναι επίσης ενεργοποιημένες. Εδώ βλέπουμε ότι δύο κατανεμητές έχουν ως αποτέλεσμα την αύξηση του αποτυπώματος μνήμης από ∼ 10% έως 20% σε σχέση με τον glibc κατανεμητή μνήμης. Επιπρόσθετα η βελτιστοποίηση μνήμης πέτυχε μείωση περίπου 40% στο αποτύπωμα μνήμης, παρόμοια με το αρχικό VPR, σε κάθε περίπτωση που ο κατανεμητής διατηρήθηκε ο ίδιος.

Η εικόνα 7.7β δείχνει πώς επηρεάζεται ο χρόνος εκτέλεσης, όταν επιτρέπονται οι βελτιστοποιήσεις στην μνήμη και οι κατανεμητές μνήμης μαζί με τη χρήση των VKernels. Όταν συγκριθεί αυτό, με την εικόνα 7.7α το trade-off μεταξύ του χρόνου εκτέλεσης και του αποτυπώματος μνήμης είναι εμφανές. Ενώ η βελτιστοποίηση μνήμης προσφέρει μια μείωση 36% στο Het-JITPR, οδηγεί σε ποινή 8% στο χρόνο εκτέλεσης. Η επίδραση



(α) Τα κέρδη στο αποτύπωμα μνήμης για το Het-JITPR με διάφορες επιλογές.



(β) Τα κέρδη σε χρόνο εκτέλεσης στο Het-JITPR με διάφορες επιλογές.

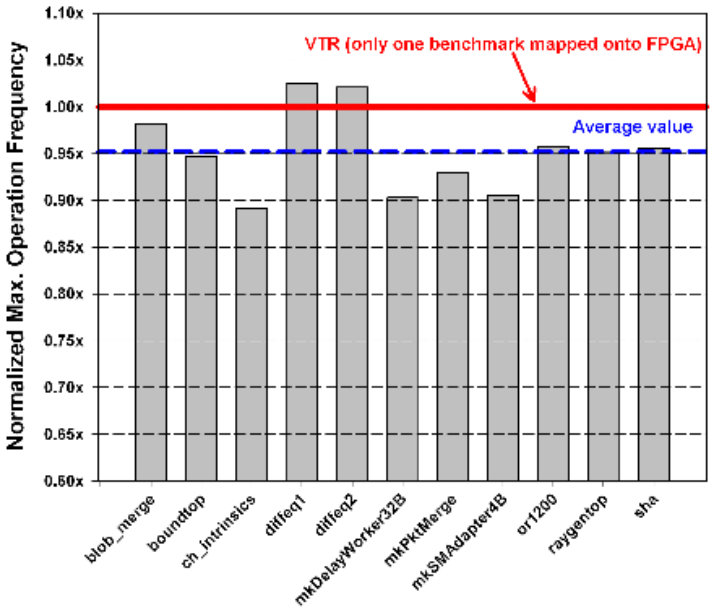
των προσαρμοσμένων κατανεμητών είναι η αντίθετη, προσφέροντας μια μείωση 5% έως 10% στον χρόνο εκτέλεσης αλλά ίση αύξηση στη χρήση της μνήμης.

Οι προαναφερθέντες συμβιβασμοί μπορούν να διερευνηθούν και μπορεί να αναπτυχθεί μια λύση που να είναι βελτιστοποιημένη για τους πόρους του συστήματος που στοχεύουμε.

7.4.5 Πολλαπλές εφαρμογές μέσω Virtual Kernels

Στα πειραματικά αποτελέσματα που παρουσιάζονται στα προηγούμενα εδάφια, τα benchmark έχουν απεικονιστεί πάνω στην αρχιτεκτονική ως stand-alone εφαρμογές (υποθέτοντας ότι η πλατφόρμα FPGA είναι άδεια). Ωστόσο, οι VKernels επιτρέπουν την υλοποίηση πολλών εφαρμογών σε ένα ενιαίο FPGA, ακόμη και αν άλλες εφαρμογές έχουν ήδη απεικονιστεί στη συσκευή. Για να ποσοτικοποιηθεί αυτό το χαρακτηριστικό, αξιολογήθηκε η αποτελεσματικότητα της προτεινόμενης λύσης στο να χειριστεί πολλαπλές εφαρμογές (δηλαδή μέσα από τη δυναμική εισαγωγή και αφαίρεση των εφαρμογών).

Για αυτό το πείραμα, η υποκείμενη αρχιτεκτονική είναι παρόμοια με τις προηγούμενες περιπτώσεις. Όσον αφορά την ακολουθία των εφαρμογών, δημιουργήθηκε μια ουρά που αποτελείται από 110 κυκλώματα από ένα υποσύνολο των προηγούμενων benchmark που έχουν απεικονιστεί δυναμικά πάνω στην αρχιτεκτονική. Επιλέχθηκαν benchmark έτσι ώστε κατά μέσο όρο να υπάρχουν πέντε με δέκα benchmark πάνω στην πλατφόρμα. Ο αριθμός των benchmark που συνυπάρχουν σε ένα FPGA επηρεάζεται άμεσα από τους πόρους του κάθε benchmark και το μέγεθος του FPGA. Σε περίπτωση που ένα νέο benchmark δεν χωράει στο FPGA αφαιρούνται πυρήνες που έχουν ήδη υλοποιηθεί μέχρι να υπάρχουν αρκετοί πόροι για το νέο VKernel. Πιο αποτελεσματικοί αλγόριθμοι scheduling, (όπως priority based scheduling) μπορούν να χρησιμοποιηθούν εύκολα από το προτεινόμενο toolflow, αλλά τέτοιοι αλγόριθμοι είναι έξω από το πεδίο εφαρμογής της παρούσας ερευνητικής εργασίας.



Σχήμα 7.7: Αξιολόγηση του προτεινόμενου πλαισίου, όταν πολλαπλές εφαρμογές απεικονίζονται ως πυρήνες στο FPGA, όσον αφορά τη μέγιστη συχνότητα λειτουργίας.

Η ευελιξία των ακανόνιστων συνεχόμενων περιοχών για την υλοποίηση VKernels, και οι πεπερασμένοι πόροι μιας πλατφόρμας FPGA επηρεάζουν τη μέγιστη συχνότητα λειτουργίας του κάθε κυκλώματος που απεικονίζεται. Κάθε φορά που δημιουργείται ένας VKernel, η αξιοποίηση των πόρων και τη χωρική κατανομή των διαθέσιμων πόρων έχουν σημαντικό ρόλο στην τελική απεικόνιση. Δεδομένου ότι υποστηρίζεται δυναμική απεικόνιση εφαρμογών, το προτεινόμενο toolflow δεν γνωρίζει εκ των προτέρων τα χαρακτηριστικά των μελλοντικών εφαρμογών.

Κάθε benchmark απεικονίστηκε αρκετές φορές και η μέση μέγιστη συχνότητα λειτουργίας παρουσιάζεται στο Σχήμα 7.7. Η αρχική λύση αντιστοιχεί στην απόδοση του κάθε benchmark όταν απεικονίζεται επάνω στο FPGA με τη χρήση του εργαλείου VPR ως αυτόνομη εφαρμογή. Κατά μέσο όρο η προτεινόμενη λύση έχει ποινή μόνο 4,7% σε σχέση με την αναφορά. Αυτό αποδεικνύει την αποτελεσματικότητα της λύσης, ακόμη και κάτω από περιορισμούς χρόνου εκτέλεσης.

7.5 Συμπεράσματα

Μια νέα μεθοδολογία για την υποστήριξη πολλαπλών εφαρμογών απεικόνισης σε ένα ενιαίο ετερογενές FPGA παρουσιάστηκε στο παρόν κεφάλαιο. Κάθε εφαρμογή μεταμορφώθηκε σε μια νέα δυναμική δομή, που ονομάζεται VKernel, που οδήγησε στην γρήγορη και αποτελεσματική υλοποίηση της εφαρμογής. Με βάση τα πειράματα, επιτυγχάνεται απεικόνιση εφαρμογής 30×, πιο γρήγορα κατά μέσο όρο, σε σύγκριση με την state-of-art

προσέγγιση, χωρίς να υποβαθμίζεται η μέγιστη συχνότητα λειτουργίας. Όταν πολλαπλές εφαρμογές απεικονίζονται πάνω στο FPGA στόχο, έχουν μια αμελητέα ποινή 4,7% όσον αφορά τη μέγιστη λειτουργία συχνότητα.

What... is the air-speed velocity of an unladen swallow?

Bridgekeeper

Κεφάλαιο 8

Συμπεράσματα

Το θέμα της παρούσας διδακτορικής διατριβής ήταν ο σχεδιασμός αρχιτεκτονικών και η απεικόνιση εφαρμογών σε επαναδιαμορφούμενες πλατφόρμες με εργαλεία λογισμικού. Ο στόχος αυτός μεταφράζεται σε ανάλυση των περιορισμών που δυσχεραίνουν την διαδικασία της απεικόνισης μιας εφαρμογής και την πρόταση νέων λύσεων που άρουν αυτούς τους περιορισμούς.

Η αποδοτικότητα στην διαδικασία της απεικόνισης ορίστηκε ως η επίτευξη υψηλότερης απόδοσης με ελάχιστη προσπάθεια από την άποψη του χρόνου σχεδιασμού και υπολογιστικών πόρων που χρησιμοποιούνται. Σε ό,τι αφορά τις προκλήσεις που αναφέρονται στο κεφαλαίο 2 οι παρακάτω λύσεις παρουσιάστηκαν στην παρούσα Διδακτορική Διατριβή.

Λύση I

Μια νέα μεθοδολογία, καθώς και το απαραίτητο λογισμικό πλαίσιο προτάθηκαν, για τη διευκόλυνση της εξερεύνησης σε επίπεδο αρχιτεκτονικής ετερογενών FPGAs. Αυτό επέτρεψε την αξιολόγηση των διαφορετικών ιεραρχιών μνήμης, και μια εξερεύνηση της βέλτιστης θέσης των μπλοκ μνήμης πάνω στο FPGA. Πειραματικά αποτελέσματα απέδειξαν την αποτελεσματικότητα της προτεινόμενης λύσης, αφού οι διαφορετικές ιεραρχίες μνήμης και αρχιτεκτονικά σχέδια διερευνήθηκαν με επιτυχία.

Λύση II

Μια νέα αρχιτεκτονική 3-D FPGA αναλύθηκε, όπου τα I/O και τα μπλοκ μνήμης ανατέθηκαν σε διαφορετικά στρώματα σε σύγκριση με τους πόρους λογικής. Αυτό το νέο αρχιτεκτονικό πρότυπο με ετερογενή 3D επίπεδα υποστηρίζεται από ένα λογισμικό πλαίσιο, που ονομάζεται 3-D NAROUTO. Κατά την διαδικασία της αξιολόγησης αποδείχθηκε η αποτελεσματικότητα της προτεινόμενης λύσης υλικού/λογισμικού, καθώς επετεύχθη απεικόνιση εφαρμογών με σημαντικά μικρότερα μήκη καλωδίων, το οποίο

με τη σειρά του οδήγησε σε βελτιώσεις στην καθυστέρηση και την κατανάλωση ισχύος. Η καθυστέρηση μειώθηκε 30% κατά μέσο όρο ενώ ταυτόχρονα παρατηρήθηκε μείωση 10% στην κατανάλωση ενέργειας, σε σύγκριση με τις συμβατικές 2-D αρχιτεκτονικές FPGA.

Λύση III

Ένα νέο πλαίσιο λογισμικού προτάθηκε το οποίο εκτελεί Just-In-Time (JIT) απεικόνιση εφαρμογών σε FPGA, που ονομάζεται JiTPR. Το πλαίσιο αυτό εκτελεί απεικόνιση εφαρμογών κατά το χρόνο εκτέλεσης και οδηγεί σε αμελητέα προβλήματα κατακερματισμού, σε σύγκριση με παρόμοιες προσεγγίσεις, οι οποίες ασχολούνται κυρίως με προυπολογισμένα αρχεία διαμόρφωσης. Αυτή η μεθοδολογία JiTPR υποστηρίζει μόνο αρχιτεκτονικές που αποτελούνται μόνο από CLBs και ομοιόμορφη δρομολόγηση. Μέσω πειραματισμού, η προτεινόμενη λύση επιτυγχάνει μια μέση επιτάχυνση στο P&R της εφαρμογής 53,5×, σε σύγκριση με το state-of-the-art εργαλείο VPR. Ακόμα κι αν κάποιος θα περίμενε ότι τα κέρδη αυτά έρχονται με ποινές στην απόδοση, το πλαίσιο επιτυγχάνει κατά μέσο όρο, 1,17× υψηλότερη συχνότητα λειτουργίας.

Λύση IV

Παρουσιάστηκε μια μεθοδολογία cloud η οποία υποστηρίζει πολλαπλές απεικονίσεις εφαρμογών κατά το χρόνο εκτέλεσης σε ένα ενιαίο ή περισσότερα FPGAs. Η μέθοδος αυτή υποστηρίζεται από ένα καινοτόμο πλαίσιο λογισμικού που θα μπορούσε να κλιμακωθεί για να φιλοξενήσει πολλαπλές αιτήσεις από εφαρμογές. Με βάση τα πειραματικά αποτελέσματα,το P&R των εφαρμογών έγινε 32× κατά μέσο όρο πιο γρήγορα σε σύγκριση με μια state-of-the-art λύση, χωρίς κυρώσεις στην μέγιστη συχνότητα λειτουργίας. Όταν πολλαπλά benchmark χρειάζεται να απεικονιστούν πάνω στο FPGA το πλαίσιο κλιμακώνεται για να υποστηρίξει έως και 73 εργασίες απεικόνισης ανά λεπτό ενώ εισάγει ποσοστό κατακερματισμού 24% στους πόρους του FPGA.

Λύση V

Μια νέα μεθοδολογία παρουσιάστηκε η οποία υποστηρίζει δυναμική απεικόνιση πολλαπλών εφαρμογών σε ένα ετερογενές FPGA. Κάθε εφαρμογή μεταμορφώθηκε σε μια δυναμική δομή, που ονομάζεται VKernel, και με τη χρήση της προτεινόμενης ροής Het-JITPR απεικονίζεται πάνω στο FPGA. Η μεθοδολογία αυτή, μαζί με βελτιστοποιήσεις στην μνήμη και προσαρμοσμένους κατανεμητές μνήμης, οδήγησε σε γρήγορη και αποτελεσματική απεικόνιση των εφαρμογών. Με βάση τα πειραματικά αποτελέσματα, η απεικόνιση εφαρμογών έγινε 15×, πιο γρήγορα κατά μέσο όρο, σε σύγκριση με μια προσέγγιση state-of-art, χωρίς να υποβαθμίζεται η μέγιστη συχνότητα λειτουργίας, χρησιμοποιώντας μόνο ένα μικρό μέρος, $1/12$, της μνήμης που χρησιμοποιείται σε ένα state-of-art εργαλείο για την εκτέλεση του προτεινόμενου toolflow. Αυτό ανοίγει τον δρόμο για

δυναμική απεικόνιση εφαρμογών σε πλατφόρμες FPGA, ακόμη και σε ενσωματωμένα συστήματα τελικού χρήστη.

Κεφάλαιο 9

Μελλοντικές εργασίες

Καθώς οι συσκευές FPGA έχουν αρχίσει να κερδίζουν τη δημοτικότητα λόγω του εγγενούς παραλληλισμού που προσφέρουν και την δυνατότητα επαναδιαμόρφωσης, νέες προκλήσεις έχουν προκύψει που επηρεάζουν την απόδοση και την ευκολία χρήσης. Ορισμένες από αυτές τις προκλήσεις αντιμετωπίστηκαν σε αυτήν την διδακτορική διατριβή, αλλά όπως και με κάθε ερευνητική εργασία υπάρχει περιθώριο βελτίωσης των ήδη προτεινόμενων λύσεων και διαφορετικές διαδρομές που πρέπει να διερευνηθούν.

9.1 Επαναδιαμορφούμενες Αρχιτεκτονικές

Καθώς οι συσκευές FPGA μεγαλώνουν σε μέγεθος και πολυπλοκότητα, υπάρχουν πολλές αρχιτεκτονικές παράμετροι και στοιχεία που πρέπει να διερευνηθούν με σκοπό να βρεθεί επίδρασή τους στη συνολική απόδοση. Σύγχρονα FPGAs περιλαμβάνουν τόσο μπλοκ DSP όσο και μπλοκ μνήμης. Στο μέλλον μια πιθανή πορεία θα είναι ενσωμάτωση των άλλων, πιο πολύπλοκων hardblocks πάνω στην αρχιτεκτονική FPGA, όπως μονάδες Floating Point. Αν και αυτό θα βελτιώσει σημαντικά την απόδοση ορισμένων εφαρμογών, θα εισάγει νέες προκλήσεις στην κατανάλωση ενέργειας, στο χρονισμό και στο εμβαδόν του ολοκληρωμένου. Μια άλλη διαδρομή θα είναι να εισαχθούν αυτά τα hardblocks σε ένα λεπτότερο επίπεδο ενσωματώνοντας τα μέσα στα λογικά μπλοκ. Αυτό απαιτεί πρόσθετη εξερεύνηση των αλγορίθμων CAD, δεδομένου ότι προστίθεται ένα νέο επίπεδο πολυπλοκότητας.

Οι τρισδιάστατες αρχιτεκτονικές υπόσχονται να ανακουφίσουν τα σημερινά προβλήματα απόδοσης, που επιβάλλονται από την σμίκρυνση των τρανζίστορ. Αν και θεωρητικά έχει γίνει αποδεκτή ως μια επιτυχημένη εναλλακτική λύση, στην πραγματικότητα, η διαδικασία κατασκευής 3D είναι μακριά από την ωριμότητα. Υπάρχουν πολλές προκλήσεις ακόμα στο επίπεδο της κατασκευής και της λειτουργίας καθώς και προβλήματα λόγω χαμηλού yield, απαγωγής θερμότητας και διανομής ηλεκτρικής ενέργειας τα οποία θα πρέπει να διερευνηθούν περαιτέρω. Επιπλέον, εκτός από τις βελτιώσεις

σε επίπεδο αρχιτεκτονικής, χρειάζονται ώριμοι και αποτελεσματικοί αλγόριθμοι που πρέπει να λαμβάνουν υπόψη τις ιδιαιτερότητες της τρίτης διάστασης. Είναι επίσης σημαντικό να αναφερθεί ότι οι 3D αρχιτεκτονικές πλεονεκτούν λόγω του επιπλέον βαθμού ελευθερίας (z άξονας), αλλά η ελευθερία αυτή έρχεται με κόστος την αύξηση του χώρου λύσεων που εξερευνούν τα εργαλεία από την σύνθεση μέχρι την δρομολόγηση.

9.2 CAD Εργαλεία

Το πεδίο των εργαλείων CAD σχετικά με FPGAs, δεν είναι πολύ ενεργό ερευνητικά, κυρίως επειδή μέχρι τώρα υπήρχαν αρκετά καλές λύσεις. Καθώς αυξάνονται όμως οι εφαρμογές σε πολυπλοκότητα, οι αδυναμίες των αλγορίθμων τους γίνεται εμφανής. Αποδείχθηκε σε αυτή τη διατριβή ότι οι συμβιβασμοί μεταξύ του χρόνου εκτέλεσης και της ποιότητας της λύσης σε αυτά τα εργαλεία CAD δεν έχει διερευνηθεί πλήρως και δεν κλιμακώνεται αποτελεσματικά σε μεγάλα μεγέθη προβλημάτων.

Ένα πολλά υποσχόμενο πεδίο έρευνας είναι να επεκταθεί το παρόν έργο προς την καλύτερη υποστήριξη ενσωματωμένων συστημάτων που περιλαμβάνουν FPGAs. Σύγχρονα ενσωματωμένα συστήματα υψηλής απόδοσης, όπως τα έξυπνα τηλέφωνα και έξυπνα ρολόγια περιλαμβάνουν CPU/s ή/και GPU/s. Θα πρέπει επίσης να διερευνηθούν τα πιθανά υπολογιστικά οφέλη και τα οφέλη στην κατανάλωση ενέργειας σε τέτοια συστήματα από τη χρήση FPGA. Αυτό θα ανοίξει το δρόμο για μια μεγαλύτερη εμπορική αποδοχή των FPGA στον ενσωματωμένο τομέα. Όπως αναφέρεται σε όλη αυτή διδακτορική διατριβή ένα βασικό χαρακτηριστικό των επαναδιαμορφούμενων αρχιτεκτονικών, είναι η επαναδιαμόρφωση. Νέες στρατηγικές και μεθοδολογίες πρέπει να αναπτυχθούν ώστε να μειωθεί όχι μόνο ο χρόνος επαναδιαμόρφωσης, αλλά να γίνει και η ίδια η διαδικασία πιο εύκολη στη χρήση και πιο ευέλικτη.

Βιβλιογραφία

[1] R.J. Gutmann, J.-Q. Lu, Y. Kwon, J.F. McDonald, and T.S. Cale. Three-dimensional (3d) ics: a technology platform for integrated systems and opportunities for new polymeric adhesives. In *Polymers and Adhesives in Microelectronics and Photonics, 2001. First International IEEE Conference on*, pages 173 180, Oct 2001. doi: 10.1109/POLYTR.2001.973276.

[2] Kirk Saban. Xilinx stacked silicon interconnect technology delivers breakthrough fpga capacity, bandwidth, and power efficiency. Technical Report WP380, Xilinx, Inc., December 2012. SSIT.

[3] C. Ababei, Y. Feng, B. Goplen, Hushrav Mogal, Tianpei Zhang, K. Bazargan, and S. Sapatnekar. Placement and routing in 3d integrated circuits. *Design Test of Computers, IEEE*, 22(6):520 531, 2005. ISSN 0740-7475. doi: 10.1109/MDT.2005.150.

[4] Kostas Siozios and Dimitrios Soudris. A tabu-based partitioning and layer assignment algorithm for 3-d fpgas. *Embedded Systems Letters*, 3(3):97 100, 2011. doi: 10.1109/LES.2011.2161571. URL http://doi.ieeecomputersociety.org/10.1109/LES.2011.2161571.

[5] Vaughn Betz, Jonathan Rose, and Alexander Marquardt, editors. *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, Norwell, MA, USA, 1999. ISBN 0792384601.

[6] Altera. Quartus ii development software, 2011. quartus.

[7] S. Gupta, M. Hilbert, S. Hong, and R. Patti. Techniques for producing 3d ics with high-density interconnect, 2004. 3d-ic.

[8] Xilinx Inc. Xilinx inc. xilinx.

[9] Altera. Altera corporation. altera.

[10] Stamatis Vassiliadis and Dimitrios Soudris. *Fine- and Coarse-Grain Reconfigurable Computing*. Springer Publishing Company, Incorporated, 1st edition, 2007. ISBN 1402065043, 9781402065040.

[11] Harry Sidiropoulos. Development of a design framework for power/ energy consumption estimation in heterogeneous fpga architectures. Master's thesis, National Technical University of Athens (NTUA), School of ECE, 2010. my-master.

[12] Joachim Pistorius and Mike Hutton. Benchmarking method and designs targeting logic synthesis for fpgas. abc.

[13] Minxi Gao, Jie-Hong Jiang, Yunjian Jiang, Yinghua Li, Sinha Subarna, and Robert Brayton. Mvsis. In *Proceedings of the International Workshop on Logic Synthesis*, IWLS, 2001.

[14] Jonathan Rose, Jason Luu, Chi Wai Yu, Opal Densmore, Jeffrey Goeders, Andrew Somerville, Kenneth B. Kent, Peter Jamieson, and Jason Anderson. The vtr project: Architecture and cad for fpgas from verilog to routing. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '12, pages 77 86, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1155-7. doi: 10.1145/2145694.2145708. URL http://doi.acm.org/10.1145/2145694.2145708.

[15] Kara K. W. Poon, Steven J. E. Wilton, and Andy Yan. A detailed power model for field-programmable gate arrays. *ACM Trans. Des. Autom. Electron. Syst.*, 10(2):279 302, April 2005. ISSN 1084-4309. doi: 10.1145/1059876.1059881. URL http://doi.acm.org/10.1145/1059876.1059881.

[16] Tom Feist. White paper: Vivado design suite. Technical report, Xilinx Inc., San Jose, CA, USA 2012. vivado.

[17] Nir Magen, Avinoam Kolodny, Uri Weiser, and Nachum Shamir. Interconnect-power dissipation in a microprocessor. In *Proceedings of the 2004 international workshop on System level interconnect prediction*, SLIP '04, pages 7 13. ACM, 2004. ISBN 1-58113-818-0.

[18] Vasilis F. Pavlidis and Eby G. Friedman. *Three-dimensional Integrated Circuit Design*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2009. ISBN 9780080921860, 9780123743435.

[19] Antonis Papanikolaou, Dimitrios Soudris, and Riko Radojcic. *Three Dimensional System Integration: IC Stacking Process and Design*. Springer Science, 2010. ISBN 978-1-4419-0962-6.

[20] S. Gupta, M. Hilbert, S. Hong, and R. Patti. Techniques for producing 3d ics with high-density interconnect. In *Proceedings of the 21st Intl. VLSI Multilevel Interconnection Conf.*, 2004.

[21] A.W. Topol, D.C.La Tulipe, L. Shi, D.J. Frank, K. Bernstein, S.E. Steen, A. Kumar, G.U. Singco, A.M. Young, K.W. Guarini, and M. Ieong. Three-dimensional integrated circuits. *IBM Journal of Research and Development*, 50(4.5):491 506, July 2006. ISSN 0018-8646. doi: 10.1147/rd.504.0491.

[22] Andre Hahn Pereira and Vaughn Betz. Cad and routing architecture for interposer-based multi-fpga systems. In *Proceedings of the 2014 ACM/SIGDA International Symposium on Field-programmable Gate Arrays*, FPGA '14, pages 75 84, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2671-1. doi: 10.1145/2554688.2554776. URL http://doi.acm.org/10.1145/2554688.2554776.

[23] Qinghong Wu and Kenneth S. McElvain. A fast discrete placement algorithm for fpgas. In *Proceedings of the ACM/SIGDA international symposium on Field Programmable Gate Arrays*, FPGA '12, pages 115 118, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1155-7. doi: 10.1145/2145694.2145713. URL http://doi.acm.org/10.1145/2145694.2145713.

[24] M. Gort and J.H. Anderson. Reducing fpga router run-time through algorithm and architecture. In *Field Programmable Logic and Applications (FPL), 2011 International Conference on*, pages 336 342, sept. 2011. doi: 10.1109/FPL.2011.67.

[25] M.L. Silva and J.C. Ferreira. Generation of partial fpga configurations at run-time. In *Field Programmable Logic and Applications, 2008. FPL 2008. International Conference on*, pages 367 372, sept. 2008. doi: 10.1109/FPL.2008.4629965.

[26] Kostas Siozios, Dimitrios Soudris, and Antonios Thanailakis. A novel allocation methodology for partial and dynamic bitstream generation of fpga architectures. *Journal of Circuits, Systems, and Computers (JCSC)*, 19(3):701 717, May 2010.

[27] A. Flynn, A. Gordon-Ross, and A.D. George. Bitstream relocation with local clock domains for partially reconfigurable fpgas. In *Design, Automation Test in Europe Conference Exhibition, 2009. DATE '09.*, pages 300 303, april 2009.

[28] Amazon elastic compute cloud. URL http://aws.amazon.com/ec2/. amazon-ecc.

[29] Cern trajectory measurement system, . URL http://portal.beam.ltd.uk/support/cern/. cern.

[30] M. Birk, S. Koehler, M. Balzer, M. Huebner, N.V. Ruiter, and J. Becker. Fpga-based embedded signal processing for 3d ultrasound computer tomography. In *Real Time Conference (RT), 2010 17th IEEE-NPSS*, pages 1 5, 2010. doi: 10.1109/RTC.2010.5750384.

[31] Chipit automated fpga-based prototyping systems, . URL http://www.synopsys.com/Systems/FPGABasedPrototyping/CHIPit/. chipit.

[32] Keith D. Underwood, K. Scott Hemmert, and Craig D. Ulmer. From silicon to science: The long road to production reconfigurable supercomputing. *ACM TRETS*, 2(4):26:1 26:15, September 2009. ISSN 1936-7406.

[33] Robert Kirchgessner, Greg Stitt, Alan D. George, and Herman Lam. Virtualrc: a virtual fpga platform for applications and tools portability. In *FPGA'12*, pages 205 208, 2012.

[34] M. Hübner, P. Figuli, R. Girardey, D. Soudris, K. Siozios, and J. Becker. A heterogeneous multicore system on chip with run-time reconfigurable virtual fpga architecture. In *Proceedings of the 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum*, IPDPSW '11, pages 143 149, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-0-7695-4577-6. doi: 10.1109/IPDPS.2011.135. URL http://dx.doi.org/10.1109/IPDPS.2011.135.

[35] Vikas Aggarwal, Alan D. George, Changil Yoon, Kishore Yalamanchili, and Herman Lam. Shmem+: A multilevel-pgas programming model for reconfigurable supercomputing. *ACM Trans. Reconfigurable Technol. Syst.*, 4(3):26:1 26:24, August 2011. ISSN 1936-7406. doi: 10.1145/2000832.2000838. URL http://doi.acm.org/10.1145/2000832.2000838.

[36] Brian Holland, Alan D. George, Herman Lam, and Melissa C. Smith. An analytical model for multilevel performance prediction of multi-fpga systems. *ACM Trans. Reconfigurable Technol. Syst.*, 4(3):27:1 27:28, August 2011. ISSN 1936-7406. doi: 10.1145/2000832.2000839. URL http://doi.acm.org/10.1145/2000832.2000839.

[37] Harry Sidiropoulos, Kostas Siozios, Peter Figuli, Dimitrios Soudris, Michael Hübner, and Jürgen Becker. Jitpr: A framework for supporting fast application's implementation onto fpgas. *ACM Trans. Reconfigurable Technol. Syst.*, 6(2):7:1 7:12, August 2013. ISSN 1936-7406. doi: 10.1145/2492185. URL http://doi.acm.org/10.1145/2492185.

[38] Katherine Compton, Zhiyuan Li, James Cooley, Stephen Knol, and Scott Hauck. Configuration relocation and defragmentation for run-time reconfigurable computing. *IEEE Trans. Very Large Scale Integr. Syst.*, 10(3):209 220, June 2002. ISSN 1063-8210. doi: 10.1109/TVLSI.2002.1043324. URL http://dx.doi.org/10.1109/TVLSI.2002.1043324.

[39] Roman Lysecky, Frank Vahid, and Sheldon X.-D. Tan. Dynamic fpga routing for just-in-time fpga compilation. In *DAC*, DAC '04, pages 954 959, New York, NY, USA, 2004. ACM. ISBN 1-58113-828-8.

[40] Maxeler technologies. URL https://www.maxeler.com/. maxeler.

[41] Sigma. URL https://www.sigmaphoto.com/. sigma.

[42] apertus. apertus open source camera, 2009. URL https://www.apertus.org/. apertus.

[43] White paper:implementing fpga design with the opencl. Technical report, Altera, Corp., San Jose, CA, USA, 2012. opencl.

[44] Christian Fobel, Gary Grewal, and Deborah Stacey. Forward-scaling, serially equivalent parallelism for fpga placement. In *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI*, GLSVLSI '14, pages 293 298, New York, NY, USA, 2014. ACM.

ISBN 978-1-4503-2816-6. doi: 10.1145/2591513.2591543. URL http://doi.acm.org/10.1145/2591513.2591543.

[45] P. Banerjee, S. Bhattacharjee, S. Sur-Kolay, S. Das, and S.C. Nandy. Fast fpga placement using space-filling curve. In *Field Programmable Logic and Applications, 2005. International Conference on*, pages 415 420, Aug 2005. doi: 10.1109/FPL.2005.1515757.

[46] B. Raza, H. Parvez, and M. Mohiuddin. Exploring alternate trade-offs of placement quality versus runtime in simulated annealing algorithm. In *Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), 2014 9th International Symposium on*, pages 1 5, May 2014. doi: 10.1109/ReCoSoC.2014.6860685.

[47] Matthew An, J.Gregory Steffan, and Vaughn Betz. Speeding up fpga placement: Parallel algorithms and methods. In *Field-Programmable Custom Computing Machines (FCCM), 2014 IEEE 22nd Annual International Symposium on*, pages 178 185, May 2014. doi: 10.1109/FCCM.2014.60.

[48] M. Gort and J.H. Anderson. Accelerating fpga routing through parallelization and engineering enhancements special section on par-cad 2010. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 31(1):61 74, Jan 2012. ISSN 0278-0070. doi: 10.1109/TCAD.2011.2165715.

[49] Yehdhih Ould Mohammed Moctar and Philip Brisk. Parallel fpga routing based on the operator formulation. In *Proceedings of the 51st Annual Design Automation Conference*, DAC '14, pages 193:1 193:6, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2730-5. doi: 10.1145/2593069.2593177. URL http://doi.acm.org/10.1145/2593069.2593177.

[50] Adrian Ludwin and Vaughn Betz. Efficient and deterministic parallel placement for fpgas. *ACM Trans. Des. Autom. Electron. Syst.*, 16(3):22:1 22:23, June 2011. ISSN 1084-4309.

[51] M. Gort and J.H. Anderson. Analytical placement for heterogeneous fpgas. In *Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on*, pages 143 150, 2012. doi: 10.1109/FPL.2012.6339278.

[52] Huimin Bian, Andrew C. Ling, Alexander Choong, and Jianwen Zhu. Towards scalable placement for fpgas. In *Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '10, pages 147 156, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-911-4. doi: 10.1145/1723112.1723140. URL http://doi.acm.org/10.1145/1723112.1723140.

[53] Elena Moscu Panainte, Koen Bertels, and Stamatis Vassiliadis. Fpga-area allocation for partial run-time reconfiguration. In *Program for Research on Integrated Systems and Circuits*, ProRISC '05, pages 415 420, 2005.

[54] Manish Handa and Ranga Vemuri. An efficient algorithm for finding empty space for online fpga placement. In *Proceedings of the 41st annual Design Automation Conference*, DAC '04, pages 960 965, New York, NY, USA, 2004. ACM. ISBN 1-58113-828-8. doi: 10.1145/996566.996820. URL http://doi.acm.org/10.1145/996566.996820.

[55] H. Kalte, G. Lee, M. Porrmann, and U. Ruckert. Replica: A bitstream manipulation filter for module relocation in partial reconfigurable systems. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 3 - Volume 04*, IPDPS '05, pages 151.2 , Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2312-9.

[56] F. Vahid, G. Stitt, and R. Lysecky. Warp processing: Dynamic translation of binaries to fpga circuits. *Computer*, 41(7):40 46, july 2008. ISSN 0018-9162. doi: 10.1109/MC.2008.240.

[57] Manuel G. Gericota, Gustavo R. Alves, Miguel L. Silva, and Jose M. Ferreira. Run-time management of logic resources on reconfigurable systems. In *Proceedings of the conference on Design, Automation and Test in Europe - Volume 1*, DATE '03, pages 10974 , Washington, DC, USA, 2003. IEEE Computer Society. ISBN 0-7695-1870-2. URL http://dl.acm.org/citation.cfm?id=789083.1022850.

[58] Kostas Siozios, Vasilis F. Pavlidis, and Dimitrios Soudris. A novel framework for exploring 3-d fpgas with heterogeneous interconnect fabric. *ACM Trans. Reconfigurable Technol. Syst.*, 5(1):4:1 4:23, March 2012. ISSN 1936-7406.

[59] Blif, berkeley logic interchange format, 1992. blif.

[60] *Altera Stratix Device Handbook*, 2005. stratix.

[61] *Virtex-II Platform FPGA Handbook*, 2001.

[62] S.J.E. Wilton and N.P. Jouppi. Cacti: an enhanced cache access and cycle time model. *Solid-State Circuits, IEEE Journal of*, 31(5):677 688, May 1996. ISSN 0018-9200. doi: 10.1109/4.509850.

[63] Jan Rabaey. *Low Power Design Essentials*. Springer Publishing Company, Incorporated, 1st edition, 2009. ISBN 0387717129, 9780387717128.

[64] Jason Luu, Ian Kuon, Peter Jamieson, Ted Campbell, Andy Ye, Wei Mark Fang, and Jonathan Rose. Vpr 5.0: Fpga cad and architecture exploration tools with single-driver routing, heterogeneity and process scaling. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '09, pages 133 142, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-410-2. doi: 10.1145/1508128.1508150. URL http://doi.acm.org/10.1145/1508128.1508150.

[65] A. Gayasen, V. Narayanan, M. Kandemir, and Arifur Rahman. Designing a 3-d fpga: Switch box architecture and thermal issues. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 16(7):882 893, 2008. ISSN 1063-8210.

[66] Mingjie Lin, Abbas El Gamal, Yi-Chang Lu, and S. Simon Wong. Performance benefits of monolithically stacked 3-d fpga. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 26(2):216 229, 2007. URL http://dblp.uni-trier.de/db/journals/tcad/tcad26.html#LinGLW07.

[67] Guy Lemieux and David A. Lewis. *Design of interconnection networks for programmable logic*. Kluwer, 2004. ISBN 978-1-4020-7700-5.

[68] David Fang, Song Peng, Chris LaFrieda, and Rajit Manohar. A three-tier asynchronous fpga. In *Proceedings of the International VLSI/ULSI Multilevel Interconnection Conference*. Citeseer, 2006.

[69] Shamik Das, Andy Fan, Kuan-Neng Chen, Chuan Seng Tan, Nisha Checka, and Rafael Reif. Technology, performance, and computer-aided design of three-dimensional integrated circuits. In Charles J. Alpert and Patrick Groeneveld, editors, *ISPD*, pages 108 115. ACM, 2004. ISBN 1-58113-817-2. URL http://dblp.uni-trier.de/db/conf/ispd/ispd2004.html#DasFCTCR04.

[70] Chen Dong, Deming Chen, S. Haruehanroengra, and Wei Wang 0003. 3-d nfpga: A reconfigurable architecture for 3-d cmos/nanomaterial hybrid digital circuits. *IEEE Trans. on Circuits and Systems*, 54-I(11):2489 2501, 2007. URL http://dblp.uni-trier.de/db/journals/tcas/tcasI54.html#DongCH007.

[71] Roto Le, Sherief Reda, and R. Iris Bahar. High-performance, cost-effective heterogeneous 3d fpga architectures. In Paul Chow and Peter Y. K. Cheung, editors, *FPGA*, page 286. ACM, 2009. ISBN 978-1-60558-410-2. URL http://dblp.uni-trier.de/db/conf/fpga/fpga2009.html#LeRB09.

[72] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. Multilevel hypergraph partitioning: Application in vlsi domain. In *Proceedings of the 34th Annual Design Automation Conference*, DAC '97, pages 526 529, New York, NY, USA, 1997. ACM. ISBN 0-89791-920-3. doi: 10.1145/266021.266273. URL http://doi.acm.org/10.1145/266021.266273.

[73] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *Proceedings of the 19th Design Automation Conference*, DAC '82, pages 175 181, Piscataway, NJ, USA, 1982. IEEE Press. ISBN 0-89791-020-6. URL http://dl.acm.org/citation.cfm?id=800263.809204.

[74] B. W. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graphs. *The Bell system technical journal*, 49(2):291 308, 1970.

[75] Dae Hyun Kim and Sung Kyu Lim. Through-silicon-via-aware delay and power prediction model for buffered interconnects in 3d ics. In *Proceedings of the 12th ACM/IEEE International Workshop on System Level Interconnect Prediction*, SLIP '10, pages 25 32, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0037-7. doi: 10.1145/1811100.1811108. URL http://doi.acm.org/10.1145/1811100.1811108.

[76] Harry Sidiropoulos, Kostas Siozios, and Dimitrios Soudris. A novel 3-d fpga architecture targeting communication intensive applications. *Journal of Systems Architecture - Embedded Systems Design*, pages 32 39, 2014.

[77] Vinod Pangracious, Emna Amouri, Zied Marrakchi, and Habib Mehrez. Architecture level optimization of 3-dimensional tree-based FPGA. *Elsevier Microelectronics Journal*, 45(4):355 366, April 2014. doi: 10.1016/j.mejo.2013.12.011.

[78] Shamik Das, Anantha Chandrakasan, and Rafael Reif. Design tools for 3-d integrated circuits. In *Proceedings of the 2003 Asia and South Pacific Design Automation Conference*, ASP-DAC '03, pages 53 56. ACM, 2003. ISBN 0-7803-7660-9.

[79] Kostas Siozios, Vasilis F. Pavlidis, and Dimitrios Soudris. A software-supported methodology for exploring interconnection architectures targeting 3-d fpgas. In *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '09, pages 172 177. European Design and Automation Association, 2009. ISBN 978-3-9810801-5-5.

[80] Chun-Lung Hsu, Yu-Sheng Huang, and Fong-Chao Lee. Interlaced switch boxes placement for three-dimensional fpga architecture design. *Int. J. Circuit Theory Appl.*, 40(5):489 502, May 2012. ISSN 0098-9886.

[81] Mcnc standard cell benchmark circuits from the microelectronics center of north carolina, 1991. URL http://vlsicad.cs.binghamton.edu/gz/PDWorkshop91.tgz. mcnc.

[82] B.K. Britton, Y.T. Oh, W. Oswald, H. T. Nguyen, S. Singh, G. Lee, W.-B. Leung, C. Spivak, J. Steward, and C-T Chen. Second generation orca architecture utilizing 0.5 μm process enhances the speed and usable gate capacity of fpgas. In *ASIC Conference and Exhibit, 1994. Proceedings., Seventh Annual IEEE International*, pages 474 478, 1994.

[83] L. McMurchie and C. Ebeling. Pathfinder: A negotiation-based performance-driven router for fpgas. In *Third International ACM Symposium on Field-Programmable Gate Arrays, 1995.*, pages 111 117, 1995.

[84] Harry Sidiropoulos, Kostas Siozios, and Dimitrios Soudris. On supporting efficient implementation of communication-intensive applications onto 3d fpgas. In *Workshop on Reconfigurable Computing (WRC)*, Jan. 2012.

[85] Melvin A. Breuer. A class of min-cut placement algorithms. In *Proceedings of the 14th Design Automation Conference, DAC '77, New Orleans, Louisiana, USA, June 20-22, 1977*, pages 284 290, 1977. URL http://dl.acm.org/citation.cfm?id=809144.

[86] Hans Eisenmann and F.M. Johannes. Generic global placement and floorplanning. In *Design Automation Conference, 1998. Proceedings*, pages 269 274, June 1998.

[87] Harry Sidiropoulos, Kostas Siozios, and Dimitrios Soudris. On supporting rapid exploration of memory hierarchies onto fpgas. *Journal of Systems Architecture*, 59(2):78 90, 2013. ISSN 1383-7621. doi: http://dx.doi.org/10.1016/j.sysarc.2012.10.003. URL http://www.sciencedirect.com/science/article/pii/S1383762112000896. Best Selected Papers from the 5th HiPEAC Workshop on Reconfigurable Computing.

[88] Russell Tessier. Fast placement approaches for fpgas. *ACM Trans. Des. Autom. Electron. Syst.*, 7(2):284 305, April 2002. ISSN 1084-4309. doi: 10.1145/544536.544540. URL http://doi.acm.org/10.1145/544536.544540.

[89] P. Figuli, M. Hubner, R. Girardey, F. Bapp, T. Bruckschlogl, F. Thoma, J. Henkel, and J. Becker. A heterogeneous soc architecture with embedded virtual fpga cores and runtime core fusion. In *Adaptive Hardware and Systems (AHS), 2011 NASA/ESA Conference on*, pages 96 103, June 2011. doi: 10.1109/AHS.2011.5963922.

[90] *7 Series FPGAs Configurable Logic Block User Guide*, November 2014. xilinx-clb.

[91] *ProASIC3 Flash Family FPGAs with Optional Soft ARM Support*, January 2013. actel-versatile.

[92] Jason Evans. A scalable concurrent malloc (3) implementation for freebsd. In *Proc. of the BSDCan Conference, Ottawa, Canada*, 2006.

[93] LockLess Inc. LockLess website. http://locklessinc.com/, 2012. lockless.