Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής
και Υπολογιστών

Εργαλείο Αυτοματοποιημένης Εξερεύνησης Απόδοσης - Επιφάνειας Υλικού - Ισχύος - Θερμικής Συμπεριφοράς Δυναμικά Επαναδιατάξιμων Συνεπεξεργαστών Υλικού

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΚΟΥΡΟΥΜΟΥΝΗΣ ΧΡΙΣΤΟΣ

Επιβλέπων : Κιαμάλ Πεκμεστζή
Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2009

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής
και Υπολογιστών

Εργαλείο Αυτοματοποιημένης Εξερεύνησης Απόδοσης - Επιφάνειας Υλικού - Ισχύος - Θερμικής Συμπεριφοράς Δυναμικά Επαναδιατάξιμων Συνεπεξεργαστών Υλικού

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΚΟΥΡΟΥΜΟΥΝΗΣ ΧΡΙΣΤΟΣ

Επιβλέπων : Κιαμάλ Πεκμεστζή
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 3η Σεπτεμβρίου 2009.

Κιαμάλ Πεκμεστζή Καθηγητής Ε.Μ.Π.	Δημήτριος Σούντρης Επίκουρος Καθηγητής Ε.Μ.Π.	Γιώργος Οικονομάκος Λέκτορας Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2009

Σκουρουμούνης Χρίστος

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

Στους γονείς μου ...

Περίληψη

Με την ανάπτυξη της τεχνολογίας VLSI η ολοκλήρωση εκατομμυρίων τρανζίστορ είναι πλέον πραγματικότητα. Υπάρχουν πλέον μεγάλες δυνατότητες στη σχεδίαση κυκλωμάτων οι οποίες μπορούν να καλύψουν τις αυξανόμενες ανάγκες που επιβάλει η συνεχόμενη τεχνολογική πρόοδος. Οι σχεδιαστές όμως δεν μπορούν να παρακολουθήσουν την ταχύτητα και τη διάταξη των κυκλωμάτων που είναι πλέον διαθέσιμες, όπως και τις συνεχής αλλαγές που γίνονται στα πρωτόκολλα και τις εφαρμογές των κυκλωμάτων. Έτσι για να ξεπεραστούν τα διάφορα προβλήματα έγινε επιτακτική η ανάγκη για σύνθεση υλικού από περιγραφή υψηλού επιπέδου, γνωστή ως High Level Synthesis.

Η σύνθεση υψηλού επιπέδου (HLS), αποτελεί το σύνολο των μεθοδολογιών που επιτρέπουν να δημιουργηθεί η περιγραφή της υλοποίησης του υλικού κυκλώματος από περιγραφή σε γλώσσα υψηλού επιπέδου. Η σύνθεση υψηλού επιπέδου μεταγλωττίζει την περιγραφή της επιθυμητής συμπεριφοράς, η οποία είναι σε γλώσσα υψηλού επιπέδου, και δημιουργεί το κύκλωμα που υλοποιεί τη συγκεκριμένη συμπεριφορά. Σκοπός αυτής της διπλωματικής εργασίας είναι η μοντελοποίηση και εξερεύνηση του χώρου σχεδίασης για τη σύνθεση υψηλού επιπέδου σε αρχιτεκτονικές Coarse Grained Reconfigurable, όπως και η υλοποίηση εργαλείου αυτόματης εξερεύνησης της απόδοσης, επιφάνειας υλικού, ισχύος και θερμικής συμπεριφοράς των συγκεκριμένων αρχιτεκτονικών.

Για τη σύνθεση υψηλού επιπέδου (HLS) χρησιμοποιήθηκε το εργαλείο SPARK. Για τα επόμενα επίπεδα εξερεύνησης χρησιμοποιήθηκαν επίσης το εργαλείο σύνθεσης Synopsys Design Compiler και το εργαλείο χωροθέτησης με δυνατότητα θερμικής διαχείρισης HotSpot που αναπτύχθηκε στο Πανεπιστήμιο της Virginia.

Η διπλωματική εργασία κινείται πάνω σε 4 βασικούς άξονες:

1. Σύσταση αυτοματοποιημένης ροής σχεδίασης από επίπεδο της σύνθεσης ψηλού επιπέδου (HLS) σε επίπεδο χωροθέτησης σχεδίου μετά τη σύνθεση (Post-Synthesis Floorplan) (HLS Framework)

2. Μοντελοποίηση αρχιτεκτονικών CGRArch (Coarse Grained Reconfigurable Architectures)

3. Επέκταση του πλαισίου εργασίας: Εξερεύνηση με πολλαπλά αφαιρετικά επίπεδα σχεδίασης

4. Επέκταση του χώρου σχεδίασης για HLS - Μοντελοποίηση και Εξερεύνηση

Λέξεις κλειδιά

Σύνθεση υψηλού επιπέδου (High Level Synthesis), Αρχιτεκτονικές Coarse Grained Reconfigurable, Σύνθεση κυκλώματος, Εργαλείο HotSpot, Εργαλείο Αυτοματοποιημένης Εξερεύνησης σε επίπεδο HLS.

Abstract

With the development of VLSI technology, the itergration millions of transistors has become a reality. Henceforth, big possibilities exist in the designing of circuits that can cover the increasing needs that the possessed technological progress imposes. The designers however cannot follow the speed and the provision of circuits that is available, as the continuous changes in protocols and applications of circuits. Thus in order to exceeded different problems, the is the need for intergration of material from a high level description, known as High Level Synthesis.

The High Level Synthesis (HLS), is a set of methodologies that give the possibility for descriping the concretisation of material circuit from a description written in a high level language. The HLS interperts the description of desirable behavior, which is in a high level language, and creates the circuit that implements the particular behavior. The main goal of this diploma thesis is the modelling and exploration of the designing space in HLS for Coarse Grained Reconfigurable Architectures, as the concretisation of tool of automatic exploration for performance, area, power and thermal aware floorplanning for the theses architectures.

For the High Level Synthesis level, SPARK was used. The following tools were also used for the next levels of exploration (i) Synopsys Design Compiler synthesis tool and (ii) floorplanning tool with possibility of thermic management HotSpot that was developed in the University of Virginia.

This diploma thesis is based on 4 fundamental axes:

1. Automated Desigh flow constitution from HLS level to Post-Synthesis Floorplan (HLS Framework)
2. Modelling of CGRArch (Coarse Grained Reconfigurable Architectures)
3. Framework extension: Exploration with multple designed abstraction levels
4. Extend of HLS design space - Modelling and Exploration

Key words

High Level Synthesis (HLS), Coarse Grained Reconfigurable Architectures (CGRArch), Synopsys Design Compiler synthesis tool, HotSpot tool, Automated exploration tool in HLS level

Ευχαριστίες

Αυτή η διπλωματική εργασία πραγματοποιήθηκε στο Εργαστήριο Μικρουπολογιστών και Ψηφιακών Συστημάτων της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών (MICROLAB) του Εθνικού Μετσόβιου Πολυτεχνείου, υπό την επίβλεψη του Καθηγητή Κιαμάλ Πεκμεστζή.

Αρχικά θα ήθελα να ευχαριστήσω τον καθηγήτη μου κ.Κιαμάλ Πεκμεστζή για την εποπτεία και την άριστη συνεργασία καθ'όλη την εκπόνηση της διπλωματικής μου εργασίας, όπως επίσης και για την συμβολή του στην ολοκλήρωση μου ως μηχανικού μέσα από τις διδασκαλίες του. Ακόμα ιδιαίτερες ευχαριστίες οφείλω στους καθήγητές μου, κ. Γιώργο Οικονομάκο και κ. Δημήτριο Σούντρη για τη σημαντική αρωγή και τις πολύτιμες συμβουλές τους κατά τη διάρκεια της συνεργασίας μας.

Ακολούθως θα ήθελα να ευχαριστήσω όλα τα μέλη του Εργαστηρίου Μικρουπολογιστών και Ψηφιακών Συστημάτων. Πρωτίστως δε το Διδακτορικό Ερευνητή Σωτήρη Ξυδή για την συνεχή και αμέριστη καθοδήγηση και βοήθεια που μου πρόσφερε, το χρόνο τον οποίο διάθεσε και την άψογη συνεργασία που είχαμε όλο αυτό τον καιρό.

Τέλος θα ήθελα να αφιερώσω τη διπλωματική εργασία στους γονείς μου, Δημήτρη και Ελισάβετ και όλη την οικογένεια μου, για την συνεχή και αμέριστη αγάπη, εμπιστοσύνη και συμπαράσταση τους όλα αυτά τα χρόνια, στη δύσκολη και συνεχή προσπάθεια για την επίτευξη των στόχων μου.

Χρίστος Σκουρουμούνης,

Αθήνα, Σεπτέμβριος 2009.

Η εργασία αυτή είναι επίσης διαθέσιμη ως Τεχνική Αναφορά ....., Εθνικό Μετσόβιο Πολυτεχνείο, Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών, Εργαστήριο Μικρουπολογιστών και Ψηφιακών Συστημάτων, Σεπτέμβριος 2009.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Γενικά

Ο τομέας των υπολογιστικών συστημάτων κυριαρχεί το ενδιαφέρον τις τελευταίες δεκαετίες με επίδραση σε όλες σχεδόν τους πτυχές της ανθρώπινης ζωής. Ο νόμος του Moore, που δηλώνει τη ραγδαία εξέλιξη των ολοκληρωμένων συστημάτων, αν και ειπώθηκε πριν από περίπου 50 χρόνια, ισχύει ακόμα υποδηλώνοντας την αύξηση των τρανζίστορ στα ολοκληρωμένα κυκλώματα και στο μέλλον. Συγκριτικά με τα πρώτα κυκλώματα που κατασκευάστηκαν στη δεκαετία του '60 και ολοκλήρωναν μερικά τρανζίστορς, τα τελευταία χρόνια ο αριθμός τρανζίστορ σ'ένα ολοκληρωμένο έχει ξεπεράσει το μισό δισεκατομμύριο. Σημαντική εξέλιξη επέφερε και η ανάπτυξη της διαδικασίας ολοκλήρωσης κυκλωμάτων με συνδιασμό χιλιάδων τρανζίστορς σε ένα εναιαίο ψηφίδα υλικού, γνωστής ως Very-large-scale integration (VLSI). Η τεχνική αυτή επέδρασε καταλυτικά και στην ανάπτυξη διαφόρων τεχνολογιών που ανήκουν στην κατηγορία των υπολογιστικών κυκλωμάτων, όπως είναι τα ενσωματομένα συστήματα, η αρχιτεκτονική υπολογιστών, η μικροηλεκτρονική κτλ.

Κύριος γνώμονας της ανάπτυξης των υπολογιστικών συστημάτων αποτελούσε η βελτίωση της απόδοσης τους στα όρια της τεχνολογίας και του κόστους. Με την πάροδο όμως του χρόνου, και καθώς η επιστήμη των υπολογιστών δημιουργούσε καινούργιες προοπτικές, η απόδοση και μόνο δεν μπορούσε να αποτελεί το μοναδικό στόχο της σχεδίασης και κατασκευής υπολογιστικών συστημάτων. Αντιθέτως, η ανάπτυξη πλέον βασίζεται σε πλαίσιο διαφόρων παραγόντων, κύριοι εκ των οποίων είναι η απόδοση, το κόστος, η επιφάνεια υλικού και η κατανάλωση ισχύος. Ανάλογα με τη χρήση και τους περιορισμούς που θέτει η χρήση κάθε υπολογιστικού συστήματος, οι παράγοντες που αναφέρθηκαν έχουν και διαφορετική βαρύτητα.

Στα ενσωματομένα συστήματα, η επιφάνεια υλικού και η κατανάλωση ισχύος αποτελούν τους κύριους παράγοντες για την σχεδίαση και κατασκευή τους. Αυτά αποτελούν υπολογιστικά συστήματα ειδικού-σκοπού, σχεδιασμένα να εκτελούν αρκετές λειτουργίες με βασικούς περιορισμούς, και αποτελούν κομμάτι στις πλείστες σύγχρονες ηλεκτρονικές συσκευές. Στα ενσωματομένα συστήματα εφαρμόζονται διάφορες αρχιτεκτονικές, όπως η Coarse Grained Reconfigurable Architectures (CGRArch). Μεγάλη χρησιμότητα στην τεχνολογία των ενσωματομένων βρήκε και η τεχνική της σύνθεσης υψηλού επιπέδου (High Level Synthesis - HLS), με την οποία μια περιγραφή συμπεριφοράς ενός του κυκλώματος δημιουργείται αυτόματα το σχέδιο σε επίπεδο υλικού.

Στα κεφάλαια που ακολουθούν παρουσιάζεται η εξερεύνηση με περιορισμούς την απόδοση, την επιφάνεια υλικού και την κατανάλωση ισχύος με χρήση High Level Synthesis σε αρχιτεκτονικές Coarse Grained Reconfigurables.

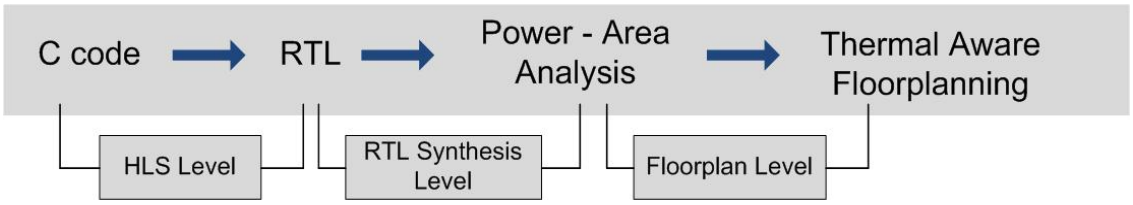
1.2 Γενικοί Στόχοι - Περιγραφή

Η διπλωματική εργασία κινείται πάνω σε 4 βασικούς άξονες καθένας από τους οποίους αναπτύσσεται ξεχωριστά στη συνέχεια.

1. **Σύσταση αυτοματοποιημένης ροής σχεδίασης από επίπεδο της σύνθεσης ψηλού επιπέδου (HLS) σε επίπεδο χωροθέτησης σχεδίου μετά τη σύνθεση (Post-Synthesis Floorplan) (HLS Framework)**
2. **Μοντελοποίηση αρχιτεκτονικών CGRArch (Coarse Grained Reconfigurable Architectures)**
3. **Επέκταση του πλαισίου εργασίας: Εξερεύνηση με πολλαπλά αφαιρετικά επίπεδα σχεδίασης**
4. **Επέκταση του χώρου σχεδίασης για HLS - Μοντελοποίηση και Εξερεύνηση**

1.2.1 *ΑΞΟΝΑΣ 1:* Σύσταση αυτοματοποιημένης ροής σχεδίασης από επίπεδο της σύνθεσης ψηλού επιπέδου (HLS) σε επίπεδο χωροθέτησης σχεδίου μετά τη σύνθεση (Post-Synthesis Floorplan) (HLS Framework)

Στόχος του αξονα αυτού, είναι η δημιουργία ενός πλαισίου εργασίας για την σύνδεση και μεταβίβαση μεταξύ των διαφόρων επιπέδων-εργαλείων εργασίας. Το πλαίσιο εργασίας αποτελείται από τα ακόλουθα μέρη:
Χρησιμοποιείται κώδικας C για διάφορα kernels (όπως dct, mpeg, fft, και άλλα) με το εργαλείο για σύνθεση υψηλού επιπέδου SPARK για να παραχθεί κώδικας επιπέδου μεταφοράς καταχωρητών (Register Transfer Level RTL). Αυτός ο κώδικας RTL μεταφέρεται στο επίπεδο της σύνθεσης και με χρήση του εργαλείου Synopsys Design Compiler παράγεται η λίστα πυλών (NetList) και γίνεται ανάλυση της ισχύος που καταναλώνεται. Τέλος τα δεδομένα για την κατανάλωση ισχύος και την επιφάνεια υλικού του σχεδίου εξάγονται για τη χωροθέτηση του κυκλώματος. Με τη χρήση του εργαλείου HotFloorplan και HotSpot παράγεται με θερμική διαχείριση σχέδιο χωροθέτησης των βασικών μονάδων.

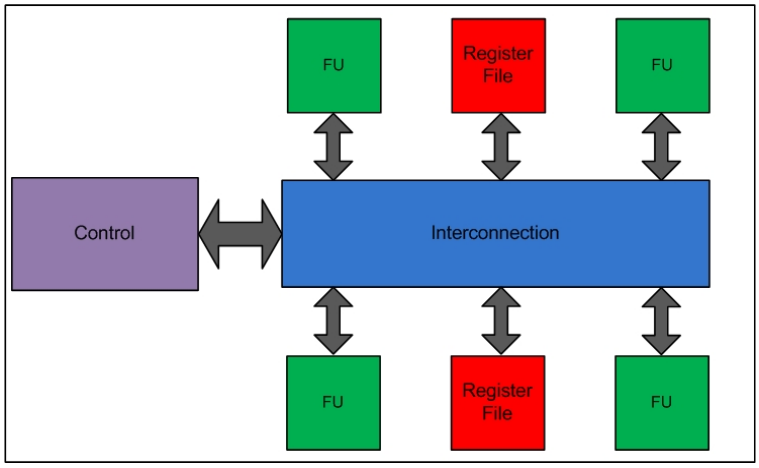


Σχήμα 1.1: Πλαίσιο εργασίας για HLS (HLS Framework).

1.2.2 *ΑΞΟΝΑΣ 2:* Μοντελοποίηση αρχιτεκτονικών για CGRArch (Coarse Grained Reconfigurable Architectures)

Χρησιμοποιείται ειδική δομή στον κώδικα C που περιέχει τα kernels για τον CGRArch με τη χρήση δομών με if-then-else και ένα σήμα για την πλοήγηση μεταξύ τους. Με αυτό τον τρόπο μοντελοποιείται κατάλληλα η δομή που θα έχει η αναπαράσταση για το CGRArch η οποία θα

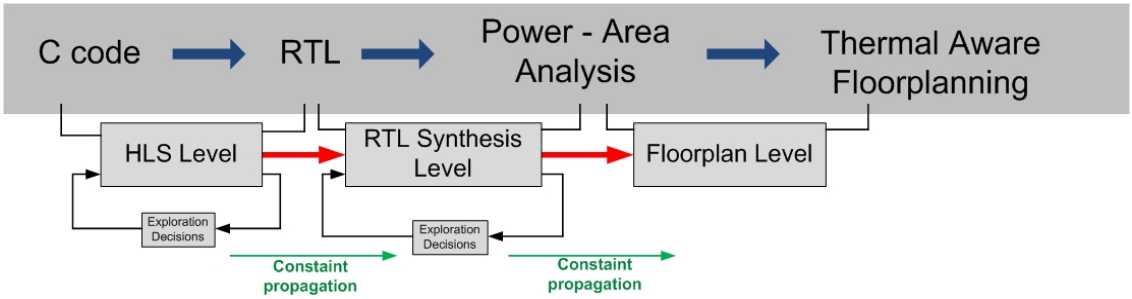
αλλάζει ανάλογα με το συγκεκριμένο κομμάτι του κώδικα που θα εκετελείται. Ανάλογα με τις ρυθμίσεις που θα υπάρχουν και το κομμάτι του κώδικα που θα εκτελεστεί θα υλοποιείται και διαφορετική αρχιτεκτονική διασύνδεσης. Το αρχιτεκτονικο πρότυπο που ακολοθείται σε αυτή την προσέγγιση φαίνεται στο Σχήμα 1.2.



Σχήμα 1.2: Μοντελοποίηση της αρχιτεκτονικής για CGRArch.

1.2.3 *ΑΞΟΝΑΣ 3:* Επέκταση του πλασίου εργασίας: Εξερεύνηση με πολλαπλά αφαιρετικά επίπεδα σχεδίασης

Επεκτήνουμε το πλαίσο εργασίας (framework) για τα πολλαπλά αφαιρετικά επίπεδα εργασίας που έχουμε στη σχεδίαση (HLS, Synthesis, Thermal Aware Floorplan).



Σχήμα 1.3: Επέκταση του πλασίου εργασίας: Εξερεύνηση με πολλαπλά αφαιρετικά επίπεδα σχεδίασης.

Όπως φαίνεται και στο σχήμα 1.3 σε κάθε επίπεδο με την σύνδεση των κομματιών της εξερεύνησης υπάρχει και επατροφοδότηση από την έξοδο που παράγεται για να παρθούν αποφάσεις για βελτιστοποίηση των αποτελεσμάτων και των περιορισμών (constrains). Αυτές οι αποφάσεις και οι περιορισμοί μεταβιβάζονται στο επόμενο στάδιο για να συνεχιστεί η εξερεύνηση περιορίζοντας έτσι σημαντικά το εύρος της και συνεπώς το χρόνο εκτέλεσης.

1.2.4 *ΑΞΟΝΑΣ 4:* Μοντελοποίηση του χώρου σχεδίασης για HLS

Επέκταση του χώρου σχεδίασης για HLS

Ο κλασσικός χώρος σχεδίασης για τη σύνθεση υψηλού επιπέδου (HLS) αφορούσε μέχρι στιγμής την εξερεύνηση του πλήθους διαθέσιμων λειτουργικών μονάδων (αριθμό αθροιστών (ALUs) και αριθμό πολλαπλασιαστών (MULs)) και αγνοούσε διάφορες άλλες παραμέτρους. Ο χώρος αυτός επεκτείνεται για να καλύψει τις διαφορες παραμέτρους που υπάρχουν στη σχεδίαση, όπως οι μετασχηματισμοί σε κώδικα εφαρμογής (Code Transformations). Οι διάφορες παραμέτροι αντιστοιχούν στα πολλαπλά αφαιρετικά επίπεδα:

- Επίπεδο HLS: Από C κώδικα σε RTL κώδικα έχουμε παραμέτρους με τις οποίες διαμορφώνεται και ο κώδικας που παράγεται. Αυτές οι παραμέτροι αφορούν στον αριθμός των λειτουργικών μονάδων, στην αλυσιδωτή εκτέλεση (chaining), στη διεύρυνση κώδικα κλπ.

- Επίπεδο Σύνθεσης: Σ'αυτό το επίπεδο από τον κώδικα RTL παράγεται μετά από σύνθεση το δίκτυο πυλών (NET-LIST) διαφοροποιημένη ανάλογα με την επιφάνεια υλικού, ανάλογα με το ρολόι χρονισμού που θα τεθεί στη σχεδίαση και την ενέργεια που καταναλώνεται.

- Επίπεδο Χωροθέτησης με δυνατότητα θερμικής διαχείρισης: Τα δεδομένα για την κατανάλωση ισχύος που εξάγουμε από το προηγούμενο επίπεδο, εισάγονται στο εργαλείο θερμικής διαχείρισης και floorplanning HotFloorplan και HotSpot για να πάρουμε σχέδιο που διαμορφώθηκε με θερμική διαχείριση (thermal aware floorplan). Αυτό το σχέδιο μπορεί να διαμορφωθεί ανάλογα με τις παραμέτρους *(i)* της αναλογίας ορθογωνίου (Aspect Ratio),*(ii)* της διαμόρφωσης της ψήκτρας και *(iii)* του διαχωρισμού του κομματιού των καταχωρητών (Register File).

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

2.1 High-Level Synthesis

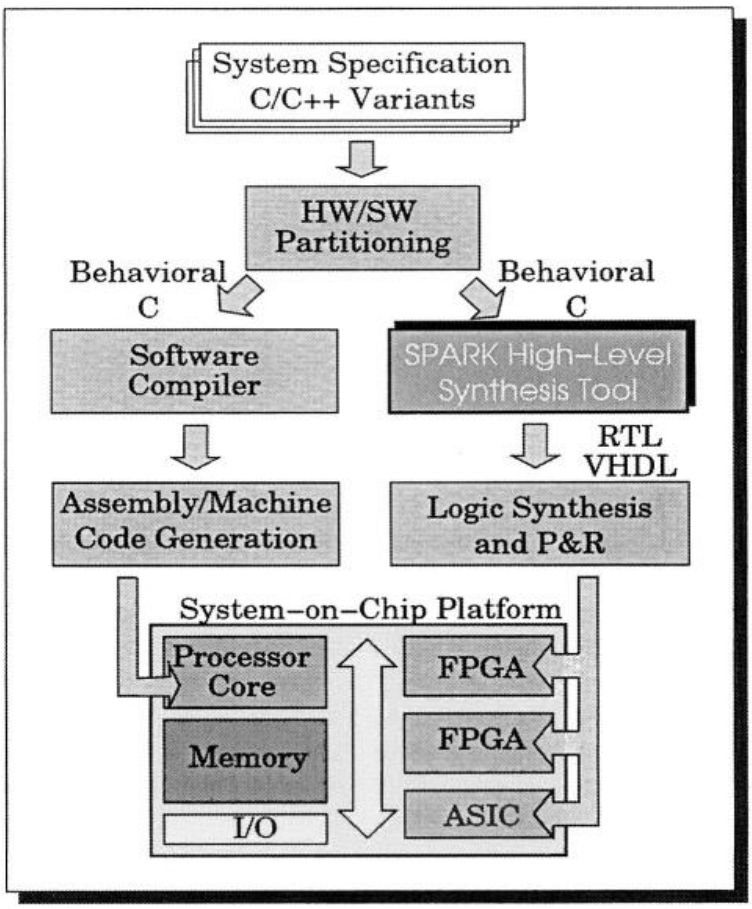
2.1.1 System-Level Design of hardware Systems

Τα μικροηλεκτρονικά συστήματα έχουν γίνει το μέσο με απεριόριστες δυνατότητες για τις ανθρώπινες προσπάθειες σε πολλούς τομείς της επιστήμης, από τους ηλεκτρονικούς υπολογιστές μέχρι και την εξερεύνηση του διαστήματος. Αυτά τα συστήματα βρίσκονται σε όλους τους τομείς της καθημερινής μας ζωής, ενσωματωμένα σε όλες τις σύγχρονες συσκευές, μέχρι και στο ανθρώπινο σώμα. Οι πρόοδοι που έχουν γίνει στη μικροηλεκτρονική τεχνολογία επιτρέπουν την ολοκλήρωση των αυξανόμενων υπολογιστικών και επικοινωνιακών δυνατοτήτων σε ένα μόνο ψηφίδα υλικού. Εντούτοις, η αύξηση της τεχνολογικής προόδου έχει ξεπεράσει κατά πολύ τη ικανότητα των σχεδιαστών να χρησιμοποιούν τη διάταξη και ταχύτητα που είναι πλέον διαθέσιμη στα κυκλώματα. Επιπλέον οι μεγάλες και συνεχείς αλλαγές στα πρωτόκολλα, στις εφαρμογές αλλά και στις ανάγκες των καταναλωτών για τα ενσωματωμένα συστήματα έχουν δημιουργήσει την ανάγκη για σχεδίαση, σύνθεση και επέκταση αυτών των συστημάτων σε μικρό χρονικό διάστημα. Αποτέλεσμα αυτών των αναγκών είναι η δυνατότητα που έχουμε σήμερα για να δημιουργούμε μικροηλεκτρονικά κυκλώματα τα οποία εκτελούν μεγάλες και πολύπλοκες εφαρμογές. Για να διαχειριστούμε το αναπτυσσόμενο μέγεθος και πολυπλοκότητα αυτών των μικροηλεκτρονικών συστημάτων και την αυξανόμενη σχεδιαστική παραγωγικότητα, είναι ανάγκη να γίνει μοντελοποίηση, σύνθεση και επικύρωση ορθής λειτουργίας σε ψηλότερα αφαιρετικά επίπεδα. Οι σχεδιαστές επιζητούν την συνδυασμένη περιγραφή του υλικού και του λογισμικού ενός ενσωματωμένου συστήματος από το επίπεδο συστήματος. Λόγω αυτών των αναγκών έγινε επιτακτική η ανάγκη για την εισαγωγή μίας καινούργιας έννοιας, τη σύνθεση υψηλού επιπέδου. Ο ορισμός της φαίνεται παρακάτω:

> Η σύνθεση υψηλού επιπέδου είναι η αυτόματη δημιουργία του υλικού κυκλώματος ενός ψηφιακού συστήματος από μία περιγραφή συμπεριφοράς.

Η σύνθεση υψηλού επιπέδου είναι ένα σημαντικό κομμάτι για τη μεθοδολογίας σχεδίασης στο επίπεδο συστήματος. Όπως φαίνεται και στο παρακάτω σχήμα 2.1, η είσοδος για αυτό τη μεθοδολογία είναι ο προσδιορισμός της εφαρμογής σε γλώσσα υψηλού επιπέδου (C, C++) και ως έξοδος λαμβάνεται μία υλοποίηση της εφαρμογής σε πλατφόρμα σύστημα-σε-κύκλωμα(system-on-a-chip,SOC).

Αρκετές γλώσσες σχεδίασης για επίπεδο συστήματος έχουν προταθεί οι οποίες αποτελούν παραλλαγές της C και της C++. Αυτές οι γλώσσες επιχειρούν να δώσουν ενοποιημένη έννοια για τον καθορισμό υλικού και λογισμικού.
Όπως φαίνεται και στο Σχήμα 2.1 το λογισμικό κομμάτι μπορεί να μεταγλωττιστεί για το πυρήνα του επεξεργαστή και το υλικό μέρος να περάσει από σύνθεση με εργαλείο σύνθεσης υψηλού επιπέδου, ακολουθούμενο από τη λογική σύνθεση και εργαλεία τοποθέτησης και δρομολόγησης.



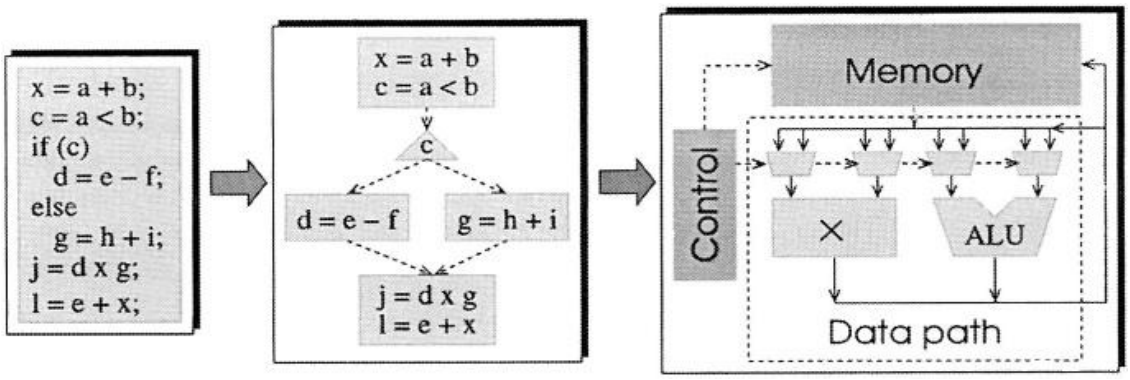
Σχήμα 2.1: Μεθοδολογία για σχεδίαση σε επίπεδο συστήματος η οποία ενσωματώνει σύνθεση υψηλού επιπέδου. [Gupt04]

2.1.2 Επισκόπηση Σύνθεσης από Υψηλό Επίπεδο (High Level Synthesis, HLS)

Η σχεδίαση υψηλού-επιπέδου (High-Level Synthesis, HLS) είναι η διαδικασία για τη μετατροπή της συμπεριφορικής (behavioural) περιγραφής σε ψηφιακό κύκλωμα το οποίο αποτελείται από το διάδρομο δεδομένων (datapath), τον κομμάτι ελέγχου (controller) και τα στοιχεία της μνήμης (memory elements). Αυτή η διαδικασία φαίνεται και στο παρακάτω σχήμα:

Αρχικά μια high-level synthesis ροή αναπαριστά την αφαιρετική συμπεριφορά σε μία ενδιάμεση αντιστοίχηση που περιέχει και τη ροή ελέγχου αλλά και τη ροή δεδομένων. Το πρόβλημα της σύνθεσης υψηλού επιπέδου συνήθως αντιμετωπιζόταν με τη διαίρεση του προβλήματος σε διάφορους στόχους.

- **Κατανομή (Allocation):** Αυτό το μέρος αφορά τον καθορισμό του αριθμού των μονάδων (resources) που έχουν κατανεμηθεί για τη σύνθεση του υλικού κυκλώματος. Συνήθως οι σχεδιαστές μπορούν να καθορίσουν την κατανομή όσον αφορά τον αριθμό για κάθε τύπο μονάδων, οι οποίες αποτελούνται όχι μόνο από λειτουργικές μονάδες (functional units, FUs), όπως μονάδες πρόσθεσης και πολλαπλασιασμού, αλλά και από καταχωρητές και στοιχεία της διασύνδεσης, όπως πολυπλέκτες και διαύλους
- **Χρονοδρομολόγηση (Scheduling):** Το πρόβλημα για τη χρονοδρομολόγηση είναι



Σχήμα 2.2: Επισκόπηση της Σύνθεσης Υψηλού επιπέδου. [Gupt04]

ο καθορισμός του χρονικού διαστήματος ή κύκλους ρολογιού στους οποίους γίνεται η εκτέλεση της κάθε λειτουργίας της σχεδίασης. Η σειρά μεταξύ των χρονοδρομολογημένων λειτουργιών είναι περιορισμένη από τις εξαρτήσεις των δεδομένων (και πιθανώς του ελέγχου) που υπάρχουν μεταξύ αυτών. Η χρονοδρομολόγηση συχνά γίνεται υπό τους περιορισμούς που υπάρχουν στον αριθμό των μονάδων (όπως καθορίζεται στη κατανομή των μονάδων).

- **Module Selection:** Άλλος ένας στόχος του προβλήματος αποτελεί η επιλογή module, δηλαδή η επιλογή του τύπου της μονάδας στην οποία μια λειτουργία θα εκτελεστεί, από τη βιβλιοθήκη των μονάδων (resources library). Η ανάγκη για αυτό το κομμάτι δημιουργείται από την παρουσία πολλών μονάδων, διαφορετικών τύπων (και διαφορετικής επιφάνειας υλικού και χρόνου) στις οποίες μία λειτουργία μπορεί να εκτελεστεί. Για παράδειγμα μία πρόσθεση μπορεί να εκτελεστεί σε ένα αθροιστή, σε μια ALU ή σε μια multiply-accumulate μονάδα Για κάθε μονάδα υπάρχει και η αντίστοιχη επιφάνεια υλικού, απόδοση και κατανάλωση ισχύος, παράγοντες που καθορίζουν την επιλογή της μονάδας. Έτσι η επιλογή module πρέπει να κάνει το πιο συνετό συνδυασμό μεταξύ διαφορετικών μονάδων έτσι ώστε ένας παράγοντας, όπως η επιφάνεια υλικού ή ο χρόνος να ελαχιστοποιηθούν, ή να φτάσουμε σε μια ισορροπημένη κατάσταση μεταξύ των διαφορετικών παραγόντων.

- **Σύνδεση (Binding):** Το κομμάτι του binding καθορίζει την απεικόνιση μεταξύ των λειτουργιών, μεταβλητών, μεταφοράς δεδομένων και ελέγχου μέσα στη σχεδίαση (design) και συγκεκριμένων μονάδων μέσα από την κατανομή των μονάδων. Με αυτό τον τρόπο οι λειτουργίες απεικονίζονται σε συγκεκριμένες λειτουργικές μονάδες, οι μεταβλητές σε καταχωρητές και η μεταφορά δεδομένων/ελέγχου σε στοιχεία της διασύνδεσης.

- **Παραγωγή ελέγχου και βελτιστοποίηση (Control Generation and Optimization):** Η σύνθεση ελέγχου παράγει μία μονάδα ελέγχου η οποία υλοποιεί τη χρονοδρομολόγηση. Αυτή η μονάδα ελέγχου παράγει σήματα ελέγχου που καθορίζουν τη ροή των δεδομένων μέσα από το datapath (π.χ μέσω των πολυπλεκτών). Η βελτιστοποίηση του ελέγχου στοχεύει στη μείωση του μεγέθους της μονάδας ελέγχου και συνεπώς και την βελτίωση στα μέτρα της επιφάνειας υλικού και ισχύος.

Όλοι οι παραπάνω στόχοι συνδέονται μεταξύ τους και είναι αλληλοεξαρτώμενοι.

2.1.3 Role of Parallelizing Compiler Transformations in HLS

Στο επίκεντρο της προσοχής βρίσκεται η σύνθεση των υπολογιστικά απαιτητικών κομματιών από εφαρμογές επεξεργασίας πολυμέσων και εικόνας και το χαμηλό latency των λειτουργικών

μονάδων από τους μικροεπεξεργαστές. Αυτές οι εφαρμογές συνήθως αποτελούνται από αριθμητικές λειτουργίες ενσωματωμένες μέσα σε φωλιασμένους βρόγχους με πολύπλοκες conditional (if-then-else) δομές. Η επιλογή αυτών των δομών ελέγχου (if-then-else, loops) έχει δραματική επίπτωση στην ποιότητα του αποτελέσματος από τη σύνθεση του υλικού. Η μοντελοποίηση σε επίπεδο γλώσσας των ψηφιακών συστημάτων (ειδικά στις μεθοδολογίες σχεδίασης επιπέδου συστήματος) δίνουν επιπλέον ελευθερία στον τρόπο περιγραφής συμπεριφοράς σε σχέση με τις περιγραφές επιπέδου καταχωρητών (Register-Transfer Level, RTL). Η σειρά και η τοποθέτηση των λειτουργιών σε συμπεριφορά behavioural υψηλού επιπέδου συνήθως διαχειρίζεται από το την προγραμματιστική ευκολία και ποικίλει από σχεδιαστή σε σχεδιαστή. Πολύ συχνά αυτή η σειρά συνήθως δε συμβάλλει στην προς τα κάτω high-level synthesis ούτε βελτιστοποιεί του στόχους.

Ένα εύρος από μετασχηματισμούς παραλληλοποίησης έχει εξερευνηθεί στο επίπεδο του μεταγλωττιστή για να εξαλείψει τα προβλήματα με τον κώδικα και τη πολύπλοκη ροή ελέγχου. Έχει βρεθεί πως αυτοί οι μετασχηματισμοί παραλληλοποίησης είναι εξίσου αναγκαίοι για τη σύνθεση υψηλού-επιπέδου για behavioural συμπεριφορές με φωλιασμένα conditionals και βρόγχους. Έτσι αναπτύχθηκε μία μεθοδολογία που εφαρμόζει ένα συντεταγμένο πακέτο από coarse-grained και fined-grained μεταγλωττιστή παραλληλοποίησης, τον κύριο μεταγλωττιστή και μετασχηματισμών σύνθεσης, που μέσω της εύρεσης και αύξησης της παραλληλοποίησης που υπάρχει στην αλγοριθμική περιγραφή, στοχεύει στη βελτίωση των αποτελεσμάτων της σύνθεσης από τους κώδικες συμπεριφοράς (behavioural codes). Αυτή η μεθοδολογία στοχεύει να βελτιώσει την ποιότητα πολλών πτυχών του αποτελέσματος που προκύπτει από τη σύνθεση υψηλού επιπέδου αλλά και να δώσει στο σχεδιαστή τον έλεγχο για τους μετασχηματισμούς που εφαρμόζονται. Υιοθετούνται μετασχηματισμοί και τεχνικές που εκμεταλλεύονται την παραλληλία σε επίπεδο εντολών, όπως θεωρητικές "κινήσεις" κώδικα (code motions), "εσωτερική" χρονοδρομολόγηση, και μετασχηματισμούς σε βρόγχους που αφορούν το coarse-grained μέρος. Οι μετασχηματισμοί στους βρόγχους προσπαθούν να εκμεταλλευτούν την παραλληλία στο επίπεδο των εσωτερικών επαναλήψεων (διεύρυνση και διοχέτευση του βρόγχου - loop unrolling and loop pipelining) ή την παραλληλία στους βρόγχους (loop fusion and loop interchange). Μία άλλη κλάση μετασχηματισμών βρόγχων επιχειρεί είτε να μειώσει τον αριθμό των λειτουργιών που εκτελούνται (loop invariant code motion) είτε να μειώσει την "ένταση" των λειτουργιών που εκτελούνται: για παράδειγμα να αντικαταστήσει τις υπολογιστικά απαιτητικές λειτουργίες όπως πολλαπλασιασμούς με προσθέσεις χρησιμοποιώντας τεχνικές "αποδυνάμωσης" (strengh reduction techniques) και επαγωγικής ανάλυσης μεταβλητών (induction variable analysis).

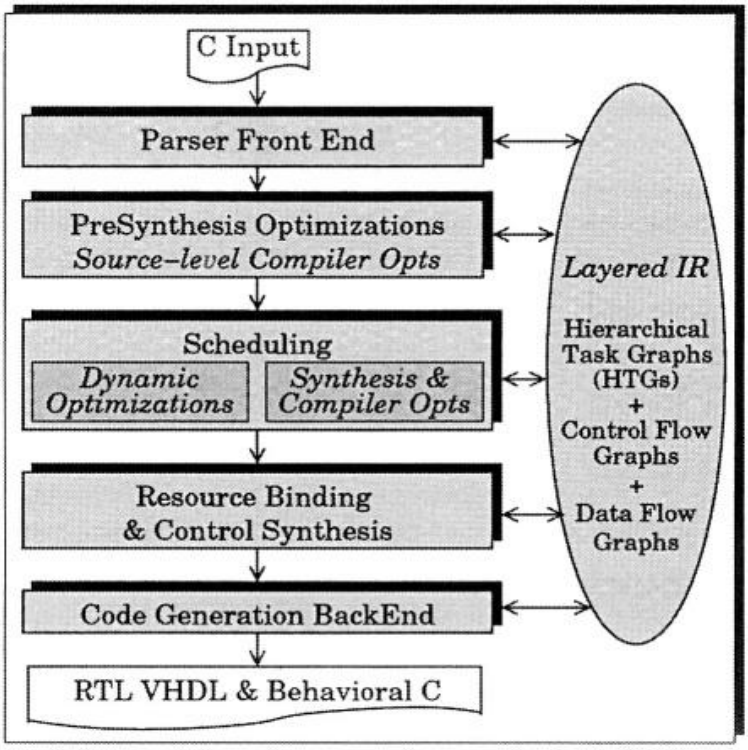
Οι περισσότεροι μετασχηματισμοί συνδέονται με βασικούς μετασχηματισμούς μεταγλωττιστή, όπως αποφυγή κοινών υπο-εκφράσεων (common sub-expression elimination), διάδοση αντιγραφής (copy propagation), διαγραφή "νεκρού" κώδικα (dead-code elimination) και άλλους οι οποίοι είναι αναγκαίοι για τη βελτίωση των αποτελεσμάτων από το high-level synthesis - ειδικά όταν συνθέτουμε με γλώσσες υψηλού επιπέδου, όπως behavioural VHDL, C, C++, οι οποίες δίνουν μεγάλη ελευθερία στον προγραμματισμό.

Άλλο ένα σημαντικό μέρος σ' αυτήν την προσέγγιση για high-level synthesis είναι οι εφαρμογές με speculative code-motions οι οποίες μεταφέρουν λειτουργίες μεταξύ conditionals και βρόγχων. Αυτές βασίζονται στην χρονική κρισιμότητα μιας λειτουργίας και στη διαδικασία φανερώνουν την παραλληλία που υπάρχει στον αλγόριθμο.

2.1.4 SPARK© High-Level Synthesis Methodology

SPARK© high-level synthesis methodology ενσωματώνει μεταγλωττιστή παραλληλοποίησης και μετασχηματισμούς μέσα σε ένα πλαίσιο HLS, στις φάσεις της "προ-σύνθεσης" και στο στάδιο

της χρονοδρομολόγησης. Αυτή η μεθοδολογία δέχεται σαν είσοδο την περιγραφή της σχεδίασης σε γλώσσα υψηλού επιπέδου, όπως η γλώσσα C, την οποία και "συλλαμβάνει" χρησιμοποιώντας μια ενδιάμεση αναπαράσταση που διατηρεί τη δομή του κώδικα μαζί με διάφορες άλλες πληροφορίες όπως τις μεταβλητές που χρησιμοποιούνται στους βρόγχους.



Σχήμα 2.3: Επισκόπηση του HLS που χρησιμοποιεί το SPARK© η οποία ενσωματώνει του μετασχηματισμούς στο μεταγλωττιστή. [Gupt04]

Κατ' αρχάς εφαρμόζεται μια σειρά από μετασχηματισμούς σε επίπεδο κώδικα στην περιγραφή της εισόδου για τη σχεδίαση στη φάση "προ-σύνθεση". Αυτοί οι μετασχηματισμοί, όπως αποφυγή κοινών υπο-εκφράσεων (common sub-expression elimination), διάδοση αντιγραφής (copy propagation), διαγραφή "νεκρού" κώδικα (dead-code elimination) και σταθερών βρόγχων code motion, στοχεύουν να μειώσουν τον αριθμό τον λειτουργιών που εκτελούνται και να μειώσουν τον περιττό κώδικα. Επίσης χρησιμοποιούνται coarse-level τεχνικές για μετασχηματισμούς βρόγχων όπως διεύρυνση του βρόγχου (loop unrolling) για την αναδόμηση του κώδικα. Αυτό αυξάνει το πεδίο για να εφαρμοστούν βελτιστοποιήσεις στην παραλληλοποίηση στο κομμάτι χρονοδρομολόγησης που ακολουθεί.

Η χρονοδρομολόγηση υιοθετεί ένα καινοτόμο πακέτο από "κινήσεις" κώδικα (code motions), που μειώνουν την επίδραση των συνταχτικών διαφορών ή του προγραμματιστικού είδους στο αποτέλεσμα της σύνθεσης. Αυτά τα code-motions επιτρέπουν τη μετακίνηση λειτουργιών διαμέσου, πέρα και μέσα σε conditionals με στόχους τη μεγιστοποίηση της απόδοσης με την εξαγωγή της κληρονομούμενης παραλληλίας στη σχεδίαση και την αύξηση της χρησιμοποίησης των μονάδων. Δεδομένου ότι αυτές οι θεωρητικές "κινήσεις" κώδικα, συχνά αναδιατάσσουν, θεωρούν και αναπαράγουν λειτουργίες, δημιουργούν με αυτό το τρόπο νέες δυνατότητες για να εφαρμοστούν πρόσθετοι μετασχηματισμοί "δυναμικά" κατά τη διάρκεια της χρονοδρομολόγησης όπως η δυναμική διαγραφή κοινών υπό-εκφράσεων (common sub-expression). Η δυναμική εξισορρόπηση κλάδων (branch balancing) είναι άλλος ένας μετασχηματισμός που εφαρμόζουμε κατά

τη διάρκεια της χρονοδρομολόγησης για να επιτρέψουμε τις κινήσεις κώδικα (code motions), ειδικά αυτών που περιλαμβάνουν διπλασιασμό κώδικα (code duplication). Αυτοί οι μετασχηματισμοί μεταγλωττιστή είναι ενσωματωμένοι με τις τυπικές τεχνικές σύνθεσης υψηλού επιπέδου (HLS) όπως η κοινή εκμετάλλευση πόρων (resource sharing), χρονοδρομολόγηση σε λειτουργίες πολλών-κύκλων (multi-cycle) και τη διασύνδεση λειτουργιών ανάμεσα στους κύκλους εκτέλεσης (operation chaining).

Η προ-σύνθεση και η χρονοδρομολόγηση μετασχηματισμών οδηγούν σε σημαντικά μικρότερα μήκη χρονοδρομολόγησης και μεγαλύτερη χρησιμοποίηση των πόρων. Εντούτοις, αυτές οι τεχνικές οδηγούν σε μια αύξηση της πολυπλοκότητας της "λογικής οδήγησης" (steering logic) στη σχεδίαση, δηλαδή στους πολυπλέκτες και στη συνδυαζόμενη λογική ελέγχου. Σημειώνεται ότι η "λογική οδήγησης" αναφέρεται και ως διασύνδεση (interconnect).

Η πολυπλοκότητα της διασύνδεσης ελαχιστοποιείται με τη βοήθεια μεθοδολογίας δέσμευσης πόρων (resource binding) η οποία αρχικά κάνει δέσμευση λειτουργιών (operation binding) και έπειτα δέσμευση μεταβλητών (variable binding). Οι λειτουργίες δεσμεύονται στις λειτουργικές μονάδες (FUs) με τέτοιο τρόπο ώστε αυτές που έχουν ίδια δεδομένα εισόδου ή εξόδου να δεσμεύονται στις ίδιες λειτουργικές μονάδες (FUs). Ομοίως, οι μεταβλητές δεσμεύονται στους καταχωρητές ώστε αυτές που αποτελούν είσοδο ή έξοδο για ίδιες λειτουργικές μονάδες (FUs) να απεικονίζονται στους ίδιους καταχωρητές.

Στη συνέχεια μια σύνθεση ελέγχου και βελτιστοποίηση παράγει παράγει μηχανή πεπερασμένης κατάστασης (FSM) για τη χρονοδρομολογημένη και συνδεδεμένη σχεδίαση. Αυτός ο ελεγκτής εκτελεί τις λειτουργίες σύμφωνα με το συγχρονισμό που διευκρινίζεται από το πρόγραμμα και παράγει τα σήματα ελέγχου για να καθοδηγήσει τα δεδομένα μέσω της διασύνδεσης όπως διευκρινίζεται από τη σύνδεση των πόρων. Για να ολοκληρώσει την πορεία από τη συμπεριφορική (behavioural) προδιαγραφή στο επίπεδο net-list (δίκτυο πυλών), η μεθοδολογία αυτή χρησιμοποιεί μια γεννήτρια κώδικα που μπορεί να αλληλεπίδραση με τα εργαλεία σύνθεσης τυποποιημένης λογικής. Αυτό επιτρέπει την αξιολόγηση των αποτελεσμάτων διάφορων coarse και fined-grain (αδρομερής και λεπτομερής) βελτιστοποιήσεων στα αποτελέσματα σύνθεσης λογικής.

Η μεθοδολογία HLS που έχει περιγραφεί υλοποιείται στο εργαλείο SPARK, ένα εργαλείο για high-level synthesis από κώδικα C σε VHDL. Το εργαλείο SPARK διαθέτει μια εργαλειοθήκη από μετασχηματισμούς και ένα σετ από αλγόριθμους που τους υιοθετούν. Μια τέτοια προσέγγιση της εργαλειοθήκης επιτρέπει στο σχεδιαστή να εφαρμόσει ευριστική προσέγγιση (heuristics) για να καθορίσει την επιλογή και τον έλεγχο μεμονωμένων μετασχηματισμών κάτω από μοντέλα ρεαλιστικού κόστους για high-level synthesis. Το εργαλείο σύνθεσης SPARK αποτελεί ένα πλήρες σύστημα για σύνθεση υψηλού επιπέδου το οποίο παρέχει τη "διαδρομή" από μια συμπεριφορική (behavioural) περιγραφή εισόδου χωρίς περιορισμούς κάτω σε κώδικα επιπέδου καταχωρητών (register-transfer level, RTL). Ο παραγόμενος RTL κώδικας μπορεί έπειτα να συντεθεί χρησιμοποιώντας τα εμπορικά εργαλεία σύνθεσης λογικής.

Το SPARK επιτρέπει τον πειραματισμό με μεγάλα "πραγματικά" σχέδια και και το πιο σημαντικό επιτρέπει μια ανάλυση του αντίκτυπου των μετασχηματισμών στο κόστος του ελέγχου και επιφάνειας του κυκλώματος στο επίπεδο πυλών. Δίνεται στο σχεδιαστή τη δυνατότητα να ελέγξει τους μετασχηματισμούς που εφαρμόζονται στην περιγραφή της σχεδίαση με τη χρήση χειρόγραφου κώδικα (scripts). Τα γραφικά αποτελέσματα βοηθούν στην απεικόνιση των ενδιάμεσων αποτελεσμάτων. Τα αποτελέσματα από τα πειράματα που εκτελούνται στις σύνθετες βιομηχανικές εφαρμογές, από εφαρμογές πολυμέσων και επεξεργασίας εικόνας επικυρώνουν τη χρησιμότητα αυτής της προσέγγισης.

2.1.5 Μοντελοποίηση και αναπαράσταση

Μοντελοποίηση του προβλήματος

Διάφορα μοντέλα χρησιμοποιούνται για την αναπαράσταση των πληροφοριών που απαιτούνται και παράγονται κατά τη διάρκεια των μετασχηματισμών του κώδικα (code transformations) και στο κομμάτι της χρονοδρομολόγησης στη σύνθεση υλικού υψηλού-επιπέδου (HLS). Μια "ενδιάμεση αναπαράσταση" (Intermediate Representation - IR) απαιτείται για να αναπαραστήσει την περιγραφή του σχεδιασμού. Στο πρόβλημα χρονοδρομολόγησης χρησιμοποιείται περιορισμός των λειτουργικών μονάδων για την εύρεση της ελάχιστης καθυστέρησης (minimum-latency resource-constrained scheduling problem). Στη συνέχεια η χρονοδρομολόγηση γίνεται λαμβάνοντας υπόψη και την παρουσία της ροής ελέγχου. Επίσης μοντελοποιούνται θεωρητικές και ιεραρχικές κινήσεις κώδικα (code motions).
Το κομμάτι της χρονοδρομολόγησης παράγει διάφορα είδη πληροφορίας όπως:

- Ενδιάμεση αναπαράσταση (Intermediate Representation IR): χρειάζεται για να αναπαραστήσει την περιγραφή σχεδιασμού.
- Περιγραφή Υλικού (Hardware Representation): Ο καθορισμός και η αναπαράσταση των λειτουργικών μονάδων και πόρων όπως καταχωρητές, διαύλους και άλλα που διατίθενται για τη χρονοδρομολόγηση του σχεδιασμού.
- Πληροφορίες Χρονισμού (Timing Information): Περίοδος ρολογιού, χρόνοι εκτέλεσης των διαφόρων λειτουργικών μονάδων και πληροφορίες που παράγονται από τη χρονοδρομολόγηση όπως τους χρόνους εκτέλεσης των λειτουργιών.
- Αντιστοίχηση πόρων (Resource Mapping): Η αντιστοίχηση κάθε λειτουργίας σε ένα πόρο για την εκτέλεση της. Η σύνδεση των πόρων καθορίζει την αντιστοίχηση κάθε λειτουργίας σε ένα συγκεκριμένο στιγμιότυπο του τύπου πόρων στο οποίο δρομολογήθηκε.

Μοντελοποίηση της περιγραφής σχεδιασμού

Για τη μεθοδολογία της σύνθεσης χρησιμοποιείται η γλώσσα ψηλού επιπέδου C. Οι σχεδιαστές προτιμούν τις γλώσσες υψηλού επιπέδου όπως η "C" αντί της γλώσσες περιγραφής συμπεριφοράς υλικού (HDLs) αφού δίνουν στο σχεδιαστή περισσότερη ελευθερία για την περιγραφή μιας συμπεριφοράς και επίσης, επιτρέπει τη διευκρίνιση μιας συμπεριφοράς χωρίς λεπτομέρειες υλοποίησης υλικού και λογισμικού. Αυτό είναι μια σημαντική προϋπόθεση των προδιαγραφών επιπέδων του συστήματος δεδομένου ότι οι στόχοι στην προδιαγραφή δεν έχουν χωριστεί ακόμα στα τμήματα υλικού και λογισμικού.

Η περιγραφή εισόδου «C» στη μεθοδολογία μας είναι ένας διαδοχικός κατάλογος δηλώσεων. Οι δηλώσεις μπορούν να είναι εκφράσεις λειτουργίας, δομές if-then-else και switch-case και δομές βρόχων (for, while, do-while). Εκτός από τις διαδικασίες που υποστηρίζονται από τη «C», υποστηρίζονται επίσης Boolean ελέγχους. Αυτοί οι έλεγχοι Boolean παράγονται με τη σύγκριση ή σχεσιακούς ελέγχους όπως $<, ==, \geq, \neq$ κτλ. . Οι σύνθετες εκφράσεις αποσυνθέτονται σε εκφράσεις τριών διευθύνσεων (του τύπου $a = b + c$). Κάθε έκφραση τριών-διευθύνσεων καλείται λειτουργία και αντιπροσωπεύεται από ένα αφηρημένο δέντρο σύνταξης.

Η ροή ελέγχου και η ροή δεδομένων καταγράφεται στην περιγραφή εισόδου χρησιμοποιώντας ένα γράφο ροής ελέγχου (Control Flow Graph - CFG) και ενός γράφου ροής δεδομένων (Data Flow Graph - DFG). Επίσης καταγράφεται η δομή του προγράμματος στην περιγραφή εισόδου χρησιμοποιώντας μια ιεραρχική ενδιάμεση περιγραφή που ονομάζεται γράφος ιεραρχικοποίησης στόχων (Hierarchical Task Graph - HTG). Οι κόμβοι αυτών των τριών γράφων διαμορφώνουν

ένα γράφο με 3 επίπεδα έτσι ώστε υπάρχει μια σχέση μεταξύ των κόμβων κάθε διαδοχικού επιπέδου. Οι γράφοι με επίπεδα ορίζονται ως εξής:

Ορισμός 2.1. *Ενας γράφος κ-επιπέδων είναι ένας συνδεδεμένος γράφος στον οποίον οι κόμβοι (vertices) χωρίζονται σε κ-σύνολα* $L = L_1, ..., L_k$ *και οι ακμές εκτείνονται μεταξύ των κόμβων διαδοχικών επιπέδων* i_i *και* I_{i-1}.

Σ' αυτή την περίπτωση το πάνω επίπεδο αποτελείται από κόμβους από τον γράφος ιεραρχικοποίησης στόχων (HTG), οι κόμβοι στο επόμενο επίπεδο από το γράφο ροής ελέγχου (CFG) και στο κατώτερο επίπεδο κόμβοι από το γράφο ροής δεδομένων (DFG).

Μοντελοποίηση των εξαρτήσεων δεδομένων Υπάρχουν τέσσερις τύποι εξαρτήσεων δεδομένων που μπορούν να υπάρξουν μεταξύ των λειτουργιών op_i και op_j:

1. εξαρτήσεις ροής ή διάβασμα μετά από γράψιμο (read-after-write): Η λειτουργία op_j διαβάζει το αποτέλεσμα που γράφεται από τη λειτουργία op_i.

2. εξαρτήσεις αντί ή γράψιμο μετά από διάβασμα (write-after-read): Η λειτουργία op_j γράφει σε μια μεταβλητή μετά που διαβάζεται από τη λειτουργία op_i.

3. εξαρτήσεις εξόδου ή γράψιμο μετά από γράψιμο (write-after-write): Η λειτουργία op_j γράφει στην ίδια μεταβλητή μετά που γράφεται από τη λειτουργία op_i.

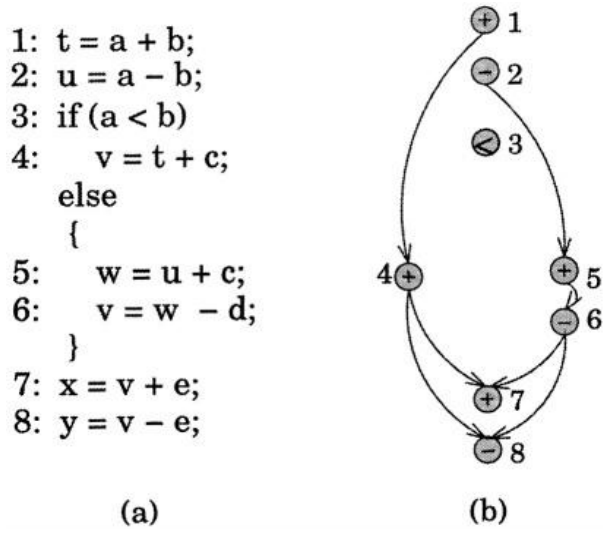
4. εξαρτήσεις εισόδου ή διάβασμα μετά από διάβασμα (read-after-read): Η λειτουργία op_j διαβάζει μια μεταβλητή μετά που διαβάζεται από τη λειτουργία op_i.

Από αυτές, οι εξαρτήσεις εισόδου (read-after-read) δεν επηρεάζουν τη χρονοδρομολόγηση του σχεδίου.
Για να διατυπώσουμε το πρόβλημα σχεδιασμού, χρησιμοποιούμε μόνο τις εξαρτήσεις ροής δεδομένων. Μια εξάρτηση ροής δεδομένων καθορίζει πότε μια λειτουργία μπορεί να αρχίσει την εκτέλεση - εάν η λειτουργία op_j έχει μια εξάρτηση ροής με την op_i, τότε η op_j μπορεί να αρχίσει να εκτελείται μόνο αφού έχει τελειώσει η εκτέλεση της op_i. Μπορούμε να καθορίσουμε ένα γράφο ροής δεδομένων που καταγράφει τις εξαρτήσεις ροής ως εξής:

Ορισμός 2.2. *Ο* ***γράφος ροής δεδομένων*** *είναι ένας κατευθυνόμενος ακυκλικός γράφος* $G_{DFG}(Ops, E_{data})$ *όπου το σύνολο των κόμβων* $Ops = \{op_i, i = 1, ..., n_{ops}\}$ *είναι το σύνολο από* n_{ops} ***λειτουργιών*** *στο σχέδιο και το σύνολο των ακμών* $E_{data} = \{(op_i, op_j); i, j = 1, ..., n_{ops}\}$ *αποτελεί τις εξαρτήσεις ροής δεδομένων. Μια κατευθυνόμενη ακμή* $e_{ij} = (op_i, op_j)$ *υπάρχει στο σύνολο των ακμών* E_{data} *εάν τα δεδομένα που παράγονται από τη λειτουργία* op_i *διαβάζονται από τη λειτουργία* op_j.

Στο σχήμα 2.4(b) παρουσιάζεται ο γράφος ροής δεδομένων που αντιστοιχεί στο κώδικα C του σχήματος 2.4(α). Οι λειτουργίες στο γράφο ροής δεδομένων (DFG) δείχνονται από τους κυκλικούς κόμβους με το σύμβολο της λειτουργίας μέσα σ' αυτούς. Οι αριθμοί στο DFG αντιστοιχούν στους αριθμούς των γραμμών του κώδικα C. Η έκφραση σε κάθε κόμβο λειτουργίας αποθηκεύεται ως αφηρημένο δέντρο σύνταξης.

Βελτίωση της απεικόνισης της σχεδίασης με τη διατήρηση των ονομάτων των μεταβλητών Η διατήρηση μόνο των εξαρτήσεων ροής δεδομένων σημαίνει ότι οι πληροφορίες για τη μεταβλητή τα ονόματα από την αρχική περιγραφή απορρίπτονται. Αυτό εξασθενίζει τη δυνατότητα να συσχετίσει η περιγραφή εισόδου την ενδιάμεση αντιπροσώπευση και τον τελικό κώδικα εξόδου που παράγονται μετά από τη σύνθεση. Αυτό, στη συνέχεια, εμποδίζει την απεικόνιση των αποτελεσμάτων της εφαρμογής μετασχηματισμών στο σχέδιο έναντι της αρχικής περιγραφής εισόδου.



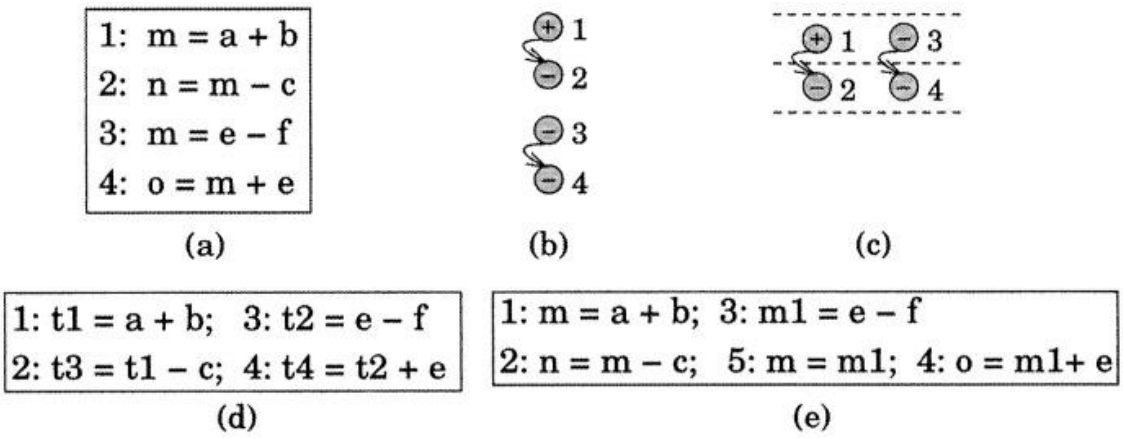
Σχήμα 2.4: (a) Παράδειγμα περιγραφής σε C και (b) Ο αντίστοιχος γράφος ροής δεδομένων (DFG). [Gupt04]

Στο σχήμα 2.5(a) φαίνεται ένα παράδειγμα περιγραφής σε C κώδικα και στο 2.5(b) ο αντίστοιχος γράφος ροής δεδομένων (DFG). Μια πιθανή χρονοδρομολόγηση φαίνεται στο σχήμα 2.5(c), όπου οι λειτουργίες 1 και 3 και οι λειτουργίες 2 και 4 ταυτόχρονα. Ο αντίστοιχος κώδικας εξόδου σ'αυτή τη χρονοδρομολόγηση της σχεδίασης, στον οποίο μόνο οι εξαρτήσεις ροής δεδομένων διατηρούνται παρουσιάζεται στο σχήμα 2.5(d). Σ' αυτό τον κώδικα εξόδου, είναι αναγκαίο να δημιουργηθούν καινούργιες μεταβλητές στις οποίες θα αποθηκευτεί το αποτέλεσμα κάθε λειτουργίας στο χρονοδρομολογημένο σχέδιο γεγονός το οποίο καταντά δύσκολο να συσχετιστούν οι δηλώσεις των λειτουργιών αυτού του κώδικα εξόδου με τις δηλώσεις λειτουργιών του κώδικα εισόδου.
Αν διατηρηθούν οι εξαρτήσεις δεδομένων εκτός-ροής, μπορεί να παραχθεί ο κώδικας εξόδου που παρουσιάζεται στο σχήμα 2.5(e). Σ' αυτό τον κώδικα, οι μεταβλητές στις οποίες γράφει η κάθε λειτουργία διατηρούνται όπως στον αρχικό κώδικα. Παρατηρώντας τον κώδικα, φαίνεται ότι η ταυτόχρονη εκτέλεση των λειτουργιών 1 και 3, απαιτεί την μετονομασία της μεταβλητής αποτελέσματος της λειτουργίας 3 σε $m1$. Μια λειτουργία αντιγραφής, η λειτουργία 5, από τη νέα μεταβλητή $m1$ στη μεταβλητή m από τον αρχικό κώδικα, εισάγεται όπως φαίνεται και στο σχήμα 2.5(e). Έτσι, η λειτουργία 4 μπορεί να εκτελεστεί ταυτόχρονα με τις λειτουργίες 2 και 5 χρησιμοποιώντας τη νέα μεταβλητή $m1$ έπειτα από εφαρμογή δυναμικής μετονομασίας μεταβλητών.

Επομένως, ένα πέρασμα ανάλυσης των εξαρτήσεων δεδομένων χρησιμοποιείται για να καταγράψει το πλήρες σύνολο των εξαρτήσεων δεδομένων που υπάρχουν στην περιγραφή εισόδου και τεχνικές όπως η δυναμική μετονομασία μεταβλητών χρησιμοποιούνται για να βοηθήσουν στην μείωση των περιορισμών που επιβάλλονται από αυτές τις εξαρτήσεις.

Μοντελοποίηση της ροής ελέγχου Η παρουσία δομών με όρους (conditional) και βρόγχους οδηγεί στην έννοια της ροής ελέγχου μέσω του σχεδίου. Κατά τη διάρκεια της εκτέλεσης του προγράμματος, όταν παρουσιαστεί μια λειτουργία όρου, η ροή ελέγχου διακλαδίζεται σε δύο ροές ελέγχου, βασισμένες εάν ο όρος αποτιμηθεί ως αληθής ή ψευδής. Στο τέλος της δομής όρου ή βρόγχου, οι δύο ροές ελέγχου συγκλίνουν ή συγχωνεύονται σε ένα νήμα ροής ελέγχου.

Οι δομές if-then-else, ως εκ τούτου, προσθέτουν δύο τύπους κόμβων ελέγχου στο γράφο σχεδίου, ένα κόμβος *διαχωρισμού* (*fork* node) και ένα κόμβο *ένωσης* (*join* node). Οι κόμβοι διαχω-



Σχήμα 2.5: (a) Παράδειγμα περιγραφής σε C. (b) Ο αντίστοιχος γράφος ροής δεδομένων (DFG). (c) Ο γράφος DFG μετά από τη χρονοδρομολόγηση: οι λειτουργίες 1 και 3 και οι λειτουργίες 2 και 4 εκτελούνται ταυτόχρονα. (d) Ο χρονοδρομολογημένος κώδικας εξόδου που παράγεται χωρίς να διατηρούνται οι εξαρτήσεις δεδομένων εκτός-ροής. (e) Ο κώδικας εξόδου που παράγεται αν οι εξαρτήσεις δεδομένων εκτός-ροής διατηρούνται Λειτουργίες στην ίδια γραμμή δηλώνουν την ταυτόχρονη εκτέλεση τους. [Gupt04]

ρισμού δηλώνουν το σημείο στο οποίο ένας όρος προκαλεί τη ροή ελέγχου να διακλαδωθεί σε πολλαπλά μονοπάτια ελέγχου. Αντιθέτως, κόμβοι ένωσης συγχωνεύουν σημεία των πολλαπλών μονοπατιών ροής ελέγχου. Η παρουσία ροής ελέγχου εισάγει την έννοια των βασικών κομματιών (blocks). Ένα βασικό κομμάτι είναι μια ακολουθία δηλώσεων από την περιγραφή εισόδου χωρίς conditionals ή τους βρόχους μεταξύ τους.

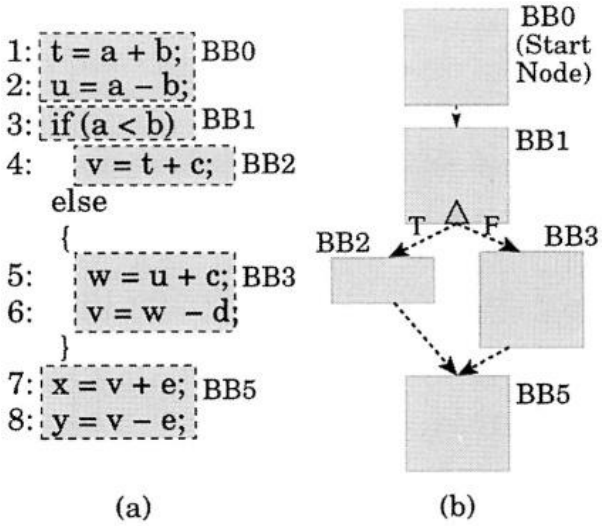
Σε μια γενική περιγραφή ενός προγράμματος σε C, η παρουσία αλμάτων στον κώδικα μπορεί να οδηγήσει σε μια αυθαίρετη ροή ελέγχου μεταξύ των βασικών κομματιών. Γι' αυτό και είναι αναγκαίος κώδικας ο οποίος είναι συνθέσιμος και οδηγεί γράφους που ανάγονται στη επιθυμητή μορφή. Αυτό σημαίνει ότι τα άλματα επιτρέπονται στον κώδικα εφόσον ο γράφος ροής ελέγχου είναι ένας αναγώγιμος γράφος. Στο μοντέλο της ροής ελέγχου, κάθε βασικό κομμάτι μπορεί να έχει δύο μονοπάτια εισόδου ελέγχου που αντιστοιχεί σε ένα κόμβο ένωσης και δύο μονοπάτια εξόδου ελέγχου που αντιστοιχούν σε ένα κόμβο διαχωρισμού. Ένα τμήμα του κώδικα με περισσότερες από δύο πορείες που προέρχονται από έναν κόμβο διαχωρισμού μπορεί να μετατραπεί σε αυτό το μοντέλο με κατ' επανάληψη χωρισμό των πολλαπλών κόμβων διαχωρισμού σε κόμβους διαχωρισμού δύο εξόδων. Παραδείγματος χάριν, μια δήλωση switch-case μπορεί να μετατραπεί στο πολλαπλά if-then-else δηλώσεις. Οι κόμβοι ένωση μπορούν ομοίως να μετατραπούν σε κόμβους ένωσης δύο εισόδων. Ένας βρόχος με τα πολλαπλάσια σημεία εξόδων μπορεί να μετατραπεί σε έναν βρόχο ενιαίας εξόδου με την παρεμβολή ενός κενού βασικού κομματιού που ενεργεί ως σημείο εξόδου για όλες τις εξόδους βρόχων.

Οι πορείες εξόδου ελέγχου ενός βασικού κομματιού έχουν έναν όρο «αληθή» ή «ψευδή» που συνδέεται με αυτούς, συνεπώς αυτές οι πορείες ελέγχου είναι γνωστές ως αληθής πορεία και ψευδής πορεία. Τα βασικά κομμάτια που δεν έχουν έναν υπό όρους έλεγχο (κανένα διαχωρισμό ροής ελέγχου), έχουν μόνο μια προκαθορισμένη αληθή πορεία εξόδου. Τα βασικά κομμάτια που είναι και τα τελευταία στο σχέδιο δεν έχουν οποιεσδήποτε πορείες ελέγχου για έξοδο. Ομοίως, το πρώτο βασικό κομμάτι στο σχέδιο (που καθορίζεται πιο κάτω) που αντιστοιχεί στο πρώτο σημείο εισόδων στο σχέδιο δεν έχει για είσοδο κάποιες πορείες ελέγχου.

Ο ορισμός για το γράφο ροής ελέγχου που καταγράφει τις πληροφορίες ροής ελέγχου μεταξύ

των βασικών κομματιών είναι ο ακόλουθος:

Ορισμός 2.3. *Ο* ***γράφος ροής ελέγχου*** *είναι είναι ένας κατευθυνόμενος γράφος* $G_{CFG}(BB, E_{control})$*, όπου το σύνολο των κόμβων* $BB = \{bb_i, i = 1, 2, ..., n_{bbs}\}$ *είναι το σύνολο από* n_{bbs} *βασικών κομματιών στο σχέδιο και το σύνολο των ακμών* $E_{control} = \{(bb_i, bb_j); i, j = 1, 2, ..., n_{bbs}\}$ *αντιστοιχεί στις ροή ελέγχου μεταξύ βασικών κομματιών. Επίσης, υπάρχει και ένα αρχικό βασικό κομμάτι* $bb_0 \epsilon BB$ *από το οποίο προέρχονται όλες οι πορείες (μονοπάτια) στο γράφο ροής δεδομένων;* bb_0 *μπορεί να ληφθεί από το* $FirstBB(G_{CFG})$*. Κάθε ακμή έχει μία ετικέτα* $L_{Cond} = \{T_{edge}, F_{edge}\}$ *με την οποία καθορίζεται εάν η ακμή είναι μία αληθής πορεία ή ψευδής πορεία. Μία κατευθυνόμενη ακμή* $e_{ij} = (bb_i, bbj)$ *υπάρχει στο σύνολο* $E_{control}$ *εάν το βασικό κομμάτι* bb_i *είναι* ***"προκάτοχο" (predecessor)*** *του βασικού κομματιού* bb_j*. Το βασικό κομμάτι* bb_j *αποτελεί το "διάδοχο" του* bb_i*.*



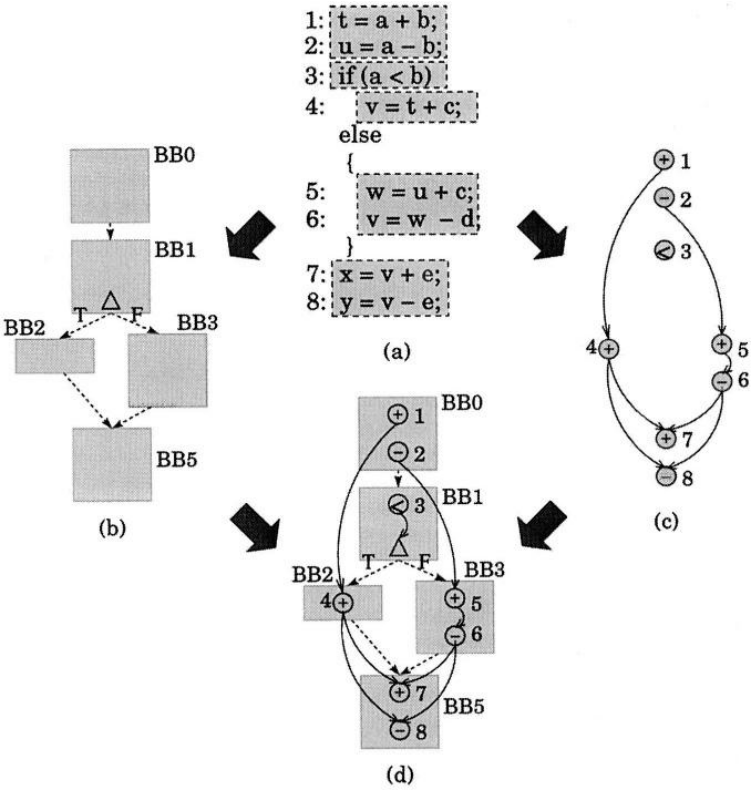
Σχήμα 2.6: (a) Παράδειγμα περιγραφής σε C και (b) Ο αντίστοιχος γράφος ροής ελέγχου (CFG). [Gupt04]

Στο σχήμα 2.6 φαίνεται η αναπαράσταση κώδικα C σε γράφο ροής ελέγχου. Κάθε ακολουθία από λειτουργίες στον πηγαίο κώδικα χωρίς κάποια ροή ελέγχου μεταξύ τους αθροίζεται σε ένα βασικό κομμάτι (όπως εμφανίζονται και με σκιασμένα κουτιά στο σχήμα 2.6(a)). Η ροή ελέγχου μεταξύ αυτών των βασικών κομματιών φαίνεται στον αντίστοιχο γράφο ροής ελέγχου στο σχήμα 2.6(b). Αυτά τα βασικά κομμάτια ονομάζονται από bb_0 μέχρι και bb_5. Ένα τρίγωνο δείχνει ένα έλεγχο συνθήκης Boolean ή ένα διαχωρισμό στη ροή ελέγχου με μία πορεία αληθή και μία πορεία ψευδή. Όλες οι άλλες ακμές ελέγχου είναι προκαθορισμένες αληθής και η ετικέτα L_{Cond} παραλείπεται.

Απεικόνιση μεταξύ ροής δεδομένων και ροής ελέγχου

Ορισμός 2.4. *Δοθέντος ενός γράφου ροής δεδομένων* $G_{DFG}(Ops, E_{data})$ *και ενός γράφου ροής ελέγχου* $G_{CFG}(BB, E_{control})$ *όπως ορίζονται στα 2.2 και 2.3 υπάρχει απεικόνιση πολλά-σε-ένα για τις λειτουργίες στα βασικά κομμάτια* $BB_{Ops} : V_{OP} \mapsto BB$*. Ένα βασικό κομμάτι στο οποίο μία λειτουργία* op_i *ανήκει μπορεί να ληφθεί από* $BB_{Ops} = bb_i \ \forall \ op_i \ \epsilon \ O_{ps}$*, όπως το* $bb_j \ \epsilon \ BB$*.*

Ο συνδυασμένος γράφος ροής δεδομένων και ελέγχου φαίνεται στο σχήμα 2.7(d) μαζί με την αρχική περιγραφή για τους γράφους ροής δεδομένων και ροής ελέγχου.



Σχήμα 2.7: (a) Παράδειγμα περιγραφής σε C και (b) Ο αντίστοιχος γράφος ροής δεδομένων (DFG). [Gupt04]

Μοντελοποίηση πόρων υλικού, Συγχρονισμού και των τύπων δεδομένων

Εκτός από την περιγραφή σχεδίου, τα εργαλεία σύνθεσης υψηλού επιπέδου χρειάζονται τις πληροφορίες για τους πόρους υλικού που διατίθενται για να σχεδιάσουν το σχέδιο και το συγχρονισμό αυτών των πόρων. Αναγκαίες είναι επίσης οι πληροφορίες για τους τύπους στοιχείων προκειμένου να παραχθεί ο κώδικας παραγωγής σε μια γλώσσα περιγραφής υλικού (HDL), όπως εξηγείται στο επόμενο τμήμα.

Μοντελοποίηση τύπων δεδομένων Η γλώσσα προγραμματισμού "C" υποστηρίζει διάφορους τύπους στοιχείων συμπεριλαμβανομένου των ακέραιων αριθμών, αριθμών με κινητή υποδιαστολή, χαρακτήρων και παραλλαγών αυτών όπως short, long, double κ.λπ. Επιπλέον, κάθε ένας από αυτούς τους τύπους στοιχείων μπορεί να έχει ή και όχι πρόσημο. Στη μοντελοποίηση για το εργαλείο του SPARK επιτρέπεται στο σχεδιαστή να καθορίσει τη σειρά των διάφορων τύπων στοιχείων ως πίνακα σε ένα αρχείο περιγραφής υλικού. Αυτός ο πίνακας έχει τρεις στήλες: ο τύπος στοιχείων, το χαμηλότερο εύρος στοιχείων και το ανώτερο εύρος στοιχείων. Ένα παράδειγμα ενός πίνακα τύπων δεδομένων παρουσιάζεται στον πίνακα 2.1. Σε αυτόν τον πίνακα, καθορίστηκε ότι οι μεταβλητές του τύπου "integer" (ακέραιος αριθμός) κυμαίνονται από -32767 έως 32768 αυτό αντιστοιχεί δεκαεξάμπιτο (16 bit) Boolean στο υλικό. Επίσης, αυτός ο ίδιος πίνακας μπορεί να χρησιμοποιηθεί για να κάνει τις καταχωρήσεις για τις συγκεκριμένες μεταβλητές από την περιγραφή εισόδου. Αυτό επιτρέπει στο σχεδιαστή για να χρησιμοποιήσει τη γνώση του για το σχέδιο για να παρέχει τους περιορισμούς στο εύρος των μεταβλητών. Παραδείγματος χάριν, στον πίνακα 2.1, έχει καθοριστεί ότι η μεταβλητή "myVariable" στην περιγραφή σχεδίου έχει εύρος από 0 έως 15. Το εργαλείο σύνθεσης μπορεί να χρησιμοποιήσει αυτές τις πληροφορίες για να παραγάγει 4 μπιτ (4 bit) Boolean στο υλικό για αυτήν την μεταβλητή αντί δεκαεξάμπιτου

Data type	Lower Data Range	upper data range
integer	−32767	32767
myVariable	0	15

Πίνακας 2.1: Παράδειγμα πίνακα για τους τύπους δεδομένων στο αρχείο περιγραφής υλικού. [Gupt04]

(16 bit) Boolean που παράγεται για όλους τους άλλους ακέραιους αριθμούς.

Οι πληροφορίες για τους τύπων των δεδομένων είναι ουσιαστικές για την παραγωγή του συνθέσιμου κώδικα HDL, δεδομένου ότι τα εργαλεία σύνθεσης λογικής απαιτούν το ακριβή εύρος των τύπων στοιχείων στον κώδικα HDL. Ως εκ τούτου, η γεννήτρια κώδικα για ένα εργαλείο σύνθεσης υψηλού επιπέδου χρειάζεται αυτόν τον πίνακα των τύπων στοιχείων για τη σωστή παραγωγή VHDL.

Μοντελοποίηση των πόρων δεδομένων Οι λειτουργικοί πόροι, ή λειτουργικές μονάδες μοντελοποιούνται μέσω μία *βιβλιοθήκης για τους πόρους υλικού (hardware resource library)*. Αυτή η βιβλιοθήκη για του πόρους υλικού περιέχει τις ακόλουθες πληροφορίες:

- *Διαδικασίες που μπορούν να "χαρτογραφηθούν" (mapped) στο λειτουργικό πόρο.*Παραδείγματος χάριν, μπορεί να καθοριστεί ότι και οι προσθέσεις και οι αφαιρέσεις μπορούν να χαρτογραφηθούν σε μια μονάδα ALU. Σε μια βιβλιοθήκη τεχνολογίας, μπορούν να υπάρξουν διάφορες λειτουργικές μονάδες στις οποίες μια λειτουργία μπορεί να εκτελεσθεί. Ο στόχος της επιλογής μια συγκεκριμένης λειτουργικής μονάδας από μια βιβλιοθήκη συστατικών, για κάθε λειτουργία στο σχέδιο είναι γνωστός ως επιλογή ενότητας (module selection). Η επιλογή ενότητας έχει αποτελέσει το αντικείμενο πολλής έρευνας στο παρελθόν· εντούτοις, στο εργαλείο SPARK υπάρχει η υπόθεση ότι ο σχεδιαστής κάνει την κατανομή των πόρων και την επιλογή ενότητας a priori.

- *Αριθμός εισόδων και εξόδων του λειτουργικού πόρου.* Το πρότυπο των πόρων στο σύστημά του SPARK περιορίζεται σε 2 εισόδους (ή 1 είσοδο) και 1 έξοδο. Η εξαίρεση είναι κλήσεις συναρτήσεων οι οποίες επίσης καθορίζονται ως πόρους υλικού. Ο αριθμός εισόδων και εξόδων μιας κλήσης συνάρτησης καθορίζεται από τη δήλωση της συνάρτησης στην περιγραφή εισόδου. Ως εκ τούτου, ένας τρόπος να διαμορφωθούν σύνθετοι πόροι, πολλαπλών εισόδων (multi-input) στο πλαίσιο σύνθεσής υψηλού επιπέδου είναι να δηλωθούν ως κλήσεις συναρτήσεων.

- *Κύκλοι και χρόνος εκτέλεσης του λειτουργικού πόρου.* Αυτές οι πληροφορίες χρησιμοποιούνται για τον καθορισμό εάν ο λειτουργικός πόρος μπορεί να chained ή είναι ένας πόρος πολλαπλών-κύκλων. Οι πληροφορίες σε αυτές τις δύο παραμέτρους είναι περιττές· μόνο ο χρόνος εκτέλεσης είναι αρκετός.

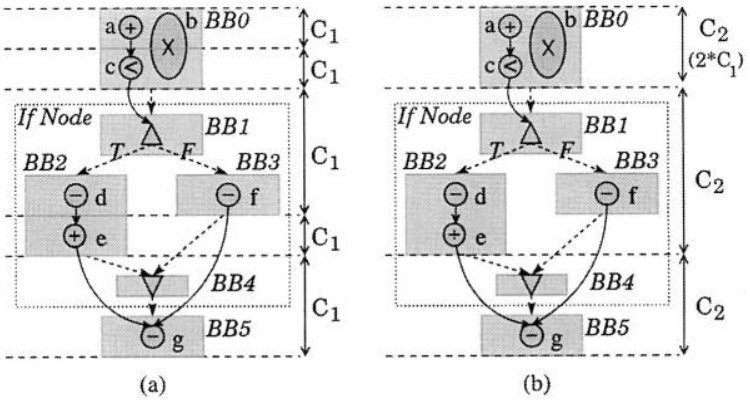
- *Αριθμός μονάδων για κάθε τύπου λειτουργικών πόρων.* Αυτό αντιστοιχεί στην κατανομή των πόρων για το σχέδιο.

- *Κόστος επιφάνειας υλικού του λειτουργικού πόρου.* Αυτό μπορεί να ενσωματωθεί στις συναρτήσεις κόστους που χρησιμοποιούνται από ευρυστικούς τρόπους (heuristics) επιλογής ενότητας κατά την επιλογή μεταξύ διάφορων πόρων στους οποίους μια λειτουργία μπορεί να χαρτογραφηθεί

Το πρότυπο των λειτουργικών πόρων υλικού που παρουσιάζεται πιο πάνω μπορεί να επεκταθεί για να περιλάβει τις πληροφορίες για τη δομική διοχέτευση και διοχέτευση καταχωρητών, επικοινωνία,διασύνδεση και κατανομή δίαυλου. Οι πόροι μνήμης όπως οι καταχωρητές και οι μόνο

ανάγνωσης και μνήμες τυχαίας προσπέλασης (ROMs και RAMs) μπορούν επίσης να διαμορφωθούν ρητά με τον ίδιο τρόπο.

Μοντελοποίηση του χρονισμού κύκλου ρολογιού Το αρχείο περιγραφής υλικού περιέχει επίσης τις πληροφορίες για την *περίοδο ρολογιών* του σχεδίου, που δείχνεται από C_{CLK}. Η περίοδος ρολογιού δηλώνει το διαθέσιμο χρόνο σε κάθε βήμα ελέγχου ή κύκλο, για τις λειτουργίες να εκτελέσουν. Ο χρονοδρομολογητής χρησιμοποιεί αυτό το χρόνο και το χρόνος εκτέλεσης των λειτουργικών πόρων (που καθορίζεται στη βιβλιοθήκη των πόρων υλικού, όπως εξηγείται στο προηγούμενο τμήμα) για να χρονοδρομολογήσει λειτουργίες σε κάθε κύκλο ρολογιού.



Σχήμα 2.8: (a) Παράδειγμα γράφου με χρονοδρομολογημένο πολλαπλασιαστή 2-κύκλων και με τις υπόλοιπες μονάδες 1-κύκλου. C_1 είναι η περίοδος κύκλου. (b) Η περίοδος κύκλου διπλασιάστηκε σε C_2. Στην περίπτωση αυτή οι πολλαπλασιαστές εκτελούν λειτουργία σε ένα κύκλο και οι υπόλοιπες μονάδες σε μισό κύκλο. Αυτό επιτρέπει chaining λειτουργιών a και c στο βασικό block bb_0 και οι λειτουργίες d και e στο bb_2. [Gupt04]

Εξετάστε το παράδειγμα στο σχήμα 2.8(a). Η περίοδος κύκλου ρολογιού που ορίζεται σε αυτό το σχέδιο είναι το C_1. Το σχέδιο έχει χρονοδρομολογηθεί με έναν πολλαπλασιαστή 2-κύκλων και από έναν αθροιστή, αφαιρέτη και συγκριτή 1-κύκλου. Έτσι, θεωρώντας το C_1 είναι 10 νανοδευτερόλεπτα (ns), άρα ο πολλαπλασιαστής εκτελεί μέσα σε $20ns$ και οι άλλοι λειτουργικοί πόροι σε $10ns$. Οι διακεκομμένες γραμμές δείχνουν τα όρια των κύκλων. Ως εκ τούτου, όπως φαίνεται σε αυτόν το σχέδιο, η λειτουργία πολλαπλασιασμού πολλαπλών-κυκλων (b) χρονοδρομολογήτε για να εκτελεστεί σε δύο κύκλους. Σημειώνεται ότι, όλες οι "εκτάσεις" για τους χρόνοι C_1 είναι ίσης αξίας ακόμα κι αν εμφανίζονται άνισες στο σχήμα 2.8(a).

Μοντελοποίηση του chaining για λειτουργίες Η *αλυσόδεση λειτουργίας (operation chaining)* είναι μια σημαντική τεχνική για τη σύνθεση υψηλού επιπέδου. Δύο διαδικασίες που αλυσοδένονται μαζί εκτελούνται στον ίδιο κύκλο χωρίς οποιοδήποτε στοιχείο μνήμης μεταξύ τους για να αποθηκεύσουν το ενδιάμεσο αποτέλεσμα. Δηλαδή δύο διαδικασίες αλυσοδένονται μέσα σε έναν βασικό κομμάτι (block) με τη χρονοδρομολόγηση τους στο ίδιο βήμα ελέγχου (ή τον κύκλο ρολογιού) και τη σύνδεση των εξόδων της μιας λειτουργίας στις εισόδους της άλλης λειτουργίας.
Εάν διπλασιαστεί η περίοδος ρολογιού στο παράδειγμα στο σχήμα 2.8(a) σε $C_2 = 2 * C_1$,τότε ένα πιθανό χρονοδρομολογημένο σχέδιο είναι όπως φαίνεται στο σχήμα 2.8(b). Η λειτουργία του

πολλαπλασιασμού παίρνει τώρα έναν κύκλο της νέας περιόδου ρολογιού C_2 για να εκτελεστεί (και οι δύο είναι $20ns$). Σε αυτό το παράδειγμα, οι διαδικασίες a και c, και οι διαδικασίες d και e αλυσοδένονται (chained) μεταξύ τους, δηλαδή εκτελούνται "πλάτη με πλάτη" στον ίδιο κύκλο χωρίς στοιχεία μνήμης μέσα-μεταξύ τους. Αυτό είναι επειδή ο συνολικός χρόνος εκτέλεσης των αλυσοδεμένων διαδικασιών είναι $10 + 10 = 20ns$. Σε αυτό το παράδειγμα υπάρχει η υπόθεση το κόστος για τη πολύπλεξη (multiplexing) και τους ελέγχους συμπεριλαμβάνονται στους χρόνους εκτέλεσης που δίνονται για κάθε λειτουργικό πόρο στη βιβλιοθήκη των πόρων υλικού. Επίσης, με τις αλυσοδεμένες διαδικασίες, τα βασικά κομμάτια bb_0 και bb_2 έχουν μόνο ένα βήμα ελέγχου το καθένα σε αντίθεση με το παράδειγμα στο σχήμα 2.8(a), όπου είχαν δύο βήματα ελέγχου το καθένα.

2.2 Thermal Management

2.2.1 Γενικά

Όλα τα σύγχρονα ηλεκτρονικά κυκλώματα είναι εξοπλισμένα με μεγάλες υπολογιστικές δυνατότητες και λογική, με αποτέλεσμα σημαντικά μεγαλύτερες ανάγκες για ισχύ. Η πυκνότητα ισχύος αυξάνεται με κάθε νέα γενιά μικροεπεξεργαστών αφού τα χαρακτηριστικά μεγέθους και συχνότητας αλλάζουν κλίμακα με πιο γρήγορους ρυθμούς απ' ότι η τάση λειτουργίας. Αυτή η ισχύς που παρέχεται μετατρέπεται σε θερμότητα, η οποία μπορεί να μην είναι ομοιόμορφα κατανεμημένη στο χώρο του συστήματος. Η συγκέντρωση θερμότητας επέρχεται πολύ πιο γρήγορα απ' ότι η διάχυση στο ολοκληρωμένο λόγω του χαμηλού ρυθμού πλευρικής διάδοσης της θερμότητας. Αυτό οδηγεί σε άνισες κατανομές θερμοκρασίας στο κύκλωμα και περιοχές με υψηλή θερμοκρασία ονομάζονται "hotspots" ("ζεστά σημεία"). Τέτοιες ψηλές θερμοκρασίες, όταν δεν τις διαχειριστούμε, μπορούν να επηρεάσουν την ορθότητα της λειτουργίας του ολοκληρωμένου, όπως επίσης και την μείωση της ταχύτητας και "ζωής" του. Στους μικροϋπολογιστές αυτό έχει οδηγήσει σε μερικές λύσεις ψύξης, από τις οποίες οι παραδοσιακές έχουν σχεδιαστεί για τη χειρότερη περίπτωση έκλυσης ισχύος (power dissipation) και έχουν επικεντρωθεί συνήθως στο θερμικό πακέτο (ψύκτρα, ανεμιστήρας κτλ.). Οι τεχνικές χαμηλής ισχύος δεν μπορούν να εξαλείψουν το σχηματισμό hotspot αφού δεν μειώνουν την πυκνότητα ισχύος σ' αυτά. Γι' αυτούς του λόγους έγινε επιτακτική η ανάγκη για την κατασκευή θερμικά ενήμερες ροές σχεδίασής (thermal aware design flows).

Όπως αναφέρθηκε και πιο πάνω, η υψηλή θερμοκρασία επηρεάζει αρνητικά την απόδοση του κυκλώματος. Με αυξημένη τη θερμοκρασία, η αντίσταση των διασυνδέσεων αυξάνεται και η ταχύτητα αλλαγής (switching speed) των τρανζίστορ μειώνεται. Και οι αργές συσκευές συνδεδεμένες με αργές διασυνδέσεις στην εγγύτητα των ζεστών λειτουργικών μονάδων (FUs) μπορούν να προκαλέσουν παραβίαση του χρονισμού και να οδηγήσουν με την πάροδο του χρόνου σε λειτουργική αποτυχία. Ακόμα και αν η υπερβολική θερμότητα δεν οδηγήσει σε ξαφνικές ζημιές, επιταχύνει τη μετακίνηση ηλεκτρονίων (electromigration) η οποία μπορεί να οδηγήσει σε μόνιμες ζημιές μακροπρόθεσμα. Επιπλέον στα θέματα αξιοπιστίας, η εξάρτηση της διαρροής ισχύος στη θερμοκρασία δημιουργεί μια πρόκληση για τη σχεδίαση χαμηλής ισχύος. Καθώς το μερίδιο της διαρροής στους συνολικούς προϋπολογισμούς αυξάνεται, η συρρίκνωση στο μερίδιο της δυναμικής ισχύος θα οδηγήσει στη χρησιμοποίηση λιγότερης πραγματικής ισχύος για την απόδοση. Η διαρροή έχει εκθετική εξάρτηση με τη θερμοκρασία. Επομένως, οι υψηλές θερμοκρασίες μπορούν προκαλέσουν μια μεγάλη μετατόπιση στις κατανομές προϋπολογισμών ισχύος στη σχεδίαση ενσωματωμένων συστημάτων με περιορισμούς ισχύος.

Διάφορες τεχνικές στη συσκευασία των ολοκληρωμένων και τεχνικές ψύξης όπως η ψήκτρα, ο ανεμιστήρας, κλπ έχουν αναπτυχθεί. Έχει παρουσιαστεί ότι η σχέση μεταξύ των ικανοτήτων ψύξης και του κόστους της λύσης είναι μη γραμμική καθώς η πυκνότητα ισχύος αυξάνεται. Αυτό δείχνει τη σημασία για τον περιορισμό της μέγιστης θερμοκρασίας για να υπάρξει αποδοτικός έλεγχος της θερμότητας. Εκτός από τη θερμική συσκευασία, υπάρχουν δύο προσεγγίσεις στη θερμική διαχείριση: 1) θερμική διαχείριση κατά τη διάρκεια της εκτέλεσης και 2) προγραμματισμός και βελτιστοποίηση της σχεδίασης κατά τη διάρκεια της σύνθεσης.

Οι πιο πρόσφατες λύσεις περιλαμβάνουν την διαχείριση της συμπεριφοράς της εφαρμογής προσαρμόζοντας την ανάλογα με τη θερμοκρασία του ολοκληρωμένου κυκλώματος. Αυτοί οι οδηγούμενοι μηχανισμοί που ανατροφοδοτούνται στο χρόνος εκτέλεσης ονομάζονται Δυναμική Θερμική Διαχείριση (DTM). Επιβραδύνουν την εκτέλεση του μικροεπεξεργαστή σε απάντηση στη θερμοκρασία που ανιχνεύουν, με συνέπεια τη μείωση της ισχύς που καταναλώνεται και ως εκ τούτου τη μείωση της θερμοκρασίας στο ολοκληρωμένο κύκλωμα.

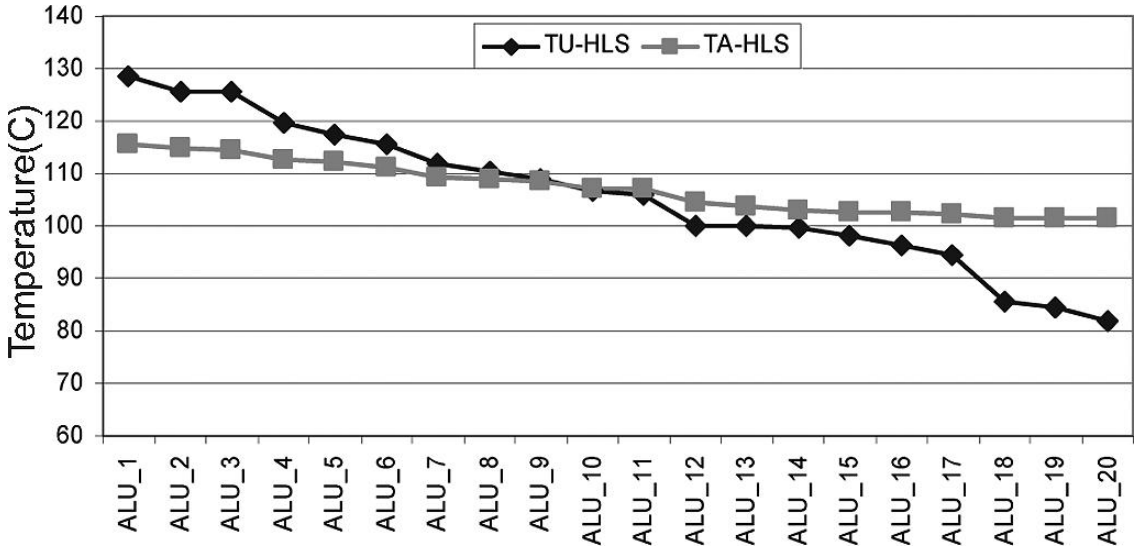
Δεδομένου ότι τα περισσότερα σχέδια DTM περιλαμβάνουν την παύση του ρολογιού επεξεργαστών ή τη μείωση της τάσης ανεφοδιασμού του, έχουν ορισμένες επιπτώσεις σε έναν υψηλής απόδοσης μικροεπεξεργαστή. Αρχικά, σε συστήματα πολυεπεξεργαστών βασισμένα σε κεντρικό υπολογιστή (multi-processor server-based systems) αυτά οδηγούν στα προβλήματα με το συγχρονισμό ρολογιών και την ακριβή διατήρηση του χρονισμού. Αφετέρου, οι εφαρμογές υψηλών επιδόσεων, που είναι απαιτητικές σε ισχύ και θερμικά απαιτητικές αναγκάζουν το DTM για να λειτουργεί με αποτέλεσμα να προκαλούν την επιβράδυνση τους. Αυτό επιφέρει αρνητικές επιπτώσεις στη προσφορά των συστημάτων σε πραγματικό χρόνο αφού οι επιβραδύνσεις που προκαλούνται είναι απρόβλεπτες και θα μπορούσαν ενδεχομένως να οδηγήσουν στη μη τήρηση των υπολογιστικών χρονοδιαγραμμάτων. Οι περιπτώσεις DTM σχεδιάζονται ως λύσεις για να αντιμετωπίσουν τις χειρότερες περιπτώσεις εφαρμογών όπου η θερμική συσκευασία αντιμετωπίζει τη μέση περίπτωση. Εντούτοις, καθώς οι επεξεργαστές κάθε καινούργιας τεχνολογικής γενιάς θερμαίνονται περισσότερο, οι εφαρμογές μεσαίας θερμικής κατάστασης τείνουν να γίνονται πιο ζεστές προκαλώντας σφάλματα αξιοπιστίας και την υψηλότερη διαρροή. Ως εκ τούτου, στατικές τεχνικές μικρο-αρχιτεκτονικής για τη διαχείριση της θερμοκρασία μπορούν να συμπληρώσουν αυτό που η DTM προσπαθεί να επιτύχει.

Μαζί με την πυκνότητα ισχύος των λειτουργικών κομματιών, ένας άλλος σοβαρός παράγοντας που έχει επιπτώσεις στη κατανομή θερμοκρασίας ενός ολοκληρωμένου κυκλώματος είναι η πλευρική διάδοση της θερμότητας στο πυρίτιο. Αυτό εξαρτάται από τη γειτνίαση λειτουργικών μονάδων που καθορίζεται κατά τη διάρκεια χωροθέτησης (floorplan) του μικροεπεξεργαστή. Παραδοσιακά, το floorplanning έχει εξεταστεί σε επίπεδο πιο κοντά στα κυκλώματα απ' ότι στο επίπεδο της μικρο-αρχιτεκτονικης. Ένας από τους λόγους για αυτό είναι το επίπεδο αναλυτικών πληροφοριών στο οποίο εξαρτάται η τοποθέτηση (floorplanning), το οποίο είναι μόνο διαθέσιμο στο επίπεδο κυκλωμάτων. Εντούτοις, με τις καθυστερήσεις καλωδίων που είναι μεγαλύτερες από τις καθυστερήσεις λογικής και τη θερμοκρασία να γίνεται ένας πρώτης θέσης περιορισμός σχεδίασης, έχει αρχίσει να εξετάζεται ακόμη και στο επίπεδο της μίκρο-αρχιτεκτονικής. Η σχέση και η ανταλλαγή μεταξύ της απόδοσης και της θερμοκρασίας έχουν εξεταστεί αρκετά σε έναν υψηλότερο επίπεδο αφαίρεσης και παρά τη χρησιμοποίηση προτύπων που δεν είναι απαραιτήτως πολύ λεπτομερή, έχει επισημανθεί η δυνατότητα της μικρο-αρχιτεκρονικής στη μείωση της μέγιστης θερμοκρασίας επεξεργαστών και της δυνατότητας της συμπληρώνοντας τη σχεδίαση DTM. Σημειώνεται ότι ο μέσος όρος της θερμοκρασίας του ολόκληρου του κυκλώματός δεν μειώνεται πάρα πολύ. Εξισώνει τις θερμοκρασίες των λειτουργικών μονάδων μέσω καλύτερης διάδοσης. Επομένως, οι πιο ζεστές μονάδες γίνονται πιο κρύες ενώ η θερμοκρασία σε μερικά από τους πιο κρύα μέρη αυξάνεται αναλόγως. Αυτή η πτυχή floorplanning είναι ιδιαίτερα ελκυστική σε σύγκριση με τη στατική εξωτερική ψύξη. Ενώ η ψύξη μειώνει την περιβαλλοντική θερμοκρασία και ως εκ τούτου η μέγιστη θερμοκρασία στο ολοκληρωμένο, δεν μειώνει την κλίση θερμοκρασίας μέσα σ' αυτό. Αυτό μπορεί να είναι κακό από μια σκοπιά αξιοπιστίας.

Για τη μελέτη και την υλοποίηση του floorplanning ώστε να έχουμε καλύτερη κατανομή θερμοκρασίας στο ολοκληρωμένο κύκλωμα και επομένως βελτιστοποίησης της θερμικής διαχείρισης κατά η διάρκεια της σχεδίασης υπάρχουν διάφορα εργαλεία, ένα από τα οποία είναι και το HotFloorplan το οποίο παρουσιάζεται παρακάτω.

2.2.2 Εργαλείο τοποθέτησης με θερμική διαχείριση - HotSpot

Το εργαλείο HotSpot αποτελεί ένα ευρέος διαδεδομένο θερμικό μοντέλο που χρησιμοποιείται επί το πλείστον από την ερευνητική κοινότητα αρχιτεκτονικής υπολογιστών. Το λογισμικό HotSpot, που δημιουργήθηκε στο Πανεπιστήμιο στης Virginia (University of Virginia), αποτελεί εργαλείο για τη μοντελοποίηση της θερμοκρασίας στους μικροεπεξεργαστές. Το HotSpot επιτρέπει στο χρήστη να καθορίσει το σχέδιο (floorplan) του επεξεργαστή μαζί με τις λειτουργικές μο-



Σχήμα 2.9: Το προφίλ των θερμοκρασιών για τις μονάδες(TU-HLS-temperature unaware HLS και TA-HLS-temperature aware).[Mukh06]

νάδες του και από αυτό το σχέδιο δημιουργεί ένα ισοδύναμο μοντέλο κυκλώματος το οποίο αντιπροσωπεύει τη μετάδοση θερμότητας στον "κουτί" του επεξεργαστή με τη συγκεκριμένη θερμική συσκευασία. Μέσα στο λογισμικό του HotSpot περιλαμβάνεται το εργαλείο για την τοποθέτηση - floorplanning σε επίπεδο μικρο-αρχιτεκτονικής με θερμική διαχείριση (thermal-aware floorplanning tool), το οποίο επεκτείνει το κλασσικό αλγόριθμο εξομοίωσης για τοποθέτηση τεμαχίων, για να αποτελέσει τη θερμοκρασία στη συνάρτηση κόστους του χρησιμοποιώντας το HotSpot.

Το HotFloorplan χρησιμοποιεί Normalized Polish Expressions (NPE) για να αναπαραστήσει το διάστημα λύσης των sliceable floorplans και χρησιμοποιεί τρεις διαφορετικούς τύπους τυχαίων κινήσεων διαταραχής για να πλοηγηθεί ανάμεσά τους. Ο περιορισμός στο λόγο διάστασης για κάθε λειτουργικό κομμάτι αντιπροσωπεύεται ως τμηματικές καμπύλες γραμμικής μορφής. Για κάθε δομή τεμαχίου που αντιστοιχεί σε ένα NPE, η ταξινόμηση για την ελάχιστη περιοχή των μεμονωμένων κομματιών γίνεται από μια κάτω προς τα επάνω, πολυωνυμικού χρόνου πρόσθεση των καμπυλών μορφής σε κάθε επίπεδο του slicing δέντρου. Μόλις γίνει η ταξινόμηση, η τοποθέτηση περνά στο HotSpot για τον υπολογισμό θερμοκρασίας σταθερής κατάστασης. Χρησιμοποιεί τις τιμές κατανάλωσης ισχύος που παράγονται από το προφίλ κάθε λειτουργικού κομματιού και της τοποθέτησης που παράγεται από το τρέχον βήμα του HotFloorplan για να επιστρέψει την αντίστοιχη μέγιστη θερμοκρασία σταθερής κατάστασης. Το HotFloorplan έπειτα συνεχίζει την εξερεύνηση μέσω της χρήσης της εξομοίωσης annealing ως σχέδιο αναζήτησης μέσω του διαστήματος λύσης.

Το HotSpot χρησιμοποιεί την ακόλουθη συνάρτηση κόστους:

$$Cost = (A + \lambda \cdot W) \cdot T \tag{2.1}$$

όπου το A είναι η περιοχή (area) που αντιστοιχεί στην ταξινόμηση ελάχιστης-περιοχής της τρέχουσας τεμαχισμένης δομής, το T είναι η μέγιστη θερμοκρασία σταθερής-κατάστασης, το W είναι το μήκος καλωδίου το οποίο δίνεται από $\sum c_{ij} d_{ij}, 1 \leq i, j \leq n$, όπου το n είναι ο αριθμός των λειτουργικών κομματιών, c_{ij} είναι η πυκνότητα καλωδίου της διασύνδεσης μεταξύ των κομματιών i και j, και το d_{ij} είναι η απόσταση Μανχάταν (Manhattan distance) μεταξύ των κέντρων των κομματιών. Το λ αποτελεί την παράμετρο βάρους για τον έλεγχο μεταξύ των A

και W. Καθώς επίσης και οι μονάδες μετρήσεις των A και W είναι διαφορετικές, η παράμετρος λ χρησιμεύει για να "ταιριάζει" τα μεγέθη στο ίδια "διάταξη". Ο όρος $(A + \lambda \cdot W)$ είναι αυτός που χρησιμοποιείται στους αρχικούς floorplanning αλγορίθμους. Ο νέος όρος T έχει περιληφθεί ως όρος πολλαπλασιασμού αντί σαν όρος πρόσθεσης. Αυτό γιατί βρέθηκε η συνάρτηση κόστους να έχει καλύτερη διαβάθμιση στην τιμή για διαφορετικά floorplans όταν ο όρος T συμπεριλήφθηκε σαν όρος πολλαπλασιασμού αντί για όρος πρόσθεσης. Αυτό επέτρεψε στο floorplanner για να βρει περισσότερες βελτιστοποιημένες λύσεις.

Υπάρχουν δύο floorplanning σχέδια. Το πρώτο, που ονομάζεται flp-basic, είναι ένα σχέδιο όπου σε όλα τα microarchitectural διαμορφωμένα καλώδια δίνεται ίσο βάρος, δηλαδή, $c_{ij} = 1, \forall i, j$. Στο δεύτερο, το οποίο ονομάζεται flp-advance, τα βάρη c_{ij} υπολογίζονται κατά τέτοιο τρόπο ώστε το $W = \sum c_{ij} d_{ij}$ είναι ανάλογο προς μια εκτίμηση της επιβράδυνσης στο χρόνο εκτέλεσης της εφαρμογής όταν οργανώνεται στο floorplan που αξιολογείται, σε σύγκριση με την επιβράδυνση με μια προεπιλογή floorplan.

Για την annealing προσομοίωσης, χρησιμοποιούμε πρόγραμμα θερμοκρασίας με σταθερή αναλογία έτσι ώστε η annealing θερμοκρασία μιας διαδοχικής επανάληψης είναι 99% της προηγούμενης. Η αρχική annealing θερμοκρασία τίθεται έτσι ώστε η πιθανότητα αποδοχής αρχικής κίνησης είναι 0.99. Η annealing διαδικασία ολοκληρώνεται μετά από 1000 επαναλήψεις ή μετά από την annealing θερμοκρασία γίνεται μικρότερη από ένα κατώτατο όριο, ανάλογα με ποίος από τους δύο περιορισμούς ικανοποιηθεί πρώτος. Το κατώτατο όριο υπολογίζεται έτσι ώστε η πιθανότητα απόρριψης κίνησης σε εκείνη την θερμοκρασία είναι 99%.

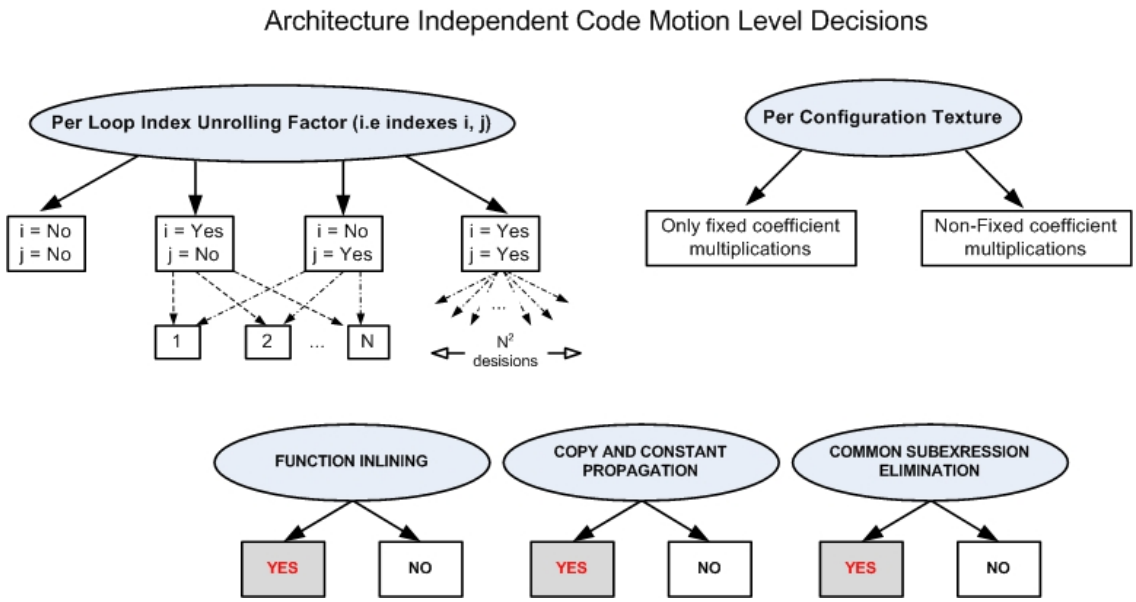
Κεφάλαιο 3

Κύριο Μέρος - Υλοποίηση

3.1 Μοντελοποίηση του χώρου σχεδίασης

Η μοντελοποίηση του χώρου σχεδίασης αφορά τις παραμέτρους με τις οποίες μπορεί να παραμετροποιηθεί ώστε να βελτιστοποιοηθεί η εξερεύνηση. Έγινε σε 4 διαφορετικά επίπεδα στα οποία έχουν αναλυθεί οι διάφορες παράμετροι που επηρεάζουν τις αποφάσεις για κάθε επόμενο επίπεδο. Η μοντελοποίηση βασίστηκε σε ορθογώνια δέντρα αποφάσεων (Orthogonal Decision Trees) τα οποία αποτελούν ένα αποτελεσματικό τρόπο για την κατασκευή μιας ακριβής και σημαντικής αναπαράστασης μεγάλων συνόλων δέντρων αποφάσεων χωρίς πλεονασμούς. Τα ορθογώνια δέντρα απόφασης είναι λειτουργικά ορθογώνια το ένα στο άλλο και αντιστοιχούν στα κύρια συστατικά του χώρου μοντελοποίησης.

3.1.1 Δέντρα αποφάσεων για το επίπεδο των μετατροπών κώδικα (Architecture Independent Code Transformations Level)



Σχήμα 3.1: Μοντελοποίηση του χώρου σχεδίασης: Επίπεδο Code Motion.

Στο επίπεδο των μετατροπών κώδικα (Code Transformations) έχουμε τις επιλογές που φαίνονται στο σχήμα 3.1.

► *Δέντρο Per Loop Unrolling Factor*: Μια σημαντική παράμετρος, που αναπαρίσταται σ' αυτό το δέντρο, είναι παράγοντας διεύρυνσης του κώδικα (loop unrolling) για κάθε δείκτη βρόγχου. Όπως φαίνεται και από το σχήμα οι επιλογές για το loop unrolling αρχίζουν από καθόλου unrolling για όλους τους δείκτες βρόγχων που υπάρχουν στον κώδικα μέχρι και τη μέγιστη διεύρυνση που επιτρέπει κάθε βρόγχος για το δείκτη του και όλους τους συνδυασμούς γι' αυτές τις περιπτώσεις. Έτσι ανάλογα με τον αριθμό των βρόγχων που υπάρχουν και το μέγιστο αριθμό επαναλήψεων για τον καθένα έχουμε και τις ανάλογες επιλογές/συνδυασμούς
Ένα παράδειγμα για το unrolling είναι:
Μια συνάρτηση σε ένα κώδικα που υπολογίζει το άθροισμα 100 στοιχείων.

```
1  for (x = 0; x < 100; x++) {
2      sum=sum+x;
3  }
```

Listing 1: Παράδειγμα κώδικα χωρίς unrolling

Σε περίπτωση που εφαρμοστεί unrolling με βαθμό 5 στην προηγούμενη συνάρτηση ο κώδικας γίνεται ως εξής:

```
1  for (x = 0; x < 100; x += 5) {
2      sum=sum+x;
3      sum=sum+(x+1);
4      sum=sum+(x+2);
5      sum=sum+(x+3);
6      sum=sum+(x+4);
7  }
```

Listing 2: Παράδειγμα κώδικα με χρήση unrolling

Όπως φαίνεται οι επαναλήψεις έχουν περιοριστεί, από 100 σε 20, με το ίδιο αποτέλεσμα αφού μέσα στο σώμα της συνάρτησης εκτελούνται οι επαναλήψεις, λόγω τις επέκτασης του κώδικα από 3 γραμμές σε 7.

► *Δέντρο Per Configuration Texture*: Στο δεύτερο δέντρο υπάρχει η παράμετρος για πολλαπλασιασμό είτε μόνο με σταθερούς όρους είτε πολλαπλασιασμό και με μεταβλητές. Έχοντας πολλαπλασιασμό μόνο με σταθερούς όρους, οι απαραίτητες λειτουργίες μπορούν να γίνουν μόνο με αθροίσματα και ολισθήσεις. Στην περίπτωση που έχουμε πολλαπλασιασμούς με μεταβλητές είναι απαραίτητη η ύπαρξη και μονάδων πολλαπλασιαστών στο σχέδιο.

► *Δέντρο Function Inlining*: Η τρίτη παράμετρος είναι η αντικατάσταση κώδικα (function inlining), δηλαδή η αντικατάσταση της κλήσης μια συνάρτησης με τον ακριβή κώδικα της συνάρτησης όταν αυτή καλείται στο κύριο μέρος.
Παράδειγμα κώδικα με Function Inlining:

```
1  int absolute (int x){
2    if (x>=0) then
3      return x;
4    else
5      return -x;
6  }
```

Listing 3: Παράδειγμα συνάρτησης που καλείται από το κυρίως πρόγραμμα

```
1  int sum (int a, int b, int c){
2    return abs(a) + abs(b) + abs(c);
3  }
```

Listing 4: Παράδειγμα κώδικα πριν εφαρμοστεί μετατροπή Function Inlining

```
1   int sum (int a, int b, int c){
2     int temp=0;
3     if (a>=0) then temp += a; else temp += -a;   /* -1- */
4     if (b>=0) then temp += b; else temp += -b;   /* -2- */
5     if (c>=0) then temp += c; else temp += -c;   /* -3- */
6     return temp;
7   }
```

Listing 5: Παράδειγμα κώδικα μετά την εφαρμογή Function Inlining

► *Δέντρο Copy and Constant Propagation*: Η τέταρτη παράμετρος είναι η διάδοση αντιγράφων κώδικα και σταθερών (copy and constant propagation), η αντικατάσταση δηλαδή άμεσων αναθέσεων σε μεταβλητές ή με σταθερούς όρους με τις τιμές τους. Χρησιμοποίηση της διάδοσης αντιγραφής φαίνεται στο ακόλουθο παράδειγμα.

No copy propagation	copy propagation	
$y = x;$ $z = 3 + y;$	$z = 3 + x;$	⟵ όπως φαίνεται στην μεταβλητή y ανατίθεται η τιμή της x γι' αυτό και στον υπολογισμό της μεταβλητής z χρησιμοποιείται απευθείας η μεταβλητή x.

Πίνακας 3.1: Παράδειγμα μετατροπής κώδικα με copy propagation

► *Δέντρο Common Subexpression Elimination*: Η τελευταία παράμετρος αφορά την "διαγραφή" κοινών υποεκφράσεων (common subexpression elimination, CSE). Η επιλογή αυτή είναι μια βελτιστοποίηση του μεταγλωττιστή με την οποία ψάχνει τον κώδικα για περιπτώσεις κοινών εκφράσεων (όπως για παράδειγμα εκφράσεις που αποτιμούν την ίδια τιμή) και αναλύει κατά πόσο αξίζει η αντικατάσταση του επαναλαμβανόμενου υπολογισμού με μια μεταβλητή που περιέχει το αποτέλεσμα. Για παράδειγμα:

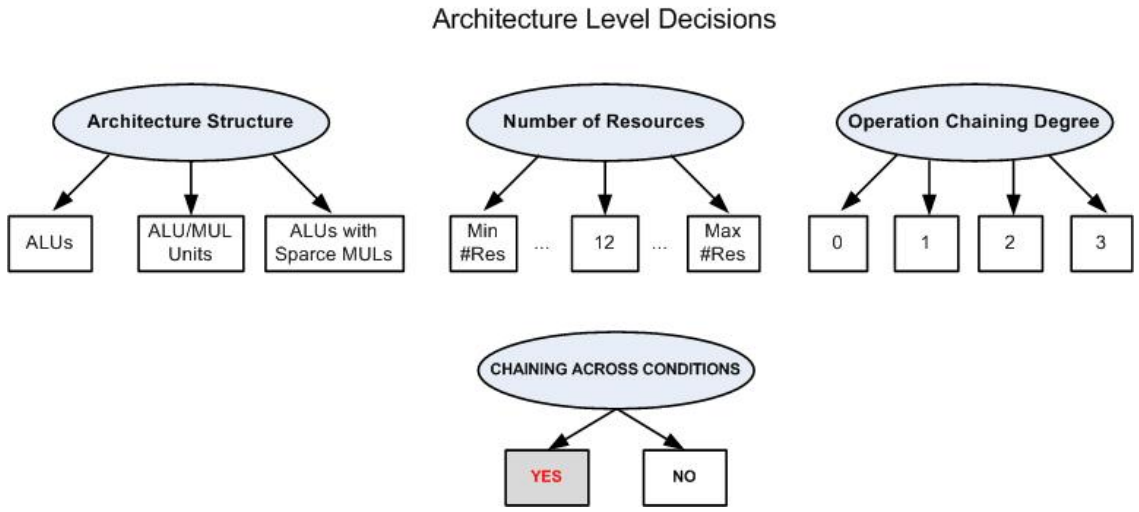
No CSE code	CSE code	
$a = b * c + g;$	$temp = b * c;$	⟵ όπως φαίνεται η τιμή $b * c$ αποτιμάται 2 φορές
$d = b * c + d;$	$a = tmp + g;$	γι' αυτό και υπολογίζεται και τοποθετείται στην μεταβλητή tmp
	$d = tmp + d;$	η οποία στη συνέχεια χρησιμοποιείται παρακάτω.

Πίνακας 3.2: Παράδειγμα μετατροπής κώδικα με "διαγραφή" κοινών υποεκφράσεων (Common Subexpression Elimination)

Για τις τρεις τελευταίες παραμέτρους (Function Inlining, Copy and constant propagation και Common subexpression elimination) η επιλογή ήταν αληθής (YES) για όλη τη εξερεύνηση που ακολουθήθηκε αφού είναι γνωστή η βελτιστοποίηση που προσφέρουν οι συγκεκριμένες επιλογές. Έτσι οποιαδήποτε εξερεύνηση με κάποια από της παραμέτρους αυτές να μην υπάρχει δε θα είχε νόημα.

3.1.2 Δέντρα αποφάσεων για το επίπεδο της αρχιτεκτονικής

Στο σχήμα 3.2 παρουσιάζονται με μορφή δέντρων οι διάφορες αποφάσεις/παράμετροι για την μοντελοποίηση του επιπέδου αρχιτεκτονικής. Αυτές οι παράμετροι πρέπει να ληφθούν υπόψη για τη εξερεύνηση και τη βελτιστοποίηση των λύσεων/περιορισμών που θα δοθούν στο επόμενο επίπεδο.



Σχήμα 3.2: Μοντελοποίηση του χώρου σχεδίασης: Επίπεδο αρχιτεκτονικής.

► *Δέντρο Architecture Structure*: Το δέντρο αυτό αφορά την αρχιτεκτονική δομή των λειτουργικών μονάδων που θα χρησιμοποιήσουμε έχοντας ως επιλογές τις απλές ALU (αριθμητική λογική μονάδα) μονάδες, τη χρήση μονάδων με μικτές λειτουργίες ALU και MUL (πολλαπλασιαστή) και τέλος τη χρήση ξεχωριστών λειτουργικών μονάδων ως ALU και MUL (ALUs with sparce MULs).

► *Δέντρο Number of Resources*: Το δεύτερο από τα δέντρα αφορά τον αριθμό των λειτουργικών μονάδων που θα χρησιμοποιηθούν στη σχεδίαση.

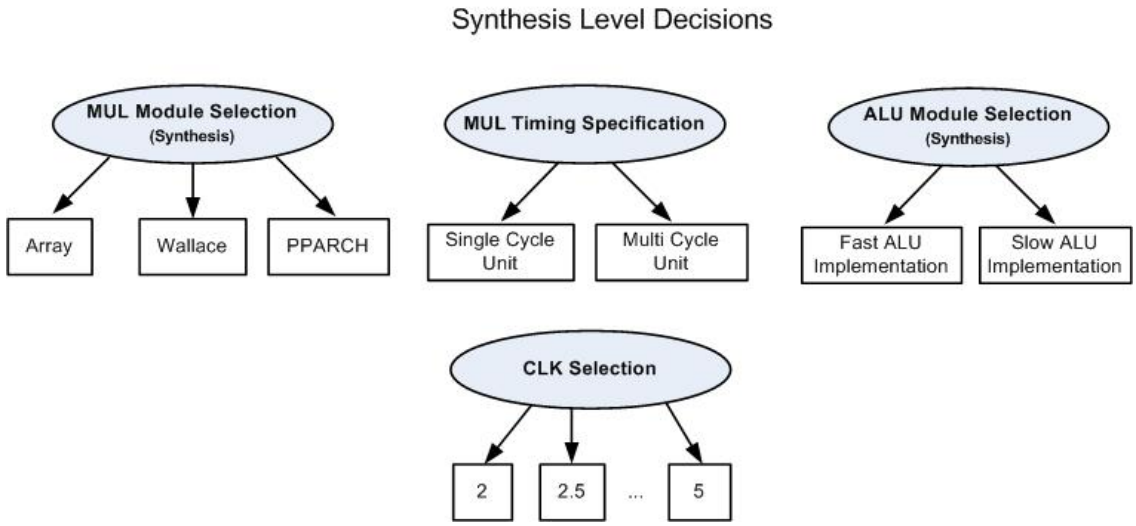
► *Δέντρο Operation Chaining Degree*: Το τρίτο δέντρο αφορά το σχηματισμό σύνθετων πρότυπων λειτουργικών μονάδων σε ένα κύκλο (Operation Chaining Degree) έχοντας ως επιλογές από chaining 0 (μηδέν) που σημαίνει κάθε λειτουργία σε ξεχωριστό κύκλο, μέχρι και το βαθμό chaining n που μεταφράζεται ως τη δυνατότητα αλυσιδωτής εκτέλεσης μέχρι και n+1 λειτουργιών με εξαρτήσεις δεδομένων (data dependent operations) σε ένα κύκλο ρολογιού (ο οποίος και προφανώς μεγαλώνει ανάλογα).

► *Δέντρο Chaining Across Conditions*: Στο τελευταίο δέντρο αποφάσεων φαίνεται η επιλογή για chaining διαμέσου των δομών ελέγχου (conditions) η οποία σε αυτή τη μοντελοποίηση αλλά και στην εξερεύνηση που υλοποιήθηκε διατηρήθηκε ως αληθής (YES) αφού παρατηρήθηκε σαφής βελτίωση με αυτή την επιλογή και δε θα είχε νόημα η περαιτέρω εξερεύνηση έχοντας την επιλογή αυτή στο όχι.

3.1.3 Δέντρα αποφάσεων για το επίπεδο της σύνθεσης (Synthesis Level)

Οι επόμενες παράμετροι του χώρου σχεδίασης αφορούν το επίπεδο της σύνθεσης. Αφορούν δηλαδή το κομμάτι στο οποίο ο κώδικας επιπέδου μεταφοράς καταχωρητών (RTL) θα χρησιμοποιηθεί για να παραχθεί το δίκτυο πυλών μαζί με τις τιμές/μετρήσεις για την επιφάνεια υλικού για κάθε κομμάτι της σύνθεσης μας όπως επίσης και για την κατανάλωση ισχύος που έχει το καθένα. Στο επίπεδο της σύνθεσης έχουμε τις επιλογές που φαίνονται στο σχήμα 3.3.

► *Δέντρο MUL Module Selection*: Η πρώτη απόφαση αφορά το μοντέλο πολλαπλασιαστή που θα χρησιμοποιηθεί για τη σύνθεση. Οι επιλογές είναι μεταξύ πολλαπλασιαστή τύπου Array,



Σχήμα 3.3: Μοντελοποίηση του χώρου σχεδίασης: Επίπεδο σύνθεσης.

Wallace και PPARCH. Η επιλογή αυτή αφέθηκε στο εργαλείο σύνθεσης που χρησιμοποιήθηκε (στη συγκεκριμένη περίπτωση το Synopsys Design Compiler).

► *Δέντρο MUL Timing Specification*: Η δεύτερη απόφαση αφορά το χρόνο εκτέλεσης για τους πολλαπλασιαστές του κυκλώματος. Αν δηλαδή το αποτέλεσμα ενός πολλαπλασιαστή θα είναι έτοιμο σε ένα κύκλο ρολογιού και άρα οι πολλαπλασιαστής θα θεωρηθεί μονάδα λειτουργίας ενός κύκλου, ή σε αντίθετη περίπτωση αν θα θεωρηθεί μονάδα λειτουργίας πολλαπλών κύκλων και το αποτέλεσμα θα παράγεται με το πέρας πολλών κύκλων. Η παράμετρος αυτή χρησιμοποιείται για τους πολλαπλασιαστές αφού αποτελούν και την πιο απαιτητική λειτουργία στο κύκλωμα.

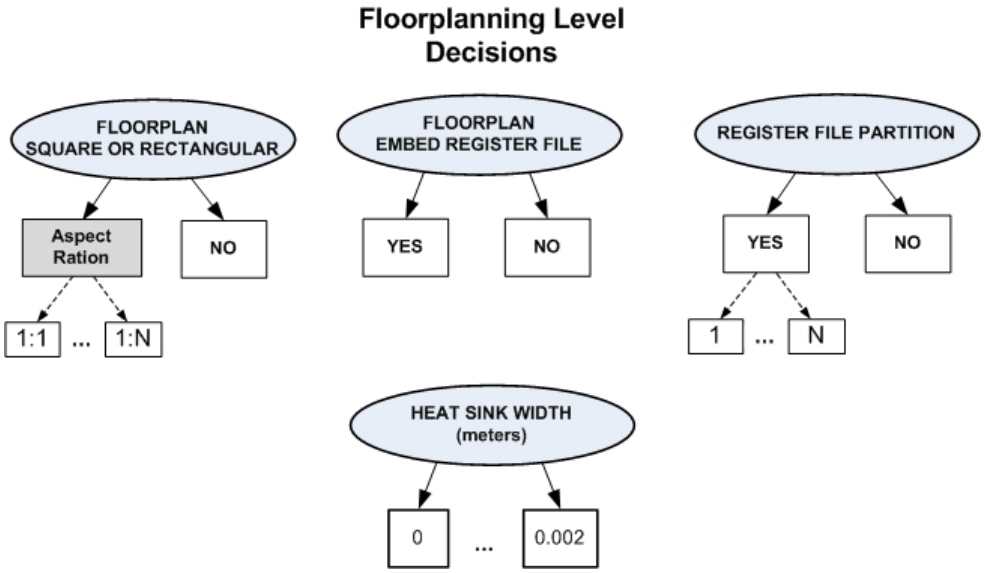
► *Δέντρο ALU Module Selection*: Το επόμενο δέντρο αποφάσεων αφορά το μοντέλο λογικής μονάδας (ALU) που θα χρησιμοποιηθεί στη σύνθεση. Οι επιλογές είναι μεταξύ εφαρμογής γρήγορης ALU και αργής ALU. Η επιλογή αυτή αφέθηκε στο εργαλείο σύνθεσης που χρησιμοποιήθηκε (στη συγκεκριμένη περίπτωση το Synopsys Design Compiler).

► *Δέντρο CLK Selection*: Η τελευταία απόφαση αφορά την επιλογή του ρολογιού της περιόδου που θα χρησιμοποιηθεί για τη σύνθεση του κυκλώματος. Αυτή η επιλογή είναι πολύ σημαντική αφού επηρεάζει εκτός από το χρονισμό του κυκλώματος και *(i)* την επιφάνεια υλικού και *(ii)* την κατανάλωση ισχύος σ' αυτό. Εφαρμόζοντας μικρή τιμή στο ρολόι το εργαλείο σύνθεσης θα προσπαθήσει να δημιουργήσει το δίκτυο πυλών με τη συγκεκριμένη περίοδο με αποτέλεσμα να αυξάνεται η επιφάνεια υλικού. Ακόμα σύμφωνα με τον τύπο της ισχύς η μικρή περίοδος επιφέρει και μεγαλύτερη κατανάλωση ισχύος για το κύκλωμα.

3.1.4 Αποφάσεις για το επίπεδο της Χωροθέτησης

Τέλος στο επίπεδο της χωροθέτησης (Floorplanning) προς μοντελοποίηση και εξερεύνηση υπάρχουν οι παράμετροι που φαίνονται στο σχήμα 3.4.

► *Δέντρο Floorplan Square or Rectangular*: Η πρώτη απόφαση αφορά το σχήμα του κάθε κομματιού (component) του κυκλώματος. Υπάρχουν οι επιλογές για το αν το σχήμα του λειτουργικού κομματιού θα είναι τετράγωνο ή ορθογώνιο. Στη περίπτωση που επιτραπεί η επιλογή για ορθογώνιο σχήμα υπάρχουν οι υποπεριπτώσεις/επιλογές για την αναλογία των διαστάσεων



Σχήμα 3.4: Μοντελοποίηση του χώρου σχεδίασης: Επίπεδο Floorplanning.

που θα έχει (aspect ratio). Η επιλογή της κατάλληλης αναλογίας διαστάσεων μπορεί να δώσει κύκλωμα με ισοκατανεμημένη θερμότητα, αποφεύγοντας έτσι τα ζεστά σημεία, hot spots.

► *Δέντρο Floorplan Embed Register File*: Για το συγκεκριμένο δέντρο αποφάσεων η επιλογές είναι η ύπαρξη ή μη μνήμης (Register File) ενσωματωμένης στην αρχιτεκτονική CGRArch. Η επιλογή που έγινε σ' αυτή την απόφαση είναι η ύπαρξη μνήμης στο σχέδιο, αφού όπως φαίνεται και από τα αποτελέσματα του 2ου επιπέδου εξερεύνησης (synthesis level) το κομμάτι των καταχωρητών έχει σημαντικό ποσοστό της επιφάνειας υλικού και της κατανάλωσης ισχύος του σχεδίου.

► *Δέντρο Register File Partition*: Το τρίτο δέντρο αποφάσεων αφορά την κατάτμηση της μνήμης (scratchpad) εφόσον αυτή υπάρχει στην αρχιτεκτονική (περιορισμός από το 2ο δέντρο αποφάσεων). Η επιλογές είναι από το 1, που ερμηνεύεται ως μία ενιαία μνήμη, μέχρι και N που σημαίνει ότι η μνήμη χωρίζεται σε N κομμάτια.

► *Δέντρο HeatSink Width*: Η επόμενη απόφαση αφορά την ύπαρξη ή όχι ψήκτρας στο floorplanning για τον υπολογισμό της θερμοκρασίας του σχεδίου. Στην επιλογή της ψήκτρας υπάρχει η παράμετρος για το πάχος που θα έχει, αφού προφανώς επηρεάζει τη θερμική κατανομή του κυκλώματος αλλά και του όγκου που αυτό θα έχει.

3.1.5 Στρατηγική εξερεύνησης για την επιλογή των καλύτερων παραμέτρων

Τέλος εφαρμόζοντας όλους τους παραπάνω στόχους (HLS framework, μοντελοποίηση της αρχιτεκτονικής CGRArch με κώδικα C, μοντελοποίηση του χώρου σχεδίασης) υλοποιείται μια ομογενή ροή μέσω κώδικα με μια καινούργια στρατηγική εξερεύνησης για την επιλογή των καλύτερων παράμετρων. Η στρατηγική αυτή δεν υλοποιεί brute εξερεύνηση, έλεγχο δηλαδή όλων των συνδυασμών των παραμέτρων που υπάρχουν αλλά με διάφορους ελέγχους κάνει αποκοπή (prunning) για διάφορες λύσεις που δε δίνουν καλύτερα αποτελέσματα. Μια πιο λεπτομερής επεξήγηση της εξερεύνησης που έχει δημιουργηθεί ακολουθεί παρακάτω.

3.2 Ανάλυση της εξερεύνησης

Ένας από τους σκοπούς αυτής της διπλωματικής εργασίας ήταν και η υλοποίηση μιας ροής σχεδίασης με την οποία τα διάφορα επίπεδα (HLS, Synthesis, Floorplan) θα διασυνδέονταν και δεδομένα αλλά και περιορισμοί θα περνούν από τα ψηλότερα επίπεδα αφαίρεσης στα χαμηλότερα (top-down approach), διευκολύνοντας την εξερεύνηση σε κάθε επίπεδο. Η εξερεύνηση που έχει υλοποιηθεί διαχωρίζεται σε 3 μέρη, όσα και τα επίπεδα της ροής που υπάρχουν.

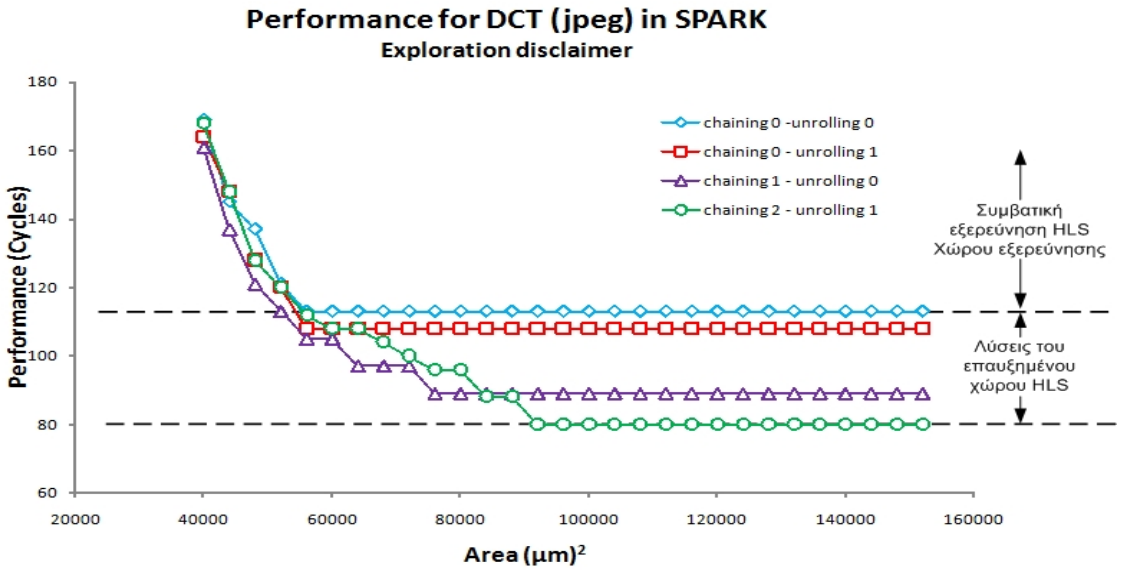
• Αρχικά υπάρχει η εξερεύνηση στο επίπεδο της σύνθεσης υψηλού επιπέδου, από την οποία με γνώμονα την απόδοση (performance) και την επιφάνεια υλικού (area) καθορίζονται οι διάφορες παράμετροι για το εργαλείο SPARK. Σαν αποτέλεσμα παράγεται ένα εύρος λύσεων που δίνουν τις κατάλληλες παραμέτρους για βέλτιστη απόδοση σε μικρότερη επιφάνεια υλικού.

• Στο επόμενο επίπεδο, αυτό της σύνθεσης, από την έξοδο του πρώτου μέρους, χρησιμοποιείται ο κώδικας RTL που αντιστοιχεί στις βέλτιστες λύσεις της εξερεύνησης. Για κάθε τέτοια λύση, εκτελείται το Design Compiler για να παραχθεί το δίκτυο πυλών με έλεγχο για το χρονισμό του ρολογιού (clock selection) για τυχόν παραβίαση στο χρόνο εκτέλεσης. Από τη σύνθεση εξάγονται δεδομένα που αφορούν την ακριβή επιφάνεια υλικού και την κατανάλωση ισχύος για κάθε σχέδιο.

• Τέλος στο επίπεδο του floorplanning χρησιμοποιούνται τα δεδομένα για την επιφάνεια υλικού και την κατανάλωση κάθε σχεδίου στο εργαλείο του HotFloorplan και HotSpot για να παραχθεί μια θερμική εικόνα. Στους υπολογισμούς και τη δημιουργία αυτής της θερμικής εικόνας υπάρχει η δυνατότητα βελτιστοποίησης στην κατανομή θερμότητας στο κύκλωμα με την αλλαγή διαφόρων παραμέτρων στα συγκεκριμένα εργαλεία.

3.2.1 Motivation

Στην παρακάτω γραφική παράσταση φαίνεται η αναγκαιότητα για εξερεύνηση στο επίπεδο του HLS με παραμέτρους που αφορούν το επίπεδο της αρχιτεκτονικής και των Code Transformations. Όπως φαίνεται λοιπόν στο σχήμα 3.5 η μέχρι τώρα εξερεύνηση με μόνες παραμέτρους τον αριθμό των λειτουργικών μονάδων δεν δίνει τη βέλτιστη λύση για την απόδοση σε κάθε επιφάνεια υλικού.

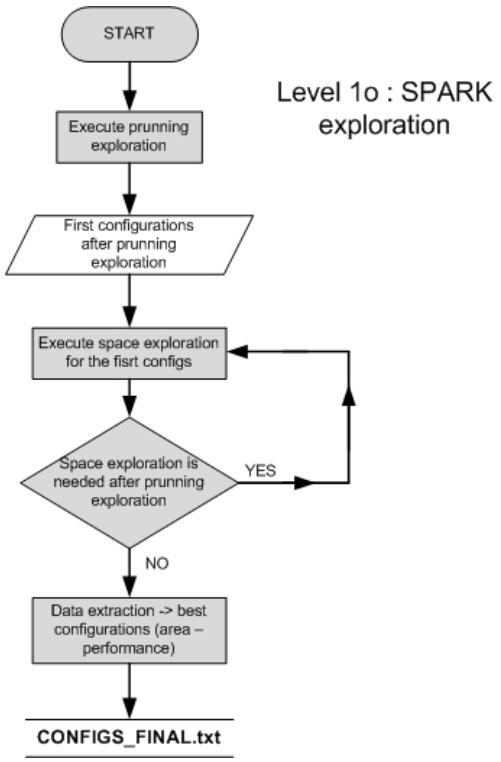


Σχήμα 3.5: Exploration disclaimer 1: Ύπαρξη βέλτιστων λύσεων στην εξερεύνηση με παραμέτρους και code motion.

3.3 Επίπεδο 1ο: HLS C code to RTL code

3.3.1 Γενικά

Το πρώτο επίπεδο της εξερεύνησης αφορά το κομμάτι του HLS όπου διάφοροι πυρήνες σε γλώσσα C εισάγονται στο εργαλείο του SPARK για να παραχθεί ο κώδικας RTL σε vhdl. Σκοπός είναι να εξαχθούν οι καλύτερες παράμετροι από αυτή την εξερεύνηση με κύριους γνώμονες την απόδοση που μετριέται σε κύκλους εκτέλεσης και την επιφάνεια υλικού η οποία είναι ανάλογη με το πλήθος των λειτουργικών πόρων/μονάδων που θα είναι διαθέσιμες για τη σχεδίαση. Βάσει των παραμέτρων αυτών παράγεται η αντίστοιχη RTL περιγραφή θα περάσει στο επόμενο επίπεδο, αυτό της σύνθεσης για περαιτέρω εξερεύνησης
Μια πρώτη, αφαιρετική εικόνα της εξερεύνησης που ακολουθείται παρουσιάζεται στο σχήμα 3.6. Μια πιο αναλυτική επεξήγηση για την εξερεύνηση ακολουθεί.



Σχήμα 3.6: Αφαιρετικό σχεδιάγραμμα ροής για την εξερεύνηση στο επίπεδο του HLS.

3.3.2 Μεθοδολογία Εξερεύνησης του Χώρου Σχεδίασης

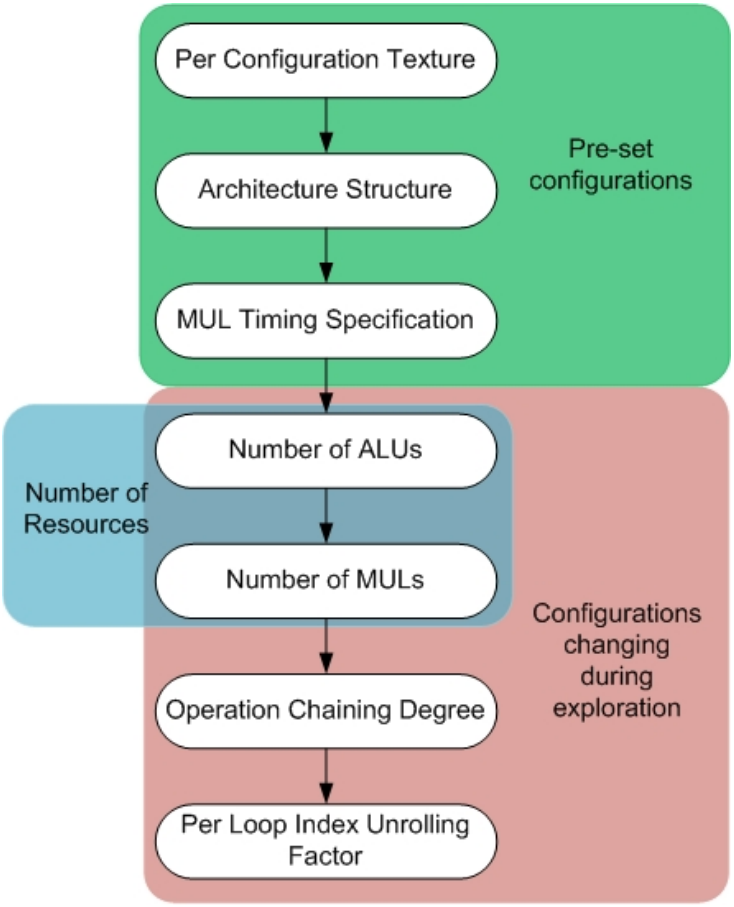
Κύριος στόχος της εξερεύνησης που υλοποιήθηκε ήταν να αποφευχθεί η εξαντλητική (και προφανώς) χρονοβόρα) διαδικασία ελέγχου όλων των δυνατών συνδυασμών για την εύρεση των καλύτερων παραμέτρων σε θέμα απόδοσης (performance) και επιφάνειας υλικού (area).

Οι παράγοντες που είχαμε για εξερεύνηση σ' αυτό το επίπεδο της ροής είναι αποφάσεις από το επίπεδο αρχιτεκτονικής (σχήμα 3.2) και το επίπεδο μετασχηματισμών κώδικα (σχήμα 3.1). Συγκεκριμένα έχουμε:

- **Αριθμός λογικών μονάδων - αθροιστών ALUs**
- **Αριθμός πολλαπλασιαστών MULs**

- **Βαθμός εκτέλεσης λειτουργιών σε ένα κύκλο (Operation Chaining Degree)**
- **Διεύρυνσης του κώδικα για κάθε δείκτη βρόγχου (Loop Unrolling)**

Η σειρά με την οποία γίνεται η εξερεύνηση και η διάταξη των αποφάσεων δεν είναι τυχαία και φαίνεται στο σχήμα που ακολουθεί (σχήμα 3.7). Η προτεινόμενη εξερεύνηση του χώρου σχεδίασης πραγματοποιήθηκε λαμβάνοντας υπόψη την ταξινόμηση των δέντρων αποφάσεων. Η ταξινόμηση αυτή ορίζει τη σειρά προτεραιότητας βάσει της οποίας θα ελεγχθεί η κάθε δυνατή λύση. Ο καθορισμός της προτεραιότητας ορίσθηκε βάσει της βαρύτητας *(i)* του κάθε δέντρου αποφάσεων στην τελική υλοποίηση και *(ii)* των περιορισμών που διαδίδονται από κατηγορία σε κατηγορία.



Σχήμα 3.7: Προτεραιότητα παραμέτρων για εξερεύνηση στο επίπεδο του HLS.

► Αρχικά υπάρχει η απόφαση για την παράμετρο ***(Per Configuration Texture)*** που αφορά τους πολλαπλασιασμούς μόνο με σταθερούς όρους είτε και με μεταβλητές.

► Στη συνέχεια ακολουθεί η απόφαση για την παράμετρο της αρχιτεκτονικής δομή ***(Architecture Stucture)*** των λειτουργικών πόρων που θα χρησιμοποιηθούν στο κύκλωμα. Αυτή η παράμετρος έπεται της παραμέτρου για το Per Configurable Texture γιατί ανάλογα με την απόφαση αν θα έχουμε πολλαπλασιασμούς μόνο με σταθερούς όρους ή και με μεταβλητούς αλλάζει και η δομή των λειτουργικών μονάδων ανάλογα. Δηλαδή υπάρχει η δυνατότητα να επιλεχθεί η ύπαρξη πολλαπλασιασμού μόνο σταθερών όρων και σαν δομή οποιαδήποτε από τις τρεις επιλογές που φαίνονται και στο σχήμα 3.1 χωρίς όμως να είναι αναγκαία η ύπαρξη πόρων πολλαπλασιαστών αφού η λειτουργία πολλαπλασιασμού με σταθερούς όρους γίνεται και μόνο από ολισθήσεις

και προσθέσεις (shifts and additions). Αντιθέτως επιλέγοντας από την προηγούμενη παράμετρο δυνατότητας πολλαπλασιασμού και μεταβλητών όρων αποκλείεται η επιλογή για ύπαρξη μόνο αθροιστών. Ο καθορισμός για το είδος αρχιτεκτονικής δομής κάθε λειτουργικής μονάδας γίνεται με ορισμό των δομικών μονάδων σαν παράμετρος στο εργαλείο του SPARK.

► Επόμενη για επιλογή απόφαση αφορά την παράμετρο χρονισμού της λειτουργίας πολλαπλασιαστή ***(MUL Timing Specification)***, αν δηλαδή ο χρόνος εκτέλεσης της θα διαρκεί ένα κύκλο (single cycle) ή πολλούς κύκλους (multi-cycle). Αυτή η επιλογή αν και υπάρχει στο επίπεδο της Σύνθεσης κυκλώματος (σχήμα 3.3) καθορίζεται και στο επίπεδο του HLS. Χρησιμοποιήθηκε ώστε να υπάρχουν καλύτερα και πιο ακριβή αποτελέσματα από την εξερεύνηση στο 1ο επίπεδο.

► Η απόφαση για τον αριθμό των λειτουργικών πόρων ***(Number of resourses)*** αποτελεί την επόμενη παράμετρο και την πρώτη με την οποία υλοποιείται και η εξερεύνηση. Αυτή διαφοροποιείται στην περίπτωση που υπάρχουν ξεχωριστές λειτουργικές μονάδες ALU και ξεχωριστές μονάδες πολλαπλασιαστών MULs. Με την επιλογή του αριθμού των λειτουργικών μονάδων καθορίζεται και η επιφάνεια υλικού που θα έχει το σχέδιο, ταυτόχρονα όμως επηρεάζεται και η απόδοση του. Για την εκτίμηση της επιφάνειας υλικού θεωρήθηκε το εξής μοντέλο:

$$\mathrm{N}umof\mathrm{ALU}s \cdot Area_{ALU} + \mathrm{N}umof MULs \cdot Area_{MUL} = TotalArea \tag{3.1}$$

Για την εκτίμηση της επιφάνειας υλικού χρησιμοποιήθηκε το παραπάνω μοντέλο για το λόγο ότι μοντελοποιούνται αλγόριθμοι DSP και τα κύρια κομμάτια του κυκλώματος (αυτά που έχουν τη μεγαλύτερη επιφάνεια υλικού και κατανάλωση ισχύος) που προκύπτουν από τη σύνθεση και αποτελούν το σχέδιο αφορούν την μοντελοποίηση του Datapath, δηλαδή τον αριθμό των λειτουργικών μονάδων και όχι τόσο το κομμάτι του Control, που λαμβάνει μικρή επιφάνεια υλικού και κατανάλωση ισχύος.
Η εξερεύνηση ξεκινά από κάποιο σταθερό αριθμό ALU και MUL μονάδων, και τις οποίες αυξάνει ελέγχοντας κάθε φορά την απόδοση που επιτυγχάνεται αλλά την επιφάνεια που καθορίζεται από των αριθμό των λειτουργικών μονάδων.

Πιο συγκεκριμένα αφού καθοριστεί ο ελάχιστος αλλά και μέγιστος αριθμός διαθέσιμων ALUs και MULs, εκτελείται η εξερεύνηση αυξάνοντας αρχικά μόνο των αριθμό των ALUs. Όταν εντοπιστεί πως δεν επιτυγχάνεται βελτίωση (βλέπε Παράγραφο: Αλγόριθμος Εξερεύνησης), αυξάνεται κατά ένα και τον αριθμό των MUL και συνεχίζεται η εξερεύνηση αυξάνοντας τις ALUs. Η αύξηση των ALUs προηγείται αυτής των MULs και αυτό γιατί οι περισσότερες πράξεις αφορούν αθροίσεις, ολισθήσεις και έτσι εξαντλείται η δυνατότητα εκμετάλλευσης των ALUs πριν αυξηθεί σε κάθε περίπτωση ο αριθμός των MULs.

► Στη σειρά εξερεύνησης των αποφάσεων επόμενη είναι η ο βαθμός chaining λειτουργιών ***(Operating Chaining Degree)***. Σε αυτή την παράμετρο υπάρχουν ως επιλογές το 0 που δείχνει καθόλου chaining μέχρι και το βαθμό 3 που δείχνει ότι 4 λειτουργίες που ήταν single-cycle, εκτελούνται πλέον σε ένα κύκλο.

Αυτή η παράμετρος έπεται σε προτεραιότητα της αύξησης των λειτουργικών πόρων, και αυτό γιατί πετυχαίνοντας τη μέγιστη δυνατή βελτίωση με αύξηση των resources (αφού με αύξηση του αριθμού ALUs MULs δε επιτυγχάνεται καλύτερο performance) σημαίνει ότι υπάρχουν αρκετοί λειτουργικοί πόροι για να γίνει και chaining (συγχώνευση σύνθετων λειτουργιών σε ένα κύκλο). Δε θα είχε νόημα να γίνεται αύξηση του παράγοντα chaining πριν εξερευνηθεί ο αριθμός των λειτουργικών πόρων που διατίθεται, να έχει δηλαδή επιτευχθεί βέλτιστη απόδοση αυξάνοντας τον αριθμό των λειτουργικών πόρων, αφού με λίγους πόρους το chaining δε μπορεί να υλοποιηθεί.

► Τελευταίος παράγοντας είναι ο παράγοντας διεύρυνσης κώδικα ανά δείκτη βρόγχου ***(Per Loop Index Unrolling Factor)*** ο οποίος ξεκινά από το μηδέν που σημαίνει και καθόλου unrolling μέχρι και $n-1$ όπου n ο μέγιστος βαθμός επανάληψης που έχει ο κάθε δείκτης βρόγ-

χου. Άρα θέτοντας το unrolling σε $n-1$ έχουμε πλήρη διεύρυνση του κώδικα του βρόγχου.

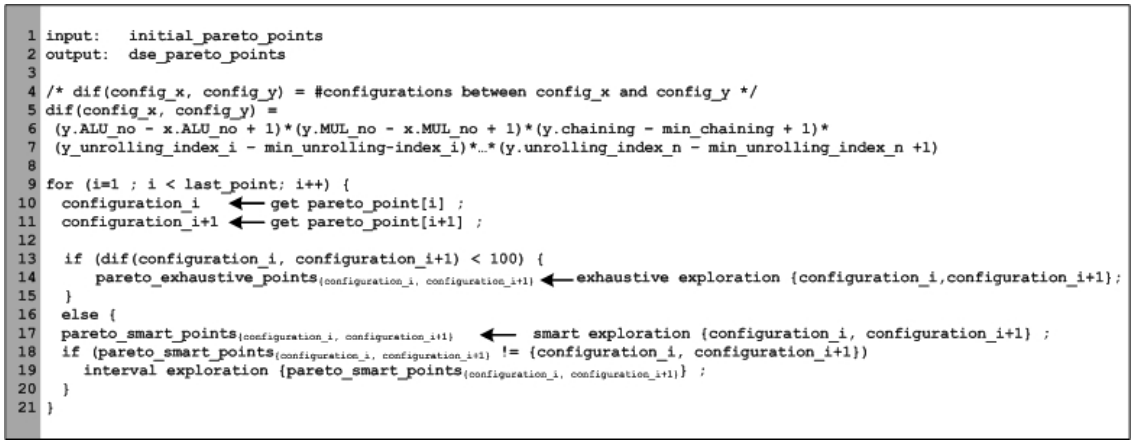
Με εφαρμογή chaining μπορούν να δεσμευτούν οι επιπλέον διαθέσιμες λειτουργικές μονάδες, για να συγχωνεύσει σύνθετες λειτουργίες σ' ένα κύκλο ρολογιού). Από την άλλη, το unrolling αυξάνει τον αριθμό των διαθέσιμων λειτουργιών (operation) (αφού γίνεται "διεύρυνση" κώδικα). Η απόφαση λοιπόν, για το βαθμό του unrolling έπεται αυτήν για το chaining. Έχοντας εξερευνήσει και πετύχει βέλτιστη απόδοση με βαθμό chaining, που μεταφράζεται ότι κάθε περαιτέρω αύξηση του παράγοντα αυτού δε δίνει καλύτερη λύση, σημαίνει πως υπάρχουν διαθέσιμες λειτουργικές μονάδες σε κάθε κύκλο ώστε να "απορροφήσουν", δηλαδή να εκτελέσουν αρκετές από τις επιπλέον λειτουργίες που θα επιφέρει αύξηση του παράγοντα unrolling.
Μια αντίστροφη τοποθέτηση των αποφάσεων, δηλαδή η απόφαση για το unrolling να προηγείται αυτήν του chaining δε θα είχε νόημα. Σε μια τέτοια περίπτωση, η εξερεύνηση και εύρεση βέλτιστων λύσεων με unrolling, σημαίνει πως έχει γίνει πλήρης εκμετάλλευση των διαθέσιμων λειτουργικών μονάδων για τις επιπλέον λειτουργίες που επιφέρει το unrolling και συνεπώς αλλαγή του βαθμού chaining δε θα έχει κάποιο αποτέλεσμα αφού δεν θα υπάρχουν διαθέσιμες λειτουργικές μονάδες για την υλοποίηση και του unrolling και του chaining.

Ο παρακάτω ψευδοκώδικας (Σχήμα 3.8) παρουσιάζει την προτεινόμενη σειρά εξερεύνησης όπως περιγράφεται και στα Σχήμα 3.7.

```
1  input:    kernel.c
2  output:   optimized configuration vectors {#ALUs, #MULs, #chaining degree, #unrolling_index_i,…, #unrolling_index_n}
3
4  temp_min = min number ALUs;
5
6  for(unrolling_index_i=0 ; unrolling_index_i < max_unrolling_index_i ; unrolling_index_i++) {
7  …
8    for(unrolling_index_n=0 ; unrolling_index_n < max_unrolling_index_n ; unrolling_index_n++) {
9
10     CDFG{ #unrolling_index_i, .., #unrolling_index_n}  ← C2CDFG(kernel.c) ;
11
12       for(chaining=min_chaining ; chaining < max_chaining ; chaining++) {
13         for(MUL_no=min_MULs ; MUL_no < max_MULs ; MUL_no++) {
14           for(ALU_no=min_ALUs ; ALU_no < max_ALUs ; ALU_no++) {
15
16                sched_CDFG     ← schedule (CDFG{ #unrolling_index_i, .., #unrolling_index_n} , #chaining, #MUL_no, #ALU_no) ;
17                xi.{#ALU_no, #MUL_no, #chaining, #unrolling_index_i, …, #unrolling_index_n} ;
18                #contol_steps  ← extract control steps (sched_CDFG) ;
19                #total_area    ← area estimation (#ALU_no, #MUL_no) ;
20                grat           ← evaluate gradient ((#total_area,#control_steps), depth_k) ;
21                                 switch grat
22                                 case   (grat < 0)  : insert (initial_curve, xi) ;i++ ;
23                                 case (grat ≥ 0)  : temp_min = x[i-depth+1].#ALU_no ;i++ ;  break ;
24             }
25           }
26         }
27       }
28     …
29   }
```

Σχήμα 3.8: Ψευδοκώδικας για το smart exploration.

Στη συνέχεια παρουσιάζεται ο ψευδοκώδικας για το *interval exploration* με σταθερό βάθος εξερεύνησης (Σχήμα 3.9), αλλά και με μεταβλητό βάθος εξερεύνησης (ανάλογα με το επίπεδο εξερεύνησης που εκτελείτε) (Σχήμα 3.10).



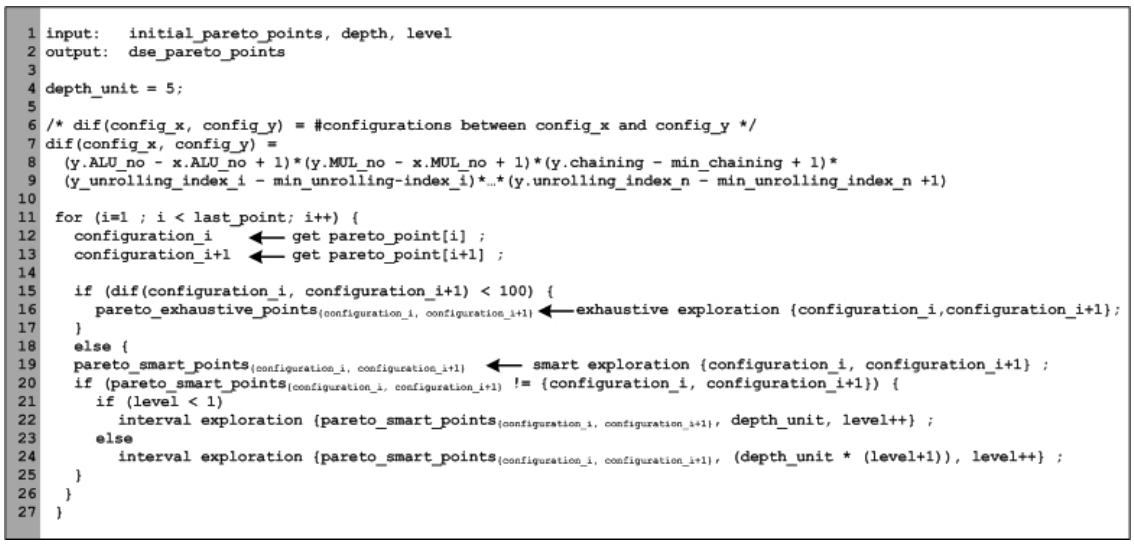
```
input:   initial_pareto_points
output:  dse_pareto_points

/* dif(config_x, config_y) = #configurations between config_x and config_y */
dif(config_x, config_y) =
 (y.ALU_no - x.ALU_no + 1)*(y.MUL_no - x.MUL_no + 1)*(y.chaining - min_chaining + 1)*
 (y_unrolling_index_i - min_unrolling-index_i)*…*(y.unrolling_index_n - min_unrolling_index_n +1)

for (i=1 ; i < last_point; i++) {
  configuration_i   ← get pareto_point[i] ;
  configuration_i+1 ← get pareto_point[i+1] ;

  if (dif(configuration_i, configuration_i+1) < 100) {
      pareto_exhaustive_points{configuration_i, configuration_i+1} ←exhaustive exploration {configuration_i,configuration_i+1};
  }
  else {
  pareto_smart_points{configuration_i, configuration_i+1}   ←  smart exploration {configuration_i, configuration_i+1} ;
  if (pareto_smart_points{configuration_i, configuration_i+1} != {configuration_i, configuration_i+1})
    interval exploration {pareto_smart_points{configuration_i, configuration_i+1}} ;
  }
}
```

Σχήμα 3.9: Ψευδοκώδικας για το interval exploration με σταθερό βάθος εξερεύνησης.



```
input:   initial_pareto_points, depth, level
output:  dse_pareto_points

depth_unit = 5;

/* dif(config_x, config_y) = #configurations between config_x and config_y */
dif(config_x, config_y) =
  (y.ALU_no - x.ALU_no + 1)*(y.MUL_no - x.MUL_no + 1)*(y.chaining - min_chaining + 1)*
  (y_unrolling_index_i - min_unrolling-index_i)*…*(y.unrolling_index_n - min_unrolling_index_n +1)

 for (i=1 ; i < last_point; i++) {
   configuration_i   ← get pareto_point[i] ;
   configuration_i+1 ← get pareto_point[i+1] ;

   if (dif(configuration_i, configuration_i+1) < 100) {
     pareto_exhaustive_points{configuration_i, configuration_i+1} ←exhaustive exploration {configuration_i,configuration_i+1};
   }
   else {
   pareto_smart_points{configuration_i, configuration_i+1}   ←  smart exploration {configuration_i, configuration_i+1} ;
   if (pareto_smart_points{configuration_i, configuration_i+1} != {configuration_i, configuration_i+1}) {
     if (level < 1)
       interval exploration {pareto_smart_points{configuration_i, configuration_i+1}, depth_unit, level++} ;
     else
       interval exploration {pareto_smart_points{configuration_i, configuration_i+1}, (depth_unit * (level+1)), level++} ;
   }
  }
 }
```

Σχήμα 3.10: Ψευδοκώδικας για το interval exploration με βάθος εξερεύνησης να μεταβάλλεται ανάλογα με το επίπεδο της εξερεύνησης.

Αλγόριθμος εξερεύνησης

Όπως φαίνεται και στο ψευδοκώδικα (Σχήμα 3.8) ο εσωτερικός βρόγχος αφορά τον αριθμό των ALUs και ο αμέσως επόμενος τον αριθμό MULs. Αυτοί οι δύο βρόγχοι διαμορφώνουν και την επιφάνεια υλικού, το ένα από τους δύο παράγοντες σύγκρισης (ο άλλος είναι η απόδοση). Η εξερεύνηση που υλοποιείται ξεκινά με ένα δεδομένο αριθμό λειτουργικών πόρων (ALUs και MULs) και αυξάνει κατά ένα κάθε φορά τον αριθμό των ALUs σημειώνοντας την επιφάνεια υλικού (area) και την απόδοση (performance) για τις συγκεκριμένες επιλογές στις παραμέτρους (αριθμό ALUs και MULs, chaining, unrolling). Σε κάθε συνδυασμό παραμέτρων που δίνει μία επιφάνεια υλικού και απόδοση, ελέγχεται και συγκρίνεται με τον αριθμό των κύκλων που έδωσε ο ακριβώς προηγούμενος αριθμός παραμέτρων (ο συνδυασμός παραμέτρων με ένα λιγότερο ALUs). Στην περίπτωση που βρεθεί ο ίδιος αριθμός κύκλων στην μέτρηση x_i με τον συνδυασμό x_{i-1} τότε σε ένα πίνακα διατηρούνται και οι δύο αριθμοί παραμέτρων, και συνεχίζεται η αύξηση κατά ένα των ALUs. Αν στην επόμενη μέτρηση x_{i+1} βρεθεί μικρότερος αριθμός κύκλων από τον συνδυασμό x_i τότε ο πίνακας που διατηρούσε τους συνδυασμούς με ίδια απόδοση (αριθμό κύκλων) αδειάζει και σαν πρώτη εγγραφή εισάγεται η μέτρηση x_{i+1}. Στη συνέχεια παράγονται

όπως αναφέρθηκε οι μετρήσεις για τον επόμενο συνδυασμό, συγκρίνονται με την εγγραφή που υπάρχει στον πίνακα, και στην περίπτωση που έχουν τον ίδιο αριθμό κύκλων εισάγεται η τρέχουσα εγγραφή σαν δεύτερη στον πίνακα. Αν ο αριθμός κύκλων του τρέχον συνδυασμού είναι μικρότερος από αυτόν που διατηρείται στον πίνακα, τότε ο πίνακας αδειάζει και εισάγεται σαν πρώτη εγγραφή ο τρέχον συνδυασμός Έτσι σε κάθε στιγμή στον "πίνακα ελέγχου" θα υπάρχει είτε μία εγγραφή (που σημαίνει ότι είχε μικρότερους κύκλους από τον προηγούμενο συνδυασμό) είτε πολλούς συνεχόμενους συνδυασμούς με τον καθένα να έχει ίσο ή μεγαλύτερο αριθμό κύκλων. Έτσι ελέγχονται τα συνεχόμενα σημεία που σχηματίζουν ευθείες με μηδενικές κλήσεις (σημεία με ίδιο αριθμό κύκλων) είτε θετικές κλήσεις (σημεία με το καθένα μεγαλύτερο αριθμό κύκλων το καθένα από το πρώτο του πίνακα).

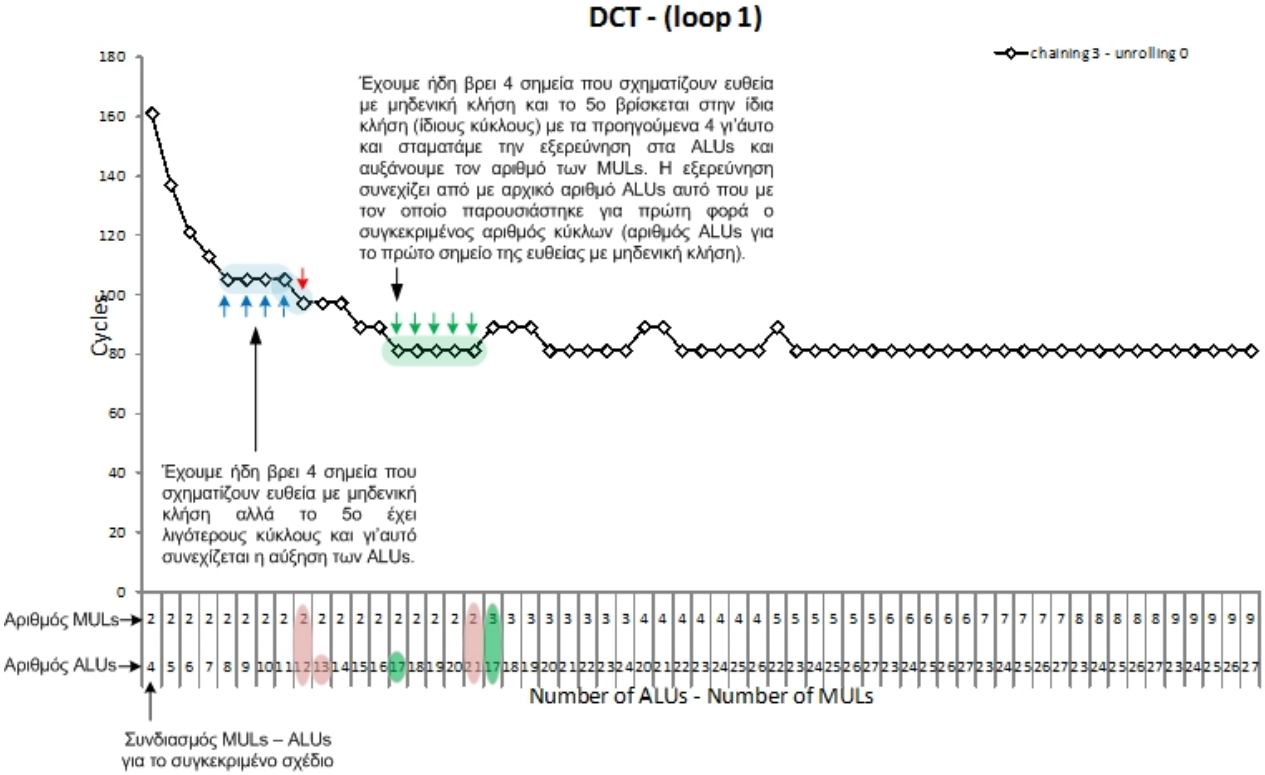
Ο μέγιστος αριθμός εγγραφών που θα διατηρούνται στον πίνακα προέκυψε από αρκετές πειραματικές μετρήσεις. Στην υλοποίηση χρησιμοποιείται μέγιστος αριθμός εγγραφών στον "πίνακα ελέγχου" ίσος με 4 και έτσι ελέγχεται σε "βάθος" 5 σημείων για ευθείες με μηδενική κλήση ή θετική. Αναφέρεται ότι ελέγχεται σε "βάθος" 5 σημείων γιατί, έχοντας 4 εγγραφές στον πίνακα ελέγχου, λαμβάνεται υπόψη ο αριθμός κύκλων που δίνει ο πέμπτος συνδυασμός παραμέτρων και αναλόγως συνεχίζεται η αύξηση των ALUs ή σταματά στο συγκεκριμένο αριθμό ALUs και αυξάνεται ο αριθμός των MULs. Ο αριθμός για το "βάθος ελέγχου μηδενικής κλήσης" (αναφέρεται έτσι γιατί ως επί το πλείστο υπάρχουν ευθείες με μηδενικές κλήσεις και όχι θετικές κλήσεις) είναι παραμετροποιήσιμος, και καθορίζει και την ταχύτητα της εξερεύνησης σε πολύ μεγάλο βαθμό.

Από τις πειραματικές μετρήσεις φάνηκε πως έχοντας μικρό "βάθος ελέγχου μηδενικής κλήσης" η εξερεύνηση ολοκληρωνόταν σε μικρό χρονικό διάστημα, χάνοντας όμως σημαντικό αριθμό βέλτιστων λύσεων και έτσι έδινε χαμηλό ποσοστό εύρεσης βέλτιστων λύσεων σε σύγκριση με τη εξαντλητική εξερεύνηση. Στην αντίθετη περίπτωση που θετόταν μεγάλο "βάθος ελέγχου μηδενικής κλήσης" υπήρχε σχεδόν εύρεση όλων των βέλτιστων λύσεων αλλά ο χρόνος εκτέλεσης της εξερεύνησης ήταν πολύ μεγάλος, κάτι που δεν εξυπηρετεί ιδιαίτερα, αφού σκοπός είναι μια γρήγορη αλλά και αποδοτική εξερεύνηση. Έτσι καθορίστηκε το "βάθος ελέγχου μηδενικής κλήσης" στο 5 (έχοντας τη δυνατότητα αλλαγής για διάφορους κώδικες).

Στην περίπτωση που βρεθεί ευθεία με μηδενική κλήση σε βάθος 5 στοιχείων, η αύξηση του αριθμού ALUs που βρίσκεται στον εσωτερικό βρόγχο σταματά, και αυξάνεται κατά ένα ο αριθμός των MULs και συνεχίζεται η εξερεύνηση με αύξηση τον αριθμό των ALUs αρχίζοντας όμως από τον αριθμό ALUs που είχε η πρώτη εγγραφή στον πίνακα ελέγχου. Με αυτό τον τρόπο αποφεύγονται οι επιπλέον υπολογισμοί που θα υπήρχαν αν ξεκινούσε η εξερεύνηση με αριθμό των ALUs από τον default αρχικό. Επίσης ξεκινώντας από τον αριθμό των ALUs της πρώτης εγγραφής του πίνακα, δε χάνονται τυχών βέλτιστες λύσεις που μπορούν να παρουσιαστούν σ' αυτό το διάστημα.

Στο σχήμα 3.11 με μπλε χρώμα εμφανίζονται 4 συνδυασμοί που είναι βρίσκονται στον πίνακα ελέγχου αφού σχηματίζουν ευθεία με μηδενική κλήση. Το 5 σημείο με κόκκινο χρώμα θα καθορίσει και την συνέχεια της εξερεύνησης. Όπως φαίνεται έχει λιγότερους κύκλους από το προηγούμενο του και γι' αυτό η εξερεύνηση συνεχίζεται με αύξηση του αριθμού των ALUs (όπως φαίνεται και στους συνδυασμούς ALUs και MULs στον οριζόντιο άξονα με ελαφρύ κόκκινο χρώμα.

Στη συνέχεια της εξερεύνησης με πράσινο χρώμα σημειώνονται 5 σημεία που έχουν μηδενική κλήση. Έτσι όπως φαίνεται και στον συνδυασμό ALUs και MULs στον οριζόντιο άξονα, η αύξηση του αριθμού των ALUs σταματά (σύγκριση συνδυασμών με 2 MULs και 21 ALUs και 3 MULs και 17 ALUs) και αυξάνεται κατά ένα ο αριθμός των MULs και η εξερεύνηση αρχίζει από τον αριθμό ALUs 17 αφού ήταν ο αριθμός ALUs της πρώτης εγγραφής στην ευθεία μηδενικής



Σχήμα 3.11: Αλγόριθμος έξυπνης εξερεύνησης με αποκοπή σε εύρεση ευθείας με μηδενική κλήση για 5 σημεία.

κλήσης σε βάθος 5.

Ολοκληρωμένη ροή εξερεύνησης

Μετά την ανάλυση του αλγόριθμου για την εξερεύνηση με αποκοπή στο προηγούμενο μέρος, παρακάτω παρουσιάζεται το πλήρες διάγραμμα ροής (flowchart) της υλοποίησης, Σχήμα 3.12.

Η ροή ξεκινά με ένα prunning exploration (με τον αλγόριθμο που περιγράφεται πιο πάνω) για όλο το εύρος του αριθμού των λειτουργικών πόρων (αριθμό ALUs και MULs), για όλο το εύρος για το βαθμό chaining (από 0 μέχρι και 3) αλλά χωρίς καθόλου αλλαγή στο βαθμό του unrolling που εφαρμόζεται στον κώδικα, οποίος είναι μηδέν (καθόλου unrolling).

Σαν αποτέλεσμα γι' αυτό παράγεται ένα αρχείο εξόδου, *output_file.txt*, το οποίο περιέχει όλα τα δεδομένα-μετρήσεις από την πρώτη prunning εξερεύνηση. Ο αριθμός του αρχείου εξόδου που παράχθηκε προστίθεται στο αρχείο *final_files.txt*, στο οποίο θα διατηρούνται οι αριθμοί "ταυτότητας" όλων των αρχείων εξόδου που θα παράγονται για την τελική διαλογή των βέλτιστων σημείων.

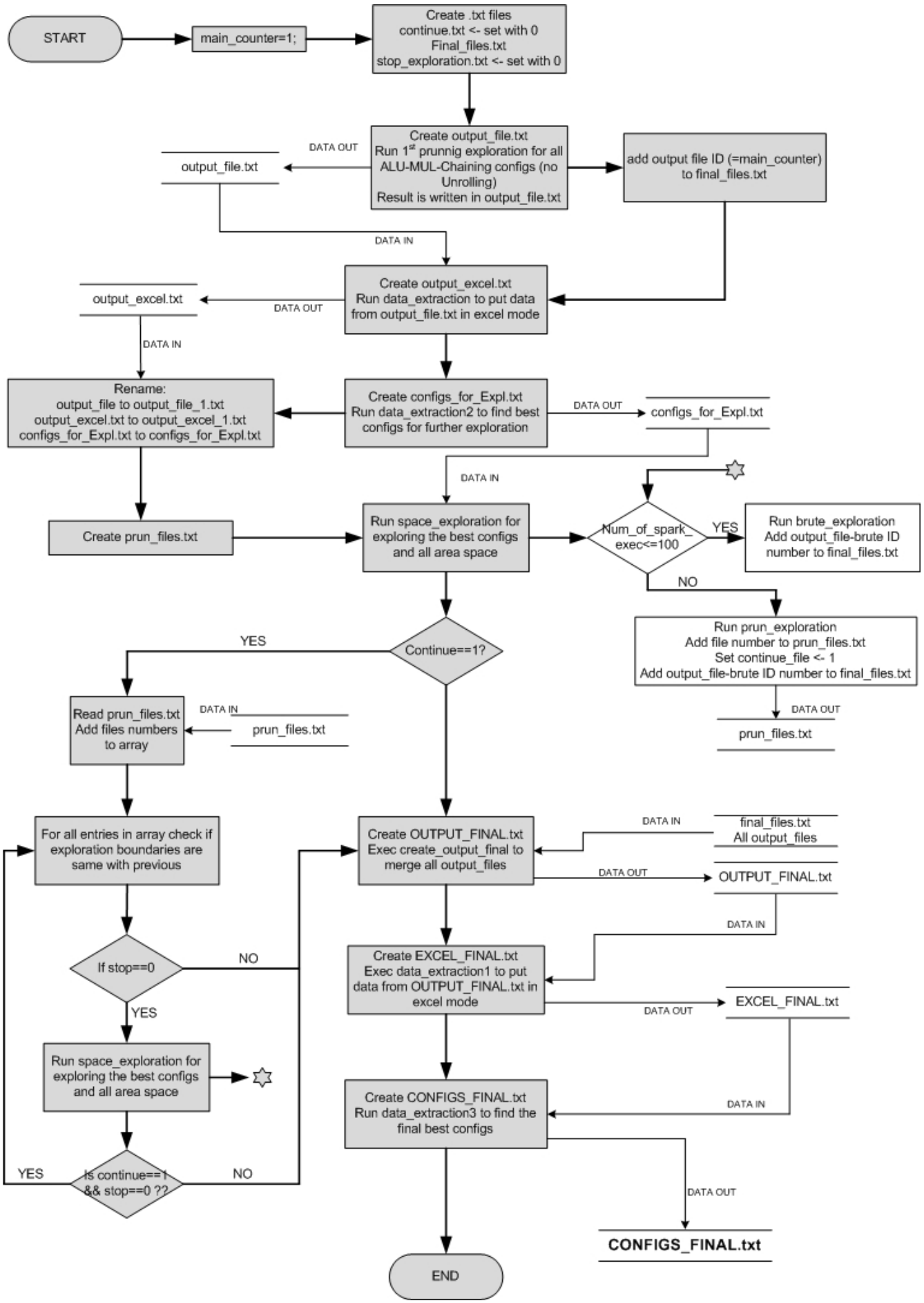
Συνεχίζοντας στη ροή της εξερεύνησης, το αρχείο εξόδου που παράχθηκε περνά από διαμόρφωση (από το script *data_extraction*) και τα αποτέλεσμα είναι στη μορφή excel. Αυτό το excel αρχείο περνά από περαιτέρω επεξεργασία και με το script *data_extraction2* παράγεται ένα αρχείο που περιέχει τα καλύτερα σημεία (συνδυασμούς παραμέτρων) από την πρώτη prunning εξερεύνηση (αρχείο *configs_for_Expl.txt*).

Στη συνέχεια με βάση αυτά τα καλύτερα σημεία εκτελείτε εξερεύνηση "χώρου" (*interval_exploration*) μεταξύ κάθε δύο σημείων και από την οποία παράγονται αρχεία εξόδου είτε με

brute exploration είτε με smart (prunning) exploration. Στις συγκεκριμένες εξερευνήσεις χρησιμοποιούμε σαν όρια για τον αριθμό των λειτουργικών πόρων αυτά που αναφέρουν τα σημεία από το αρχείο *configs_for_Expl.txt* και πλήρης εξερεύνηση για τις παραμέτρους chaining και unrolling. Οι αριθμοί όλων των αρχείων εξόδου που παράγονται εισάγονται στο αρχείο *final_files.txt* που αναφέρονται και πιο πάνω. Επιπλέον τα αρχεία που παράγονται από smart (prunning) exploration χρειάζονται περαιτέρω εξερεύνηση γι' αυτό και ο κωδικός αριθμός τους εισάγεται σ'ενα αρχείο (*prun_files.txt*), από το οποίο στη συνέχεια θα διαβαστούν τα αποτελέσματα και θα παραχθούν με χρήση των scripts *data_exploration* και *data_exploration2* τα καλύτερα σημεία.

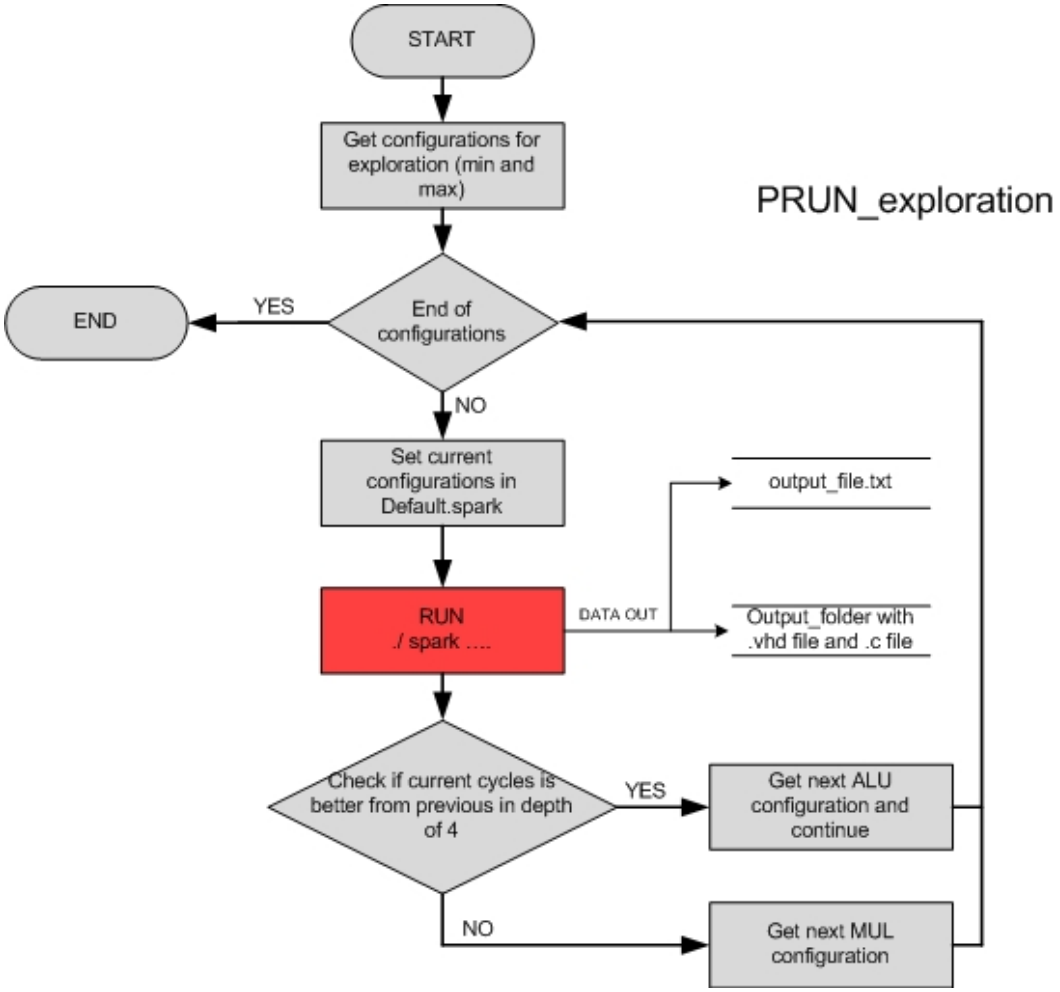
Αυτά τα σημεία που παράχθηκαν από κάθε prunning εξερεύνηση ελέγχονται αν συμπίπτουν με αυτά τις προηγούμενης εξερεύνησης στο χώρο που καθορίζουν κάθε ζεύγος από σημεία. Αν βρεθούν διαφορετικά σημεία συνεχίζουμε σε μια επιπλέον εξερεύνηση χώρου, ενώ στην περίπτωση που συμπίπτουν η εξερεύνηση για το συγκεκριμένο χώρο σταματά.

Όταν ολοκληρωθεί όλη η εξερεύνηση (δεν βρέθηκαν καλύτερα σημεία για κάθε prunning εξερεύνηση που έγινε σε σχέση με την προηγούμενη που έγινε στο ίδιο χώρο) τότε δημιουργείται ένα ενιαίο αρχείο εξόδου (*OUTPUT_FINAL.txt*), που περιλαμβάνει όλα τα αρχεία εξόδου που παράχθηκαν από κάθε εξερεύνηση που έγινε (prunning ή brute) και τα διαμορφώνουμε σε excel μορφή (*EXCEL_FINAL.txt*). Αυτό το αρχείο περνάει σαν είσοδος στο script *data_extraction3* με το οποίο επιλέγονται τα καλύτερα τελικά σημεία (*CONFIGS_FINAL.txt*).



Σχήμα 3.12: Ολοκληρωμένο διάγραμμα ροής (flowchart) για την εξερεύνηση.

Prunning exploration Στο σχήμα 3.13 παρουσιάζεται μια πιο λεπτομερής εικόνα για την εξερεύνηση prunning που χρησιμοποιείται στο ολοκληρωμένο flow του προγράμματος.

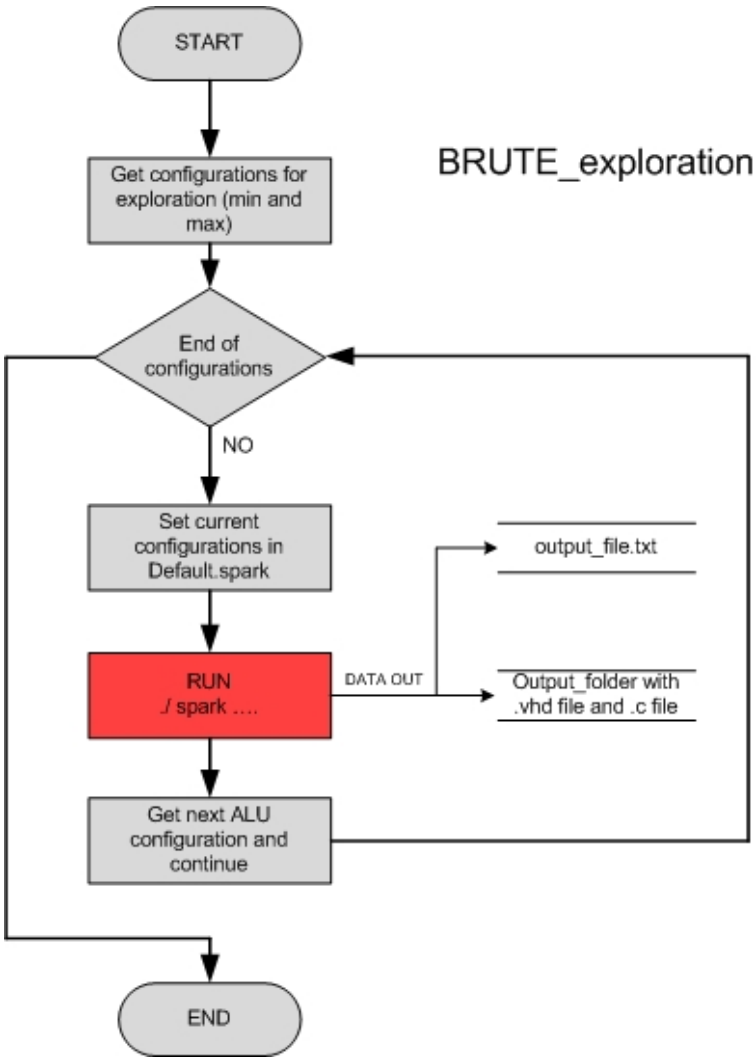


Σχήμα 3.13: Διάγραμμα ροής (flowchart) για την prunning εξερεύνηση.

Όπως φαίνεται και στο σχήμα, για την prunning εξερεύνηση εισάγεται το εύρος των παραμέτρων (το ελάχιστο και το μέγιστο για όλες τις παραμέτρους, αριθμό ALUs, αριθμός MULs, βαθμό chaining, βαθμό unrolling). Ο πρώτος συνδυασμός των παραμέτρων εγγράφεται και στις αντίστοιχες του αρχείου *default.spark* και εκτελείτε η σύνθεση υψηλού επιπέδου (με το λογισμικό SPARK) σύμφωνα με το συγκεκριμένο συνδυασμό παραμέτρων. Ελέγχεται αν παρουσιάστηκε ευθεία 5 σημείων με μηδενική κλήση (όπως περιγράφεται προηγουμένως). Αν όχι (και συνεπώς με το συγκεκριμένο αριθμό ALUs και τον υπόλοιπο συνδυασμό παραμέτρων αντιστοιχούν λιγότεροι κύκλοι λειτουργίας από τον προηγούμενο) τότε συνεχίζεται η αύξηση του αριθμού των ALUs με βάση και το ψευδοκώδικα στο Σχήμα 3.8. Στην περίπτωση που εντοπιστεί ευθεία μηδενικής κλήσης 5 σημείων τότε σταματά η αύξηση στον εσωτερικό βρόγχο, αυτό του αριθμού των ALUs, αυξάνεται ο αριθμός των MULs και συνεχίζεται το exploration με αριθμό ALUs αυτόν που παρουσιάζεται σαν πρώτος στην ευθεία 5 σημείων (σχήμα 3.11).
Η εξερεύνηση εκτελείται και για τους άλλους παράγοντες (chaining και unrolling per index) με το ίδιο μοτίβο να διατηρείται στους 2 εσωτερικούς βρόγχους (που αφορούν αριθμό ALUs και MULs) (ψευδοκώδικας Σχήμα 3.8).

Brute exploration Στη υλοποίηση χρησιμοποιήθηκε και εξαντλητική (exhaustive) εξερεύνηση σε όσα διαστήματα εξερεύνησης (μέσω της εξερεύνησης "χώρου" που αναλύεται παρακάτω) κρίνεται ότι είναι αρκετά μικρά, ώστε να μην είναι αρκετά χρονοβόρα η εξαντλητική εξερεύνηση σ' αυτά. Εφαρμόζοντας εξαντλητική εξερεύνηση σ' αυτά τα διαστήματα, τελειώνει και η οποιαδήποτε περαιτέρω εξερεύνηση (σε αντίθεση με την εξερεύνηση prunning σε διάφορα διαστήματα που τελειώνει όταν βρεθούν ίδια καλύτερα σημεία ή γίνει στα καλύτερα σημεία που προκύπτουν από prunning εξερεύνηση εξαντλητική εξερεύνηση μέσω της εξερεύνησης χώρου).

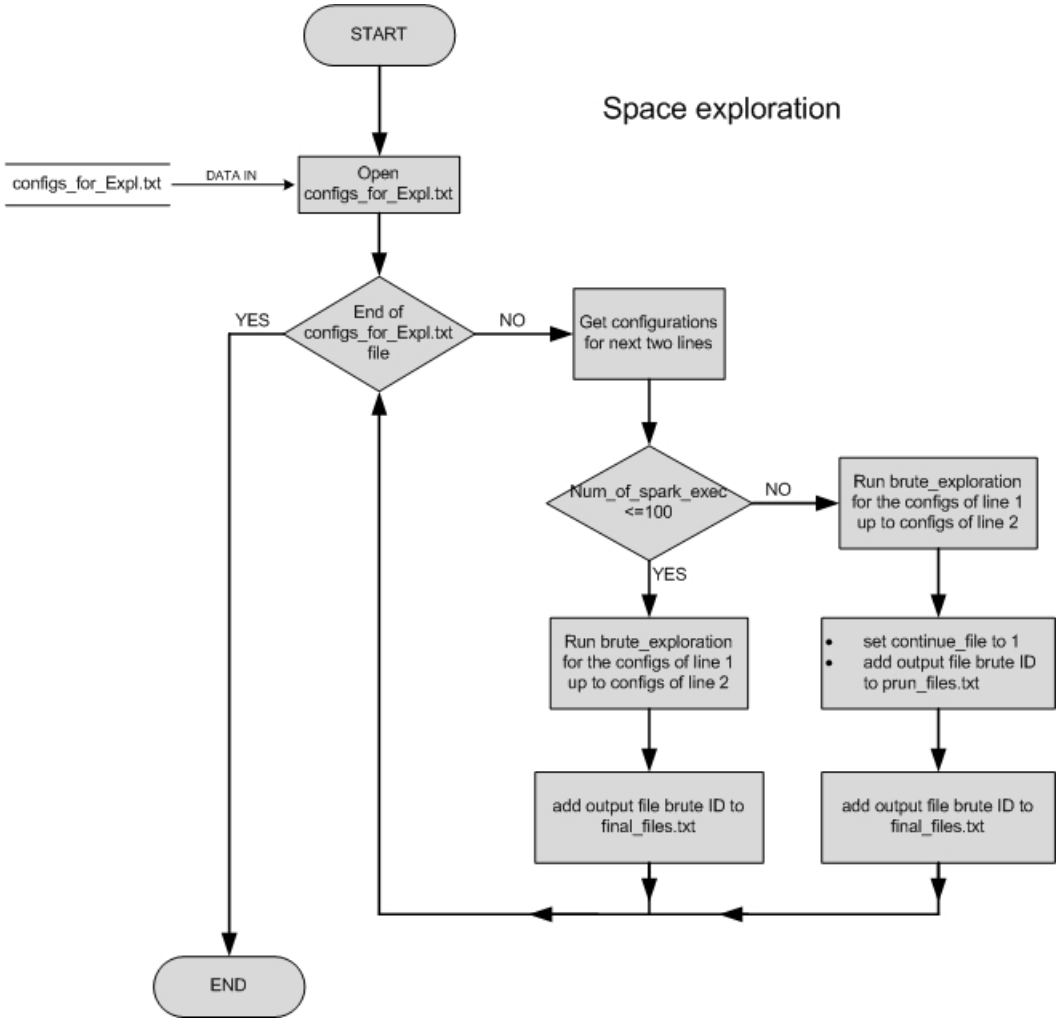
Στο σχήμα 3.14 παρουσιάζεται το διάγραμμα ροής για το μέρος της εξαντλητικής εξερεύνησης. Αφού πάρουμε τα αρχικές τιμές για όλες τις παραμέτρους (το ελάχιστο και το μέγιστο για όλες τις παραμέτρους, αριθμό ALUs, αριθμός MULs, βαθμό chaining, βαθμό unrolling) για κάθε ένα συνδυασμό από τις επιλογές εκτελούμε HLS με το λογισμικό SPARK.



Σχήμα 3.14: Διάγραμμα ροής (flowchart) για την εξαντλητική (brute) εξερεύνηση.

Εξερεύνηση χώρου (Interval space exploration) Όπως αναφέρθηκε και σε προηγούμενες παραγράφους μετά από την πρώτη prunning εξερεύνηση (που αφορά μόνο τον αριθμό των ALUs, MULs και το βαθμό chaining) εξάγονται τα καλύτερα σημεία (συνδυασμοί παραμέτρων) και χρησιμοποιείται το interval space exploration (εξερεύνηση χώρου) για την περαιτέρω εξε-

ρεύνηση με τον παράγοντα του unrolling. Η εξερεύνηση στα διαστήματα που δημιουργούνται μεταξύ των βέλτιστων λύσεων από κάθε prunning exploration (και από το το πρώτο) ανάλογα με το πλήθος των συνδυασμών που προκύπτουν είναι prunning ή εξαντλητική (Σχήμα 3.9).

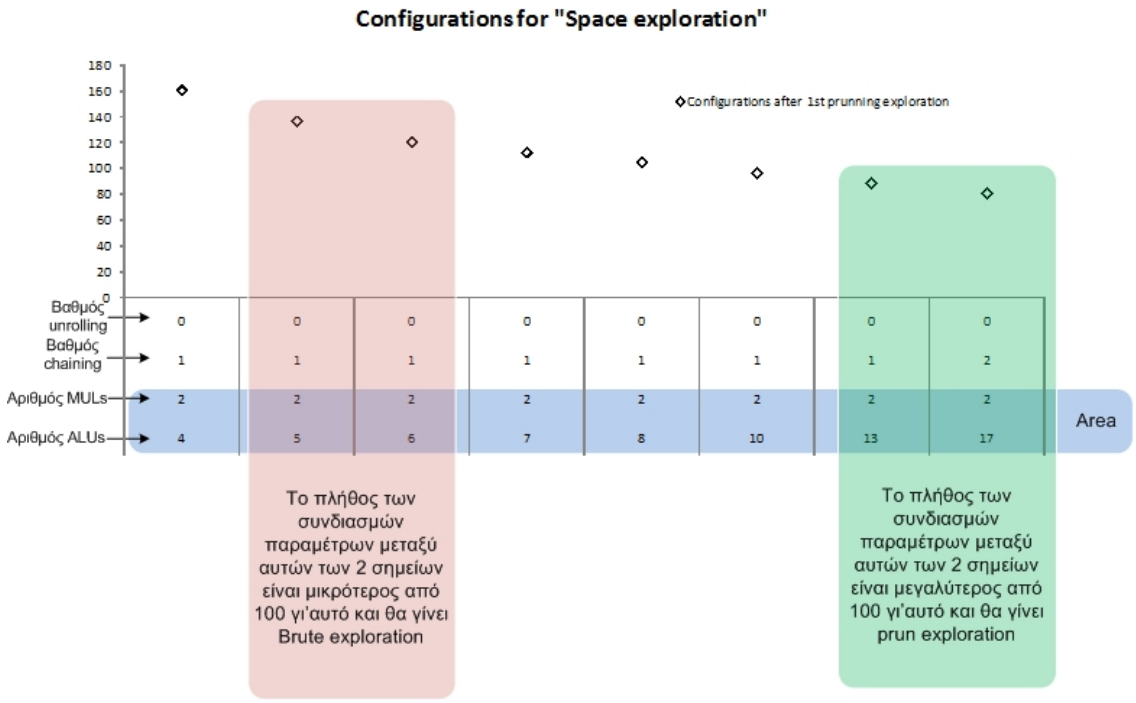


Σχήμα 3.15: Διάγραμμα ροής (flowchart) για την εξερεύνηση χώρου (space exploration).

Όπως φαίνεται και στο σχήμα 3.15 με βάση το πλήθος των συνδυασμών επιλέγεται prunning exploration (για πλήθος συνδυασμών μεγαλύτερο από 100) και εξαντλητική εξερεύνηση (για πλήθος συνδυασμών ίσο ή μικρότερο από 100). Αυτό γίνεται για κάθε ζεύγος συνεχόμενων σημείων-συνδυασμών που βρίσκονται στο αρχείο *conifgs_for_Expl.txt*.

Παράδειγμα για την εξερεύνηση χώρου φαίνεται στο σχήμα 3.16. Συγκεκριμένα τα σημεία που φαίνονται αποτελούν τα καλύτερα σημεία-συνδυασμοί που εξάγονται από το πρώτο prunning exploration με βαθμό unrolling μηδέν. Από αυτά και ανά δύο συνεχόμενα γίνεται περαιτέρω εξερεύνηση και για το βαθμό του unrolling ανάλογα με το πλήθος των συνδυασμών που προκύπτουν. Με ελαφρύ κόκκινο χρώμα παρουσιάζεται περίπτωση 2 διαδοχικών σημείων που το πλήθος των συνδυασμών στο διάστημα μεταξύ τους είναι μικρότερο από 100, γι' αυτό και στο συγκεκριμένο διάστημα εκτελείτε εξαντλητική εξερεύνηση. Με πράσινο χρώμα σημειώνεται μια περίπτωση 2 διαδοχικών σημείων που το πλήθος των συνδυασμών στο διάστημα μεταξύ τους είναι μεγαλύτερο από 100 και στο συγκεκριμένο διάστημα εκτελείται prunning εξερεύνηση.

Σημειώνεται ότι σε κάθε space exploration το τελευταίο σημείο πάντα σχηματίζει διάστημα και



Σχήμα 3.16: Διάγραμμα ροής (flowchart) για την εξερεύνηση χώρου (space exploration).

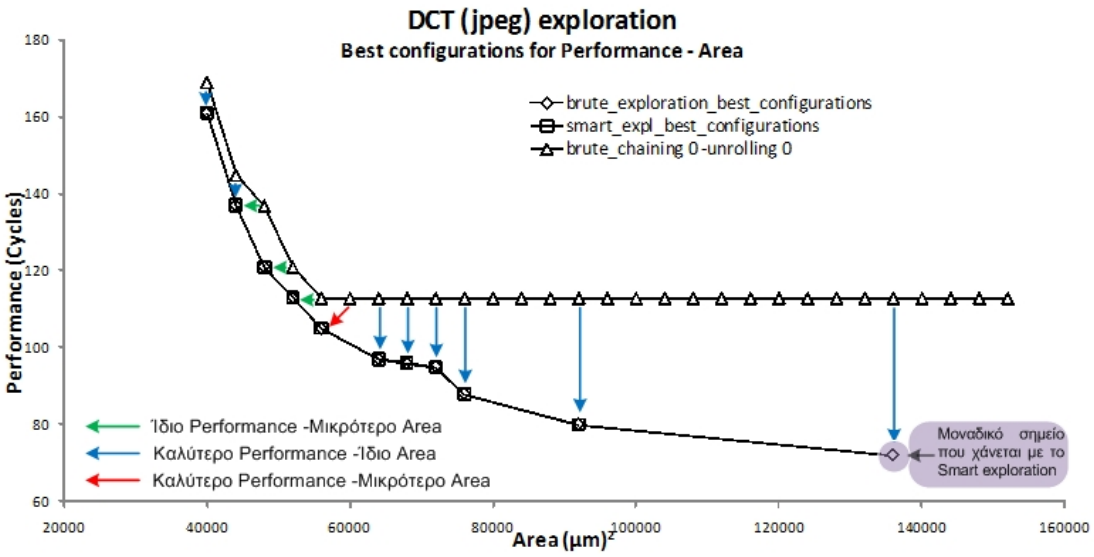
με το σημείο-συνδυασμό που σχηματίζεται από το μέγιστο αριθμό ALUs,MULs και chaining.

Αποτέλεσμα της εξερεύνησης - Εύρεση βέλτιστων σημείων Σαν τελικό αποτέλεσμα από τη εξερεύνηση στο HLS επίπεδο παράγεται το αρχείο *FINAL_CONFIGS.txt* το οποίο περιέχει τα βέλτιστα σημεία-συνδυασμούς παραμέτρων για απόδοση και επιφάνεια υλικού. Συγκρίνοντας αυτό το αποτέλεσμα των βέλτιστων σημείων με τα σημεία που δίνει μία εξερεύνηση που βασίζεται μόνο στον αριθμό των ALUs και MULs φαίνεται καθαρά η ύπαρξη και επισήμανση καλύτερων λύσεων από την εξερεύνηση που υλοποιείται. Αυτά τα σημεία έχουν *(i)* καλύτερη απόδοση με την ίδια επιφάνεια υλικού με τα αντίστοιχα της αρχικής καμπύλης χωρίς chaining και code motions, *(ii)* ίδια απόδοση αλλά μικρότερη επιφάνεια υλικού και *(iii)* καλύτερη απόδοση αλλά και μικρότερη επιφάνεια υλικού. Τέτοιο παράδειγμα παρουσιάζεται και στο σχήμα 3.17.

Στο Σχήμα 3.17 με τη στρατηγική που χρησιμοποιήθηκε για την εξερεύνηση των βέλτιστων λύσεων επιτυχάνεται σχεδόν ολόκληρο το σύνολο των λύσεων χωρίς τη χρήση εξαντλητικής εξερεύνησης με όλες τις αρνητικές συνέπειες που έχει αυτή (κυρίως τεράστιο χρόνο εκτέλεσης). Φαίνεται πως και η εξαντλητική αλλά και η "έξυπνη" εξερεύνηση έχουν βέλτιστα σημεία κάτω και αριστερά από την αρχική καμπύλη. Με μωβ χρώμα σημειώνεται το μόνο σημείο που χάνει η "έξυπνη" εξερεύνηση σε αντίθεση με την εξαντλητική. Η σύγκριση χρόνου εκτέλεσης όμως την κάνει πολύ πιο αποδοτική.

Σύγκριση χρόνου εκτέλεσης για την εξερεύνηση (για το παράδειγμα που εμφανίζεται στα σχήματα 3.5 και 3.17):

Όπως φαίνεται στον πίνακα 3.3 ο χρόνος εκτέλεσης της εξερεύνησης που υλοποιήθηκε και χρησιμοποιήθηκε είναι αρκετές τάξεις μικρότερος από το χρόνος μιας εξαντλητικής εξερεύνησης και με επιτυχή εύρεση των καλύτερων λύσεων σε ποσοστό 90%.



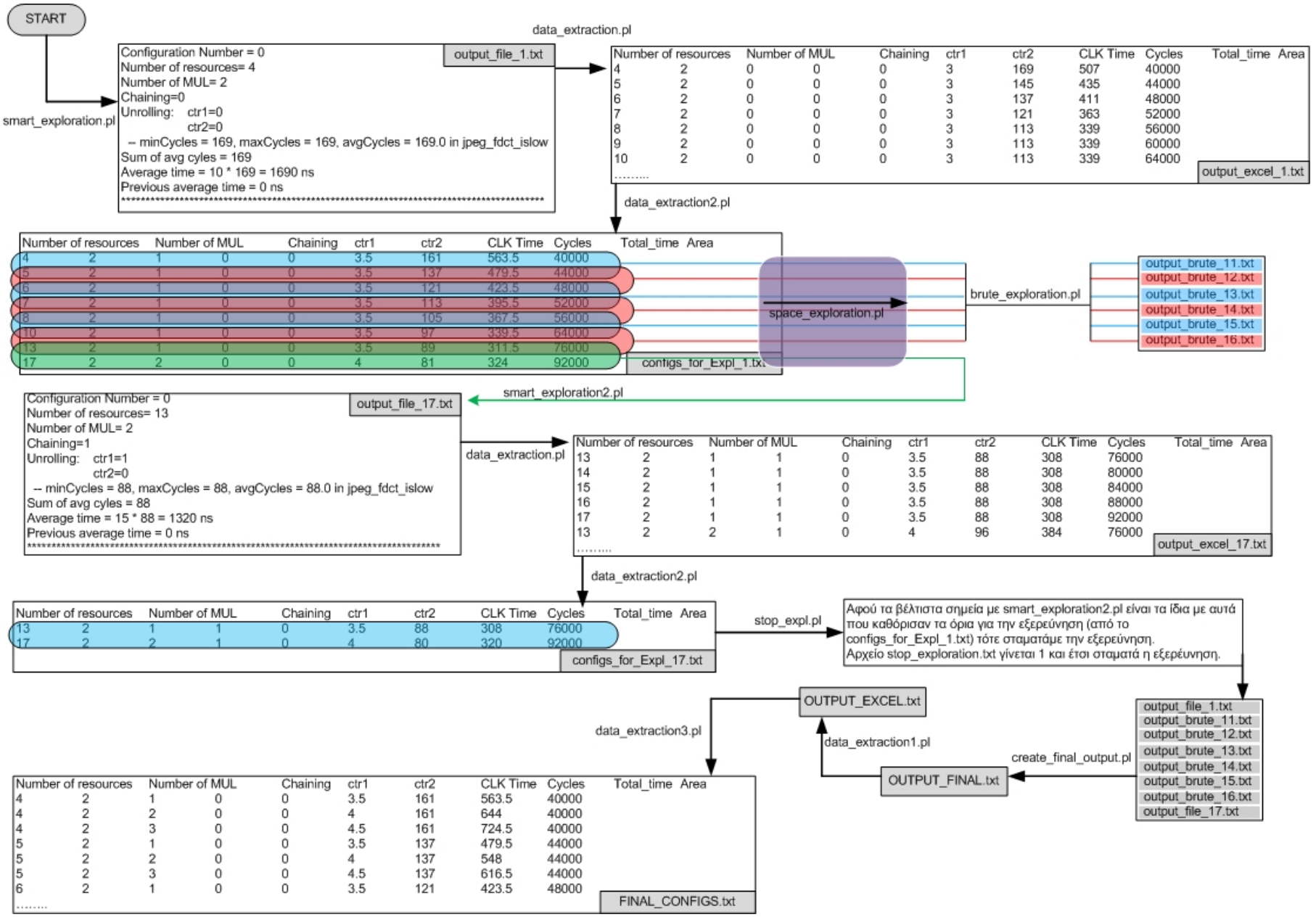
Σχήμα 3.17: Σύγκριση αρχικής καμπύλης λύσεων χωρίς chaining-unrolling με την καμπύλη λύσεων με "smart" και εξαντλητική εξερεύνηση.

Τύπος εξερεύνησης	Χρόνος εκτέλεσης	Ποσοστό εύρεσης (σε σχέση με την εξαντλητική εξερεύνηση)
Εξαντλητική	22 ώρες, 49 λεπτά και 2 δευτερόλεπτα	100%
"Έξυπνη"	0 ώρες, 9 λεπτά και 34 δευτερόλεπτα	90.9%

Πίνακας 3.3: Σύγκριση χρόνων εκτέλεσης της εξαντλητικής εκτέλεσης και της έξυπνης εξερεύνησης.

Παρακάτω (Σχήμα 3.18) παρουσιάζεται και ένα παράδειγμα για τα αρχεία δεδομένων που δημιουργούνται σε κάθε βήμα της εξερεύνησης, τη διαμόρφωση τους αλλά και την επιλογή για εξαντλητική ή "έξυπνη" εξερεύνηση στο κομμάτι για την εξερεύνηση χώρου. Στο αρχείο *configs_for_Expl_1.txt* με γαλάζιο και ελαφρύ κόκκινο χρώμα εναλλάξ, παρουσιάζονται τα σημεία στα διαστήματα των οποίων θα γίνει εξαντλητική εξερεύνηση. Με πράσινο χρώμα σημειώνεται το διάστημα μεταξύ δύο σημείων στο οποίο εκτελείται "έξυπνη" εξερεύνηση. Στο επόμενο αρχείο *configs_for_Expl_17.txt* υπάρχουν μόνο 2 βέλτιστες λύσεις που προέκυψαν από το διάστημα σημείων με πράσινο χρώμα. Αυτές συγκρίνονται με τα σημεία που καθόρισαν το διάστημα για την εξερεύνηση και αφού βρέθηκα οι ίδιες παράμετροι στις δύο λύσεις δεν γίνεται περαιτέρω εξερεύνηση στο διάστημα που δημιουργούν. Η σύγκριση γίνεται με τη χρήση του *stop_expl.pl*. Στη συνέχεια ενώνονται όλα τα αρχεία εξόδου που προέκυψαν από την εξερεύνηση *output_file.txt* (από "έξυπνη εξερεύνηση") και *output_brute.txt* (από εξαντλητική εξερεύνηση) στο αρχείο *OUTPUT_FINAL.txt*. Αυτό μετατρέπεται στη μορφή excel και με το *output_extraction3.pl* εξάγουμε τις τελικές βέλτιστες λύσεις στο αρχείο *FINAL_CONFIGS.txt*.

Ακολουθεί ο Πίνακας 3.4 με όλα τα εργαλεία-scripts που δημιουργήθηκαν για τη ροή της εξερεύνησης στο επίπεδο του HLS με μια περιγραφή για τη λειτουργία τους αλλά και τα αρχεία εισόδου που παίρνουν αλλά και τα αρχεία εξόδου που παράγουν.



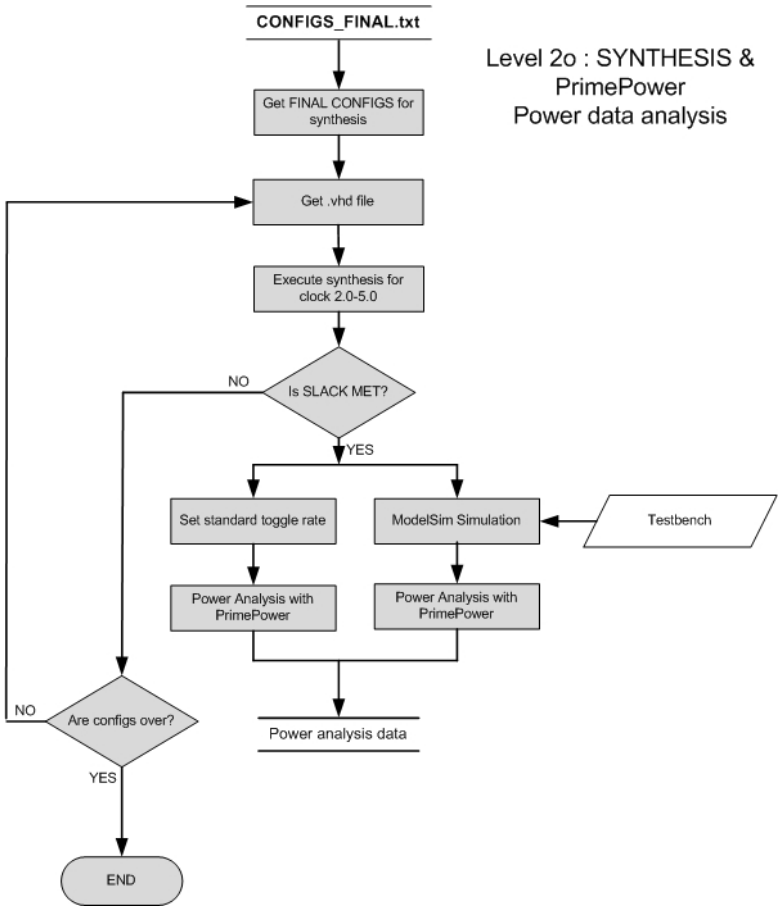
Σχήμα 3.18: Παράδειγμα για αρχεία δεδομένων που δημιουργούνται στην διάρκεια της εξερεύνηση.

Όνομα script	Περιγραφή λειτουργίας	Παράμετροι Εισόδου	Αρχεία Εισόδου	Αρχεία Εξόδου
smart_exploration_1	εκτελεί την πρώτη prunning εξερεύνηση με συνδυασμούς μόνο τον αριθμό ALUs, MULs και chaining (χωρίς unrolling)	ελάχιστος και μέγιστος αριθμός ALUs και MULS, ελάχιστο και μέγιστο βαθμό chaining		output_file.txt
data_extraction	διαμορφώνει το αρχείο εξόδου από το smart_exploration σε μορφή excel		output_file.txt	output_excel.txt
data_extraction_2	από το αρχείο output_excel.txt διαλέγει τα καλύτερα σημεία-συνδυασμούς με μικρότερο βαθμό chaining και unrolling (αν υπάρχει)		output_excel.txt	configs_for _Expl.txt
space_exploration	από το αρχείο configs_for_Expl.txt διαβάζουμε τα σημεία ανά δύο συνεχόμενα και ανάλογα με το πλήθος των συνδυασμών που προκύπτουν στο διάστημα τους εκτελείται prunning ή brute εξερεύνηση		configs_for_Expl.txt	output_file_no.txt output_brute.txt
smart_exploration_2	εκτελεί την prunning εξερεύνηση στο διάστημα που προκύπτει από 2 σημεία στο configs_for_Expl.txt με πλήθος συνδυασμών $>$ 100 με συνδυασμούς τον αριθμό ALUs, MULs, chaining και unrolling	ελάχιστος και μέγιστος αριθμός ALUs και MULS, ελάχιστο και μέγιστο βαθμό chaining		output_file.txt
brute_ exploration	εκτελεί την εξαντλητική εξερεύνηση στο διάστημα που προκύπτει από 2 σημεία στο configs_for _Expl.txt με πλήθος συνδυασμών $\leq$ 100 με συνδυασμούς τον αριθμό ALUs, MULs, chaining και unrolling	ελάχιστος και μέγιστος αριθμός ALUs και MULS, ελάχιστο και μέγιστο βαθμό chaining		output_ brute.txt
data_extraction_3	από το αρχείο EXCEL_FINAL.txt διαλέγει τα καλύτερα σημεία-συνδυασμούς (μπορεί να προκύπτουν αρκετά με ίδια απόδοση και επιφάνεια υλικού αλλά διαφορετικό chaining και unorlling). Όλα τα σημεία που προκύπτουν κατατάσσονται με αύξων αριθμό ALUs, MULs, chaining και μετά unrolling.		EXCEL_FINAL.txt	FINAL_CONFIGS.txt

Πίνακας 3.4: Περιγραφή για τα αρχεία εισόδου που χρησιμοποιούνται στο HLS exploration.

3.4 Επίπεδο 2ο: RTL code to Post Synthesis

Στο δεύτερο επίπεδο χρησιμοποιούνται τα αποτελέσματα για τις βέλτιστες λύσεις από το πρώτο επίπεδο, αυτό του HLS για να παραχθούν τα δίκτυα πυλών με ακριβείς μετρήσεις για την επιφάνεια υλικού και το ρολόι χρονισμού αλλά και για να παραχθούν δεδομένα κατανάλωσης ισχύος για το σχέδιο. Σκοπός της εξερεύνησης που υλοποιείται στο επίπεδο αυτό είναι για κάθε βέλτιστη λύση από HLS να ελεγχθεί για ποιες τιμές ρολογιού το δίκτυο πυλών είναι έγκυρο και σε κάθε περίπτωση ποια είναι η επιφάνεια υλικού για το συγκεκριμένο ρολόι και η κατανάλωση ισχύος. Τα δεδομένα για την επιφάνεια υλικού και κατανάλωση ισχύος θα είναι διαθέσιμα μετά για το επίπεδο της χωροθέτησης.
Στο σχήμα 3.19 παρουσιάζεται μια αφαιρετική ροή της 2ου επιπέδου εξερεύνησης.



Σχήμα 3.19: Αφαιρετικό σχεδιάγραμμα ροής για την εξερεύνηση στο επίπεδο της σύνθεσης.

Η εξερεύνηση σ' αυτό το επίπεδο αφορά την τιμή του ρολογιού χρονισμού του σχεδίου για κάθε συνδυασμό παραμέτρων από τα βέλτιστα σημεία που προέκυψαν από την HLS εξερεύνηση. Έτσι για κάθε δεδομένο σχέδιο (κώδικα RTL που παράγεται από το εργαλείο του SPARK) ελέγχεται με σύνθεση σε τιμές ρολογιού από 2 ns μέχρι και 5 ns, με αύξηση ανά 0,5ns. Για όσα δίκτυα πυλών (net list) είναι έγκυρα (παρουσιάζουν την ένδειξη MET στο αρχείο του *timing.rpt*) τα στοιχεία τους που αφορούν την επιφάνεια υλικού καταγράφονται.
Στη συνέχεια σ' αυτά τα "έγκυρα" δίκτυα πυλών γίνεται ανάλυση δεδομένων ισχύος με το εργαλείο PrimePower για εξομοίωση της κατανάλωσης ισχύος σε δύο φάσεις. Η πρώτη μία με standard toggle rate εξομοίωση και η δεύτερη γίνεται με τη χρήση testbench στο εργαλείο εξομοίωσης ModelSim Simulator.

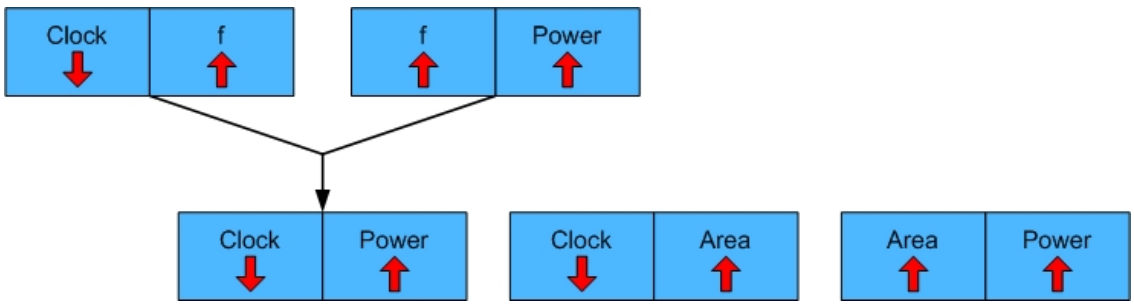
Σαν αποτέλεσμα, για κάθε συνδυασμό παραμέτρων (για κάθε βέλτιστο σημείο από την HLS εξερεύνηση) υπάρχουν αρκετές (μέχρι και 7) λύσεις, μία για κάθε τιμή ρολογιού για την οποία το δίκτυο πυλών του σχεδίου είναι έγκυρο. Κάθε μία έγκυρη λύση θα έχει μία τιμή για επιφάνεια υλικού (ακριβής υπολογισμός) και μία τιμή χρόνου ($latency = clocktime * cycles$). Επίσης για κάθε τέτοια λύση, το σύνολο των έγκυρων συνθέσιμων για κάθε συνδυασμών παραμέτρων θα εμφανίζεται σαν μια "διαταραχή" λύσεων.

Η αλλαγή τις τιμής του ρολογιού χρονισμού έχει άμεση επίδραση τόσο στην κατανάλωση ισχύος του σχεδίου αλλά και στην επιφάνεια υλικού που θα καταλαμβάνει. Ο τύπο για την κατανάλωση ισχύος δίνεται στον ακόλουθο τύπο (3.2).

$$P = \frac{1}{2} \cdot C \cdot f \cdot V_{dd}^2 \tag{3.2}$$

Η άμεση εξάρτηση τύπου έχει η τιμή του ρολογιού περιόδου με την κατανάλωση ισχύος φαίνεται στον παραπάνω τύπο (3.2). Η ισχύς είναι ανάλογη της συχνότητας f ($P =\propto f$). Και η συχνότητα είναι αντιστρόφως ανάλογη με το ρολόι χρονισμού, αφού $Clock = \frac{1}{f}$. Άρα το ρολόι με την ισχύ συνδέονται με τη σχέση $P =\propto \frac{1}{Clock}$ και αντιστρόφως $Clock =\propto \frac{1}{P}$. Έτσι με αύξηση της τιμής του ρολογιού δίνει και μικρότερες τιμές κατανάλωσης ισχύος, γι' αυτό και γίνεται μια εξερεύνηση σε κάθε λύση του HLS exploration για διάφορες τιμές ρολογιού.

Η σύνδεση της επιφάνειας υλικού με την τιμή του ρολογιού περιόδου είναι επίσης άμεση. Με τη μείωση της τιμής του ρολογιού, το κύκλωμα "πιέζεται" για να παράξει αποτέλεσμα σε μικρότερη χρονική περίοδο. Ως εκ τούτου, οι ενισχυτές (buffers) που τοποθετούνται στην έξοδο κάθε λειτουργικής μονάδας για να προωθούν το αποτέλεσμα, πρέπει να λειτουργούν πολύ γρήγορα. Αυτό έχει ως αποτέλεσμα να τοποθετούνται μεγάλοι ενισχυτές στο κύκλωμα, και τρανζίστορς για τη λειτουργία τους, από το εργαλείο σύνθεσης του δικτύου πυλών, και συνεπώς το κύκλωμα να έχει και μεγαλύτερη επιφάνεια υλικού. Η σχέση φαίνεται και Σχήμα 3.20 που δείχνει την εξάρτηση που έχουν μεταξύ τους οι παράγοντες.



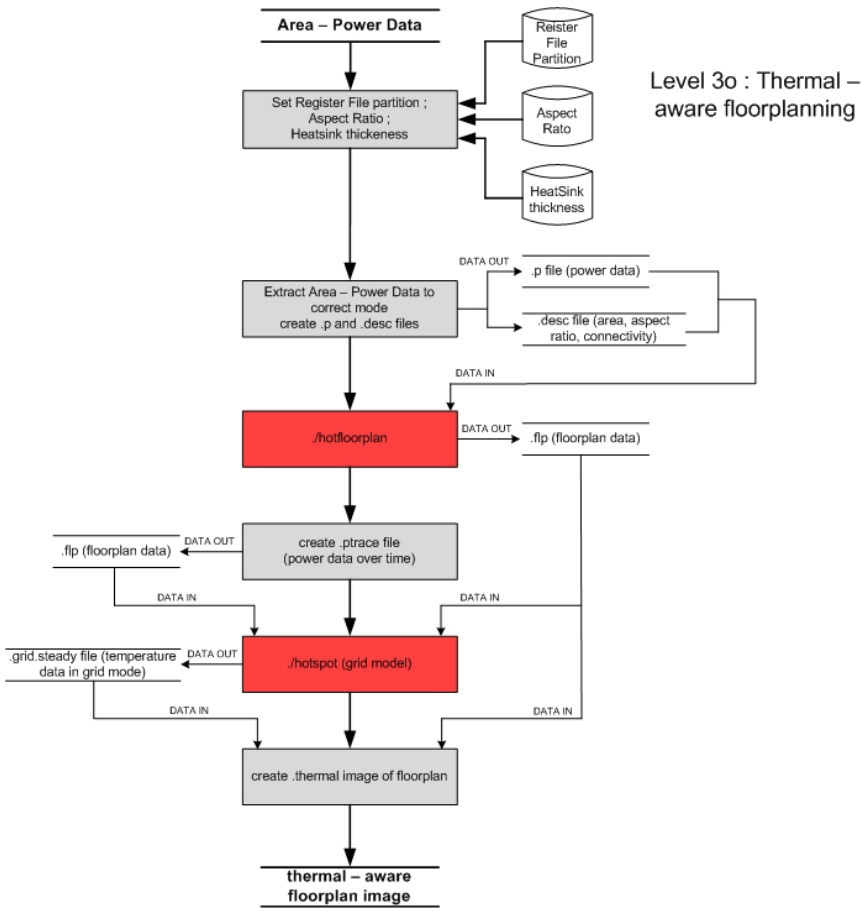
Σχήμα 3.20: Εξάρτηση των παραγόντων ρολόι περιόδου - ισχύς - επιφάνεια υλικού.

3.5 Επίπεδο 3ο: Power Data to Thermal Floorplan - Management

Το τρίτο και τελευταίο επίπεδο της εξερεύνησης αφορά τη χωροθέτηση του δικτύου πυλών, που παράγεται από το επίπεδο της σύνθεσης. Η χωροθέτηση γίνεται με βάση των δεδομένων επιφάνειας υλικού που παράγονται από τη σύνθεση του κυκλώματος αλλά και της κατανάλωση ισχύος που έχει το κύκλωμα, που δίνει τη δυνατότητα θερμικής διαχείρισης στη χωροθέτηση. Σαν αποτέλεσμα δίνεται μια εικόνα του κυκλώματος, η οποία παρουσιάζει τη χωροθέτηση των επιμέρους λειτουργικών μονάδων που το αποτελούν αλλά και θερμική κατάσταση στο κύκλωμα.

Σύμφωνα με τις αποφάσεις που υπάρχουν για αυτό το επίπεδο (βλέπε Σχήμα 3.4), οι διάφορες επιλογές των παραμέτρων επηρεάζουν και τη χωροθέτηση του σχεδίου αλλά και τη θερμική εικόνα που θα παρουσιάσει. Επιτρέποντας την ύπαρξη αρχείου καταχωρητών (Register File) στο κύκλωμα, και αλλάζοντας την κατάτμηση που αυτή θα έχει, δημιουργούνται διάφορα κομμάτια του Register File αντί για ένα, τα οποία μπορεί να τοποθετηθούν σε διαφορετικά μέρη του σχεδίου. Ακόμη αλλάζοντας το Aspect Ratio αλλάζει και η τοπολογία των κομματιών αλλάζοντας προφανώς και τη χωροθέτηση. Τέλος το πλάτος που θα έχει η ψήκτρα (heatsink) που τοποθετείται στο κύκλωμα αλλάζει και τη θερμική εικόνα του, μειώνοντας προφανώς τη θερμότητα του.

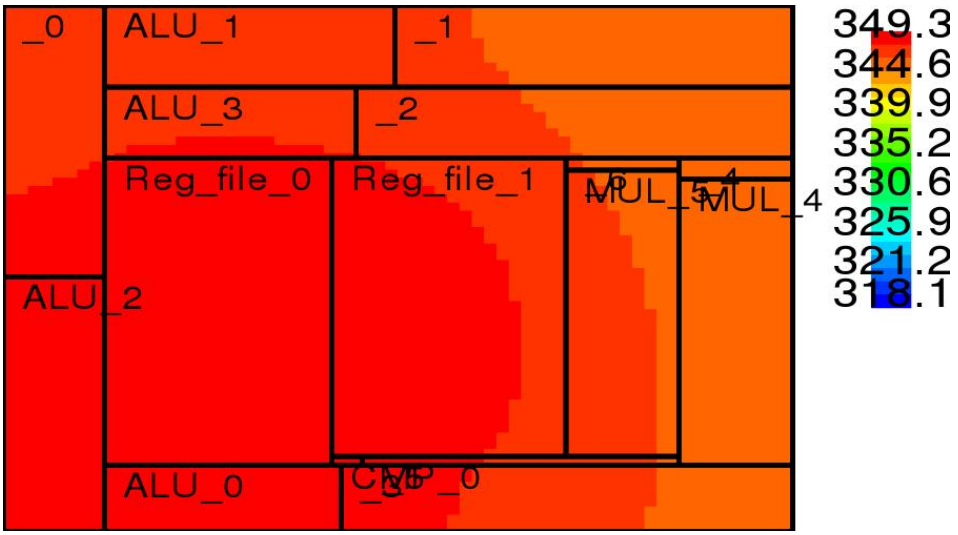
Στο σχήμα 3.21 παρουσιάζεται η ροή που ακολουθείται για το επίπεδο της Χωροθέτησης με δυνατότητα θερμικής διαχείρισης.



Σχήμα 3.21: Διάγραμμα ροής για το επίπεδο της Χωροθέτησης.

Η ροή στο επίπεδο χωροθέτησης ξεκινά με την είσοδο των δεδομένων για την επιφάνεια υλικού και την κατανάλωση ισχύος του κυκλώματος (δικτύου πυλών). Στη συνέχεια πρέπει να καθοριστούν οι παράμετροι (που παρουσιάζονται και στο Σχήμα 3.4) για την κατάτμηση του Register File, το Aspect Ratio που θα χρησιμοποιηθεί στη χωροθέτηση και το πάχος της ψήκτρας που θα χρησιμοποιηθεί (HeatSink thickness). Αυτό γίνεται έτσι ώστε στο επόμενο στάδιο, τα δεδομένα για την επιφάνεια υλικού και κατανάλωση ισχύς να μετατραπούν στη σωστή μορφή για τη χρήση τους στο εργαλείο του Hotfloorplan και να δημιουργηθούν τα αρχεία εισόδου (*.p* με τα δεδομένα κατανάλωσης ισχύος και *.desc* με τα δεδομένα επιφάνειας υλικού, επιτρεπόμενο aspect ratio και την καλωδίωση του κυκλώματος). Αφού παραχθούν τα κατάλληλα αρχεία εκτελείτε το εργαλείο του Hotfloorplan και παράγεται το αρχείο *.flp* με τη διάταξη του σχεδίου. Αυτό το αρχείο, μαζί με το αρχείο *.ptrace* που δημιουργείται με το script *create_ptrace_file* και το οποίο αποτελεί το power trace file, δίνονται για είσοδο στο εργαλείο HotSpot για να γίνει θερμική ανάλυση του κυκλώματος. Αυτή η θερμική ανάλυση διαχωρίζει το κύκλωμα σε μορφή πλέγματος με ανάλυση που καθορίζεται στο *hotspot.config* (default 64·64) και υπολογίζει την θερμοκρασία που υπάρχει σε κάθε "στοιχείο" του πλέγματος. Από αυτό τα στάδιο παράγεται το αρχείο *.grid.steady* που περιέχει τη θερμοκρασία για κάθε στοιχείο του πλέγματος. Για να δημιουργηθεί η θερμική εικόνα του κυκλώματος στο δεδομένο floorplan, τα αρχεία *.flp* και *.grid.steady* δίνονται σαν είσοδος στο script *grid_thermal_map2.pl*. Έξοδος είναι η θερμική εικόνα του σχεδίου.

Παράδειγμα θερμικής εικόνας σχεδίου φαίνεται στο Σχήμα 3.22, που αντιστοιχεί στο kernel dct του jpeg. Μέγιστη θερμοκρασία του κυκλώματος είναι οι 76.17 °C (349.32 Kelvin), ενώ η μέση θερμοκρασία που επικρατεί στο υπόλοιπο μέρος είναι $\simeq$ 68 °C. Φαίνεται πως η θερμότητα είναι κατανεμημένη σε όλο το κύκλωμα και η μέγιστη θερμοκρασία δεν απέχει πολύ από την ελάχιστη. Ακόμη, στο ακόλουθο σχήμα φαίνεται και το αρχείο των καταχωρητών (Register File) το οποίο είναι χωρισμένο σε δύο μέρη. Τη μέγιστη θερμοκρασία παρουσιάζουν το ένα μέρος του αρχείου καταχωρητών (Reg_file_0) και δύο μονάδες ALU. Αυτό είναι αναμενόμενο αφού οι περισσότερες λειτουργίες αφορούν πρόσθεση, αφαίρεση και ολίσθηση, όπως και πρόσβαση σε καταχωρητές. Στο σχέδιο υπάρχει επίσης και μία μονάδα σύγκρισης. Τέλος παρατηρούνται και ορισμένα κομμάτια που δεν αντιπροσωπεύουν κάποια συγκεκριμένη μονάδα λειτουργίας αλλά τοποθετούνται από το εργαλείο του HotSpot για να δημιουργηθεί το ορθογώνιο σχήμα του κυκλώματος και για να υλοποιηθούν οι συνδέσεις μεταξύ των μονάδων (wiring) του κυκλώματος.

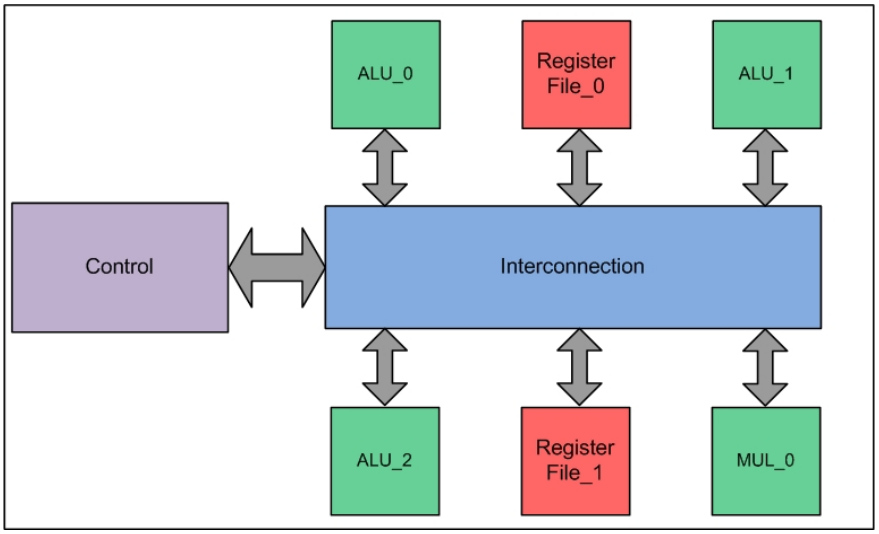


Σχήμα 3.22: Θερμική εικόνα χωροθετημένου σχεδίου για τον πυρήνα DCT(loop1) με αριθμό λειτουργικών μονάδων: 4 ALUs, 2 MULs, 1 CMP, 2 Register Files.

3.6 Ενδεικτική μοντελοποίηση αρχιτεκτονικών Coarse Grained Reconfigurable (CGRArch) σε επίπεδο C

Η μοντελοποίηση αρχιτεκτονικών CGRArch γίνεται με δομές (blocks) *if-then-else* και τη χρήση μιας γενικής μεταβλητής (global variable) για την "πλοήγηση" ανάμεσα στους πυρήνες που χωρίζονται με τα *if-then-else* blocks.

Το πρότυπο αρχιτεκτονικής για τη μεθοδολογία παρουσιάζεται στο Σχήμα 3.23. Το κλειδί αυτού είναι ότι πρόκειται να χρησιμοποιήσει τα coarse grained blocks, δηλαδή τις λειτουργικές μονάδες και τα register files σε ένα reconfigurable datapath που κατασκευάζεται χρησιμοποιώντας ένα ανά πυρήνα διαμορφούμενο δίκτυο διασύνδεσης.

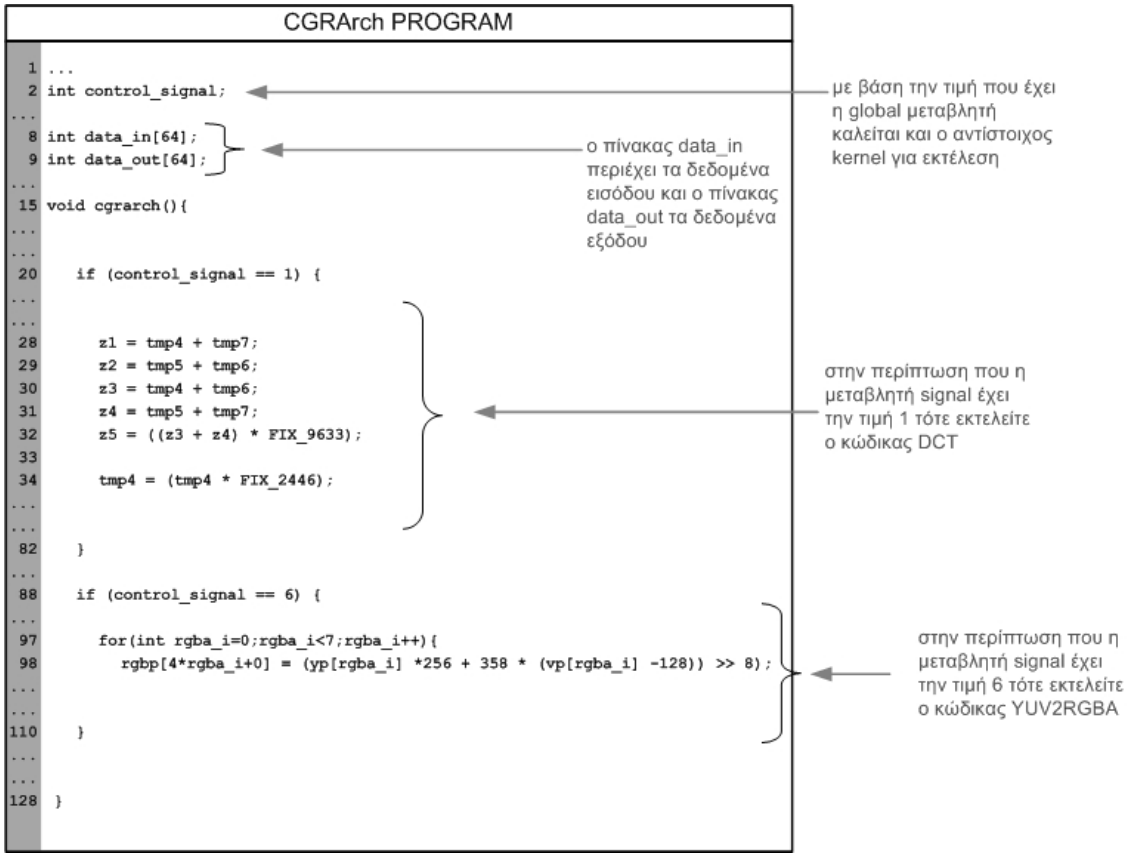


Σχήμα 3.23: Μοντέλο της αρχιτεκτονικής για CGRArch.

Ανάλογα με τον πυρήνα C (κώδικα) που επιλέγεται να εκτελεστεί, η διασύνδεση (interconnection) της αρχιτεκτονικής CGRArch τροποποιείται, και οι διαφορετικές λειτουργίες (operations) του κάθε πυρήνα απεικονίζονται (mapped) στις λειτουργικές μονάδες. Ο κύριος στόχος στις αρχιτεκτονικές CGRArch είναι να ελαχιστοποιηθούν οι διασυνδέσεις μεταξύ των FUs για κάθε διαφορετικό πυρήνα που εκτελείται, αλλά και η μέγιστη χρησιμοποίηση των διαθέσιμων μονάδων ώστε να έχουμε μεγαλύτερη απόδοση.

Ένα παράδειγμα για την διαφορετική απεικόνιση λειτουργιών στις μονάδες του CGRArch φαίνεται στο Σχήμα 3.25. Η απεικόνιση των εντολών που φαίνονται στο Σχήμα **??**, απεικονίζονται στις διαθέσιμες λειτουργικές μονάδες και σε κάθε control step με περιορισμούς τις διάφορες εξαρτήσεις (Read-after-Write, Write-after-Read και Write-after-Write).

Φαίνεται στον πίνακα απεικόνισης (Σχήμα 3.25) ότι ανάλογα με το πυρήνα που καλείται να εκτελεστεί στην αρχιτεκτονική CGRArch, κάτι το οποίο καθορίζεται από μία γενική (global) μεταβλητή στο κύριο πρόγραμμα, διαφορετικές λειτουργίες χρονοδρομολογούνται στις διάφορες λειτουργικές μονάδες, σε κάθε χρονικό βήμα. Ακόμη διαφορετικό είναι το δίκτυο διασύνδεσης που υλοποιείται στην αρχιτεκτονική για να βελτιστοποιηθεί η απόδοση.



Σχήμα 3.24: Μοντελοποίηση της αρχιτεκτονικής για CGRArch για διάφορους πυρήνες σε γλώσσα C.

		Functional Units				Register Files	
KERNEL	Control Steps	ALU_0	ALU_1	ALU_2	MUL_1	RegFile_0	RegFile_1
DCT	1	z1 = tmp4 + tmp7;	z2 = tmp5 + tmp6;	z3 = tmp4 + tmp6;	---	---	---
	2	z4 = tmp5 + tmp7;	---	---	---	---	---
	3	REG_1 = z3 + z4	---	---	---	REG_0 = FIX_9633	---
	4	---	---	---	z5 = TEMP * REG_0	REG_1 = FIX_2446	---
	5	---	---	---	tmp4 = (tmp4 * REG_1);	---	---
YUV2RGBA	1	---	---	---	---	REG_0 = yp[rgba_i]	REG_1 = up[rgba_i]
	2	REG_3 = REG_0 << 8	REG_4 = REG_2 – 128	---	---	REG_2 = vp[rgba_i]	---
	3	---	---	---	REG_5= REG_4 * 358	---	---
	4	REG_7 = REG_3 + REG_	REG_6 = rgba_i <<2	---		---	---
	5	REG_6 = REG_6 + 0	REG_3 = REG_0 << 8	REG_4 = REG_2 – 128	REG_5 = REG_1 – 128	---	---
	6				REG_4 = REG_4 * 88	rgbp[REG_6] = REG_7	

Σχήμα 3.25: Απεικόνιση των λειτουργιών από τους πυρήνες DCT και YUV2RGBA στις λειτουργικές μονάδες του CGRArch μοντέλου.

Κεφάλαιο 4

Πειραματικά Αποτελέσματα

4.1 Επίπεδο 1ο: HLS C code to RTL code

4.1.1 Αποτελέσματά πειραματικών μετρήσεων

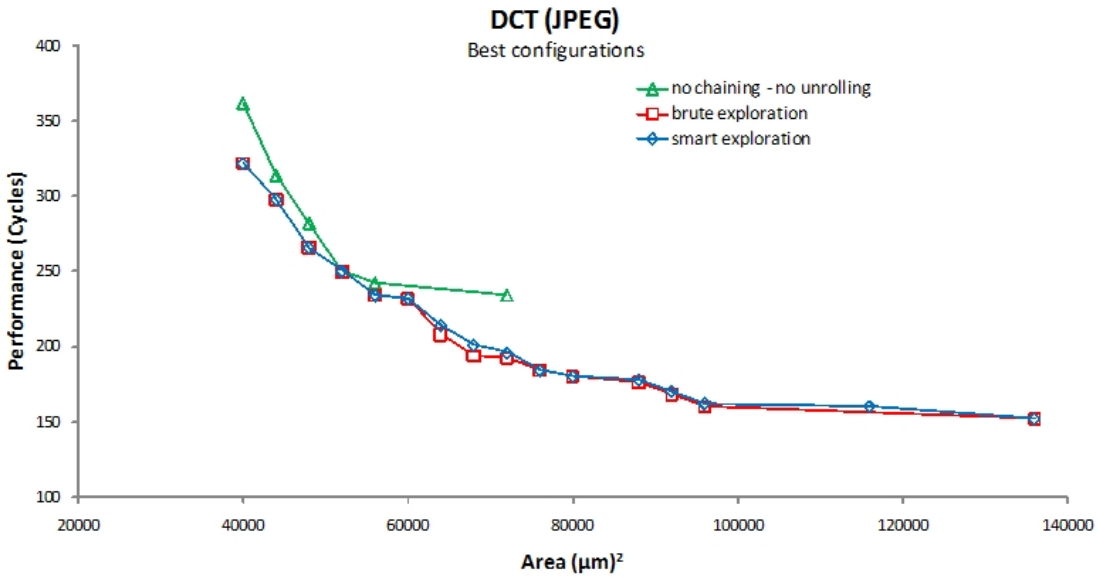
Για να υπάρξει μία πιο ακριβής και τεκμηριωμένη εικόνα για τη λειτουργία και την απόδοση του framework που δημιουργήθηκε για το επίπεδο από κώδικα C σε κώδικα RTL, χρησιμοποιήθηκαν αρκετά διαφορετικά kernels (πυρήνες). Για λόγους σύγκρισης έγινε και εκτέλεση εξαντλητικής εξερεύνησης για κάθε kernel. Τα αποτελέσματα παρουσιάζονται στον Πίνακα 4.1. Το ποσοστό επιτυχίας (Accuracy) είναι το ποσοστό εύρεσης βέλτιστων λύσεων-συνδιασμών με τη "έξυπνη" υλοποίηση σε σχέση με τα βέλτιστες λύσεις-συνδιασμούς που προκύπτουν από την εξαντλητική εξερεύνηση.

Όνομα kernel	Brute exploration time	Smart exploration time	Speedup	Accuracy
dct_jpeg.c	36 hours, 10 minutes and 53 seconds (130253 seconds)	4 hours, 2 minutes and 2 seconds (14522 seconds)	$\simeq$ x 9	100%
idct_mpeg.c	120 hours, 0 minutes and 40 seconds (432040 seconds)	0 hours, 14 minutes and 26 seconds (866 seconds)	$\simeq$ x 499	80% (4/5)
plus_mpeg.c	245 hours, 42 minutes and 8 seconds (884528 seconds)	17 hours, 21 minutes and 52 seconds (62512 seconds)	$\simeq$ x 14	100% (3/3)
fft.c	0 hours, 1 minutes and 18 seconds (78 seconds)	0 hours, 0 minutes and 14 seconds (14 seconds)	$\simeq$ x 6	100%
dwt.c	27 hours, 54 minutes and 8 seconds (100448 seconds)	1 hours, 0 minutes and 56 seconds (3656 seconds)	$\simeq$ x 27.5	75% (6/8)
yuv2rgba.c	0 hours, 59 minutes and 59 seconds (3599 seconds)	0 hours, 12 minutes and 29 seconds (749 seconds)	$\simeq$ x 5	92% (11/12)
matmul_mesa.c	0 hours, 18 minutes and 9 seconds (1089 seconds)	0 hours, 2 minutes and 12 seconds (132 seconds)	$\simeq$ x 8	60% (7/12)

Πίνακας 4.1: Αποτελέσματα από πειραματικές μετρήσεις για διάφορα kernels.

Στη συνέχεια παρουσιάζονται τα αποτελέσματα για κάθε πυρήνα (kernel) ξεχωριστά σε γραφήματα.

DCT (JPEG)

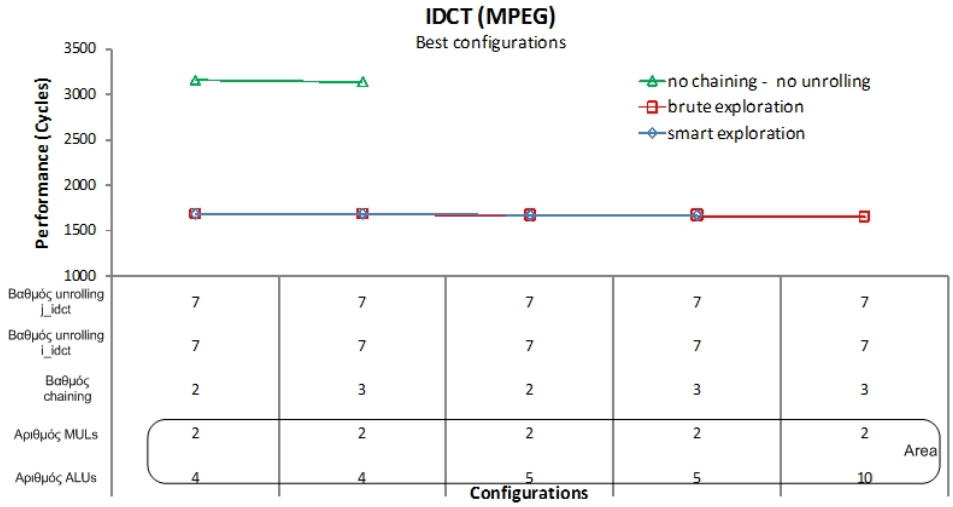


Σχήμα 4.1: Γράφημα για την εκτέλεση του πυρήνα του dct (jpeg).

Στο Σχήμα 4.1 φαίνονται οι βέλτιστες λύσεις από την "έξυπνη" εκτέλεση και την εξαντλητική εξερεύνηση. Ακόμα με πράσινο χρώμα φαίνονται οι βέλτιστες λύσεις που θα προκύψουν αν στην εξερεύνηση δεν ληφθούν υπόψη οι παράγοντες του chaining και του unrolling. Οι λύσεις αυτές είναι λιγότερες από τις περιπτώσεις της "έξυπνης" και της εξαντλητικής εξερεύνησης γιατί δεν παρουσιάστηκαν καλύτερες για μεγαλύτερη επιφάνεια υλικού. Φαίνεται ξεκάθαρα ότι αν οι παράγοντες chaining και unrolling δεν αποτελούν μέρος της εξερεύνησης χάνονται σχεδόν όλες οι βέλτιστες λύσεις. Σημειώνεται ότι οι μικρές διάφορες στην απόδοση σε ίδια επιφάνεια υλικού μεταξύ της "έξυπνης" και της εξαντλητικής εξερεύνησης οφείλεται σε διάφορους εξωγενής παράγοντες (όπως τη χρήση του μηχανήματος (server) στον οποίο έγινε η προσομοίωση).

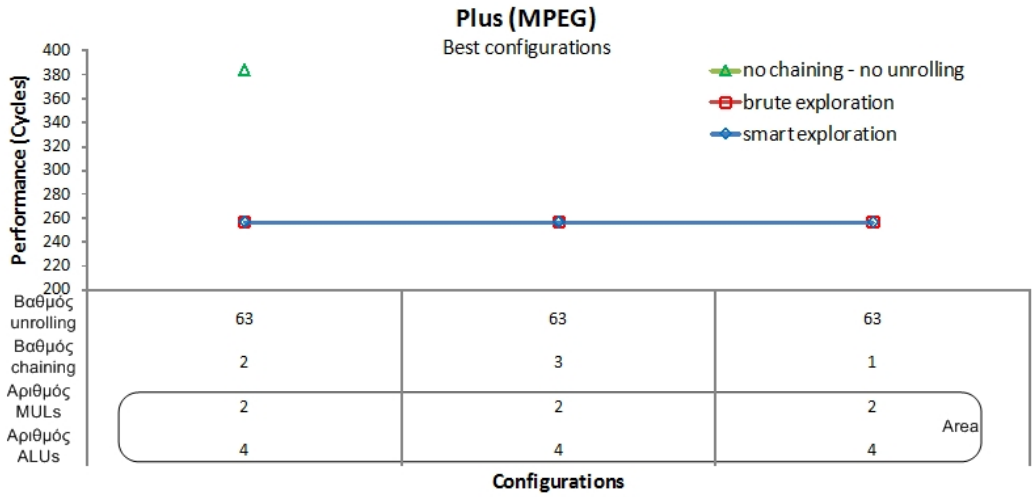
IDCT (MPEG)

Στο Σχήμα 4.2 φαίνονται οι βέλτιστες λύσεις από την "έξυπνη" εκτέλεση και την εξαντλητική εξερεύνηση. Στον οριζόντιο άξονα υπάρχουν οι συνδυασμοί των παραμέτρων (αυτό έγινε αφού οι βέλτιστες λύσεις ήταν λίγες σε πλήθος και υπήρχε η δυνατότητα να παρουσιαστούν). Ακόμα με πράσινο χρώμα φαίνεται η βέλτιστη λύση που θα προκύψει αν στην εξερεύνηση δεν ληφθούν υπόψη οι παράγοντες του chaining και του unrolling. Οι λύσεις αυτές είναι λιγότερες και χειρότερες από τις περιπτώσεις της "έξυπνης" και της εξαντλητικής εξερεύνησης γιατί δεν παρουσιάστηκαν καλύτερες για μεγαλύτερη επιφάνεια υλικού. Φαίνεται ξεκάθαρα ότι αν οι παράγοντες chaining και unrolling δεν αποτελούν μέρος της εξερεύνησης χάνονται σχεδόν όλες οι βέλτιστες λύσεις. Σημειώνεται ότι με τη χρήση της έξυπνης μεθόδου χάνεται μία λύση (σε σχέση με τη εξαντλητική εξερεύνηση), λόγω του ότι ο συγκεκριμένος πυρήνας παρουσιάζει αρχική ευθεία με βάθος μεγαλύτερο του 5, και η "έξυπνη" εξερεύνηση δε ξεφεύγει από αυτήν.



Σχήμα 4.2: Γράφημα για την εκτέλεση του πυρήνα του idct (mpeg).

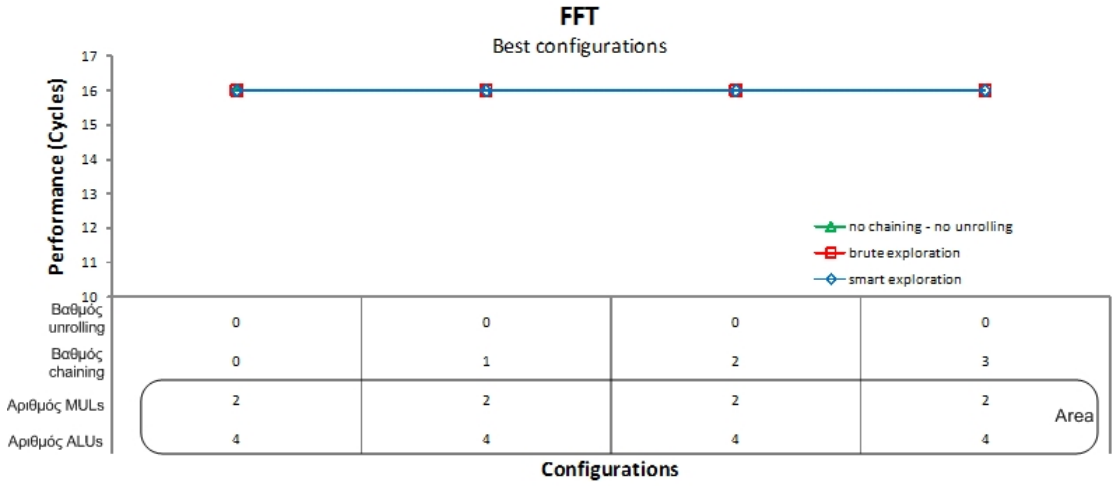
PLUS (MPEG)



Σχήμα 4.3: Γράφημα για την εκτέλεση του πυρήνα του plus (mpeg).

Οι βέλτιστες λύσεις για τον πυρήνα *plus (mpeg)* παρουσιάζονται στο Σχήμα 4.2. Στον οριζόντιο άξονα υπάρχουν οι συνδυασμοί των παραμέτρων (αυτό έγινε αφού οι βέλτιστες λύσεις ήταν λίγες σε πλήθος και υπήρχε η δυνατότητα να παρουσιαστούν). Ακόμα με πράσινο χρώμα φαίνονται οι βέλτιστες λύσεις που θα προκύψουν αν στην εξερεύνηση δεν ληφθούν υπόψη οι παράγοντες του chaining και του unrolling. Οι λύσεις αυτές είναι λιγότερες και χειρότερες από τις περιπτώσεις της "έξυπνης" και της εξαντλητικής εξερεύνησης γιατί δεν παρουσιάστηκαν καλύτερες για μεγαλύτερη επιφάνεια υλικού. Φαίνεται ξεκάθαρα ότι αν οι παράγοντες chaining και unrolling δεν αποτελούν μέρος της εξερεύνησης χάνονται σχεδόν όλες οι βέλτιστες λύσεις. Σημειώνεται ότι με τη χρήση της έξυπνης μεθόδου χάνεται μία λύση (σε σχέση με τη εξαντλητική εξερεύνηση), λόγω του ότι ο συγκεκριμένος πυρήνας παρουσιάζει αρχική ευθεία με βάθος μεγαλύτερο του 5, και η "έξυπνη" εξερεύνηση δε ξεφεύγει από αυτήν.

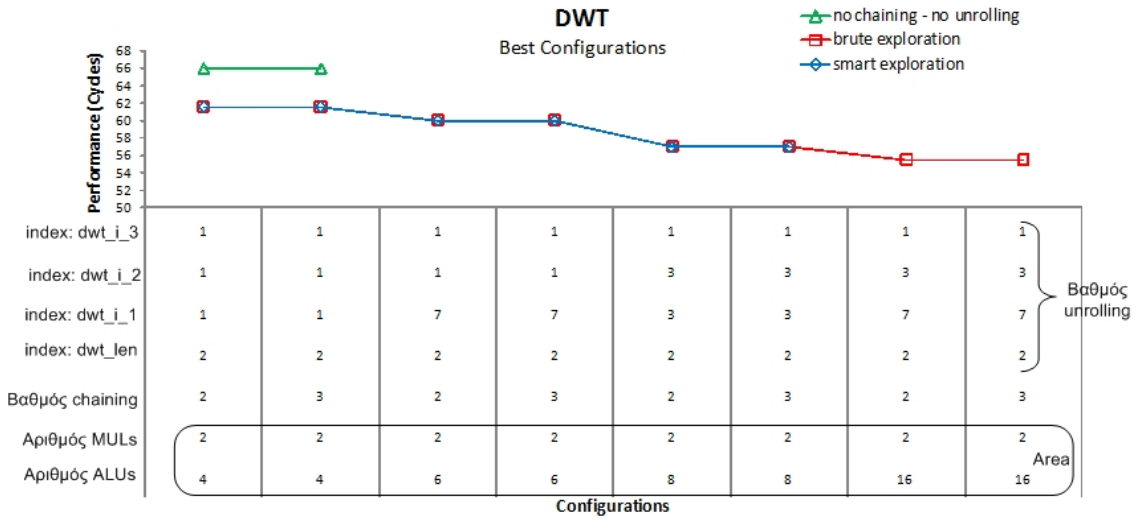
FFT



Σχήμα 4.4: Γράφημα για την εκτέλεση του πυρήνα του fft.

Στο Σχήμα 4.4 παρουσιάζονται οι βέλτιστες λύσεις για τον πυρήνα (kernel) *fft*. Και σε αυτό το γράφημα στον οριζόντιο άξονα υπάρχουν οι συνδυασμοί των παραμέτρων. Στη περίπτωση που γίνει εξερεύνηση χωρίς να ληφθούν υπόψη οι παράγοντες του chaining και unrolling, ως βέλτιστη λύση παρουσιάζεται μόνο μία. Οι υπόλοιπες λύσεις αφορούν την εξερεύνηση με chaining και unrolling. Φαίνεται πως στο συγκεκριμένο πυρήνα υπάρχει 100% επιτυχία στην εύρεση βέλτιστων λύσεων με την "έξυπνη" εξερεύνηση σε σχέση με την εξαντλητική.

DWT

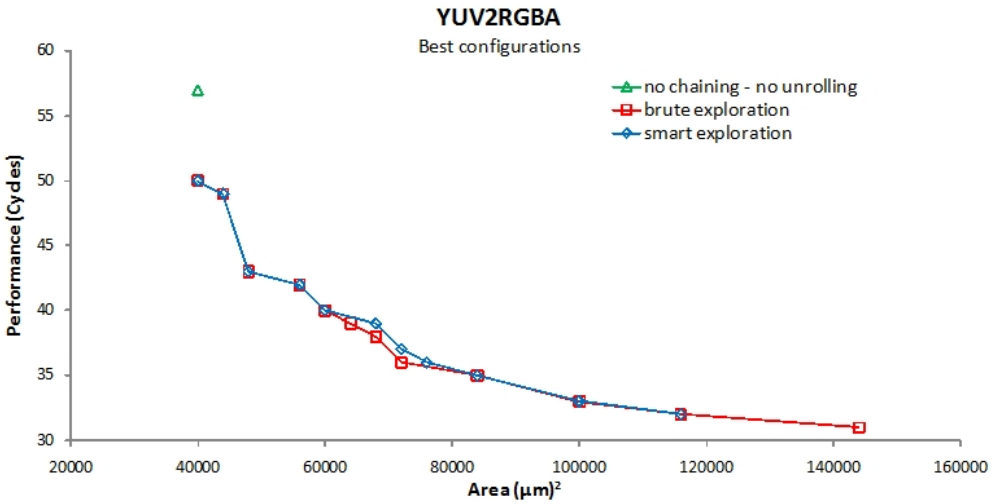


Σχήμα 4.5: Γράφημα για την εκτέλεση του πυρήνα του dwt.

Οι βέλτιστες λύσεις από την "έξυπνη" εκτέλεση και την εξαντλητική εξερεύνηση παρουσιάζονται στο Σχήμα 4.5, και με πράσινο χρώμα φαίνονται οι βέλτιστες λύσεις που θα προκύψουν αν στην

εξερεύνηση δεν ληφθούν υπόψη οι παράγοντες του chaining και του unrolling. Οι λύσεις αυτές δίνουν χειρότερη απόδοση (κύκλους) από τις λύσεις που σημειώνονται με μπλε και κόκκινο χρώμα. Φαίνεται ξεκάθαρα ότι αν οι παράγοντες chaining και unrolling δεν αποτελούν μέρος της εξερεύνησης χάνονται σχεδόν όλες οι βέλτιστες λύσεις.

YUV2RGBA

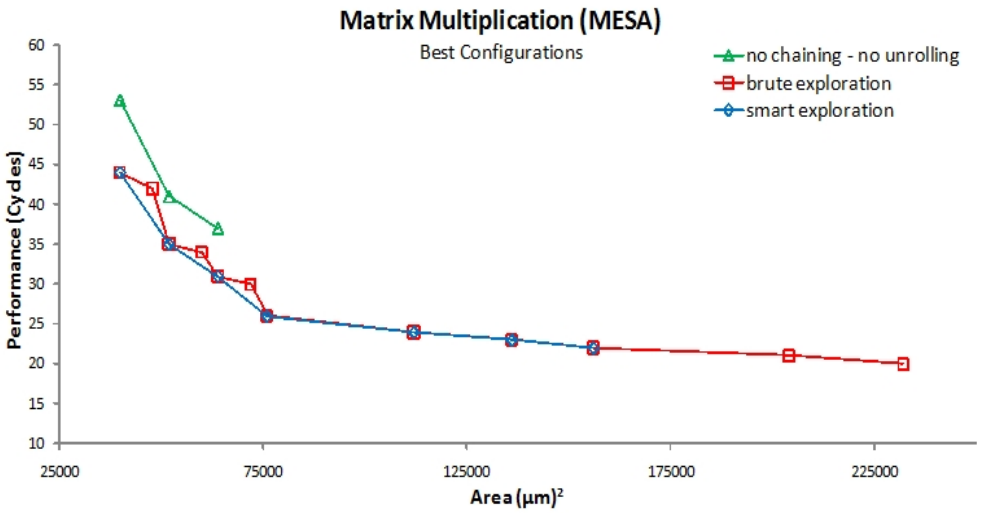


Σχήμα 4.6: Γράφημα για την εκτέλεση του πυρήνα του yuv2rgba.

Στο Σχήμα 4.6 παρουσιάζονται οι βέλτιστες λύσεις για τον πυρήνα του yuv2rgba_u8 από την "έξυπνη" εκτέλεση και την εξαντλητική εξερεύνηση. Με πράσινο χρώμα φαίνεται η μία λύση που θα προκύψει αν στην εξερεύνηση δεν ληφθούν υπόψη οι παράγοντες του chaining και του unrolling. Η μία μόνο λύση στην περίπτωση αυτή δείχνει και τη μεγάλη διάφορα με την εφαρμογή chaining και unrolling. Φαίνεται ξεκάθαρα ότι αν οι παράγοντες chaining και unrolling δεν αποτελούν μέρος της εξερεύνησης χάνονται όλες οι βέλτιστες λύσεις. Σημειώνεται ότι οι μικρές διαφορές στην απόδοση σε ίδια επιφάνεια υλικού μεταξύ της "έξυπνης" και της εξαντλητικής εξερεύνησης (η διακύμανση μεταξύ των μπλε σημείων και των κόκκινων) οφείλεται σε διάφορους εξωγενής παράγοντες (όπως τη χρήση του μηχανήματος (server) στον οποίο έγινε η προσομοίωση).

MATRIX MULTIPLICATION (MESA)

Στο Σχήμα 4.7 παρουσιάζονται οι βέλτιστες λύσεις για τον πυρήνα πολλαπλασιασμού πινάκων (matrix multiplication) του πακέτου MESA ("έξυπνη" εκτέλεση και εξαντλητική εξερεύνηση). Με πράσινο χρώμα φαίνονται οι λύσεις που θα προκύψουν αν στην εξερεύνηση δεν ληφθούν υπόψη οι παράγοντες του chaining και του unrolling. Φαίνεται ξεκάθαρα ότι αν οι παράγοντες chaining και unrolling δεν αποτελούν μέρος της εξερεύνησης χάνονται σχεδόν όλες οι βέλτιστες λύσεις. Παρατηρείται ότι τρία ενδιάμεσα κόκκινα σημεία, λύσεις της εξαντλητικής εξερεύνησης δε σημειώθηκαν από την "έξυπνη" εξερεύνηση.



Σχήμα 4.7: Γράφημα για την εκτέλεση του πυρήνα πολλαπλασιασμού πινάκων (matrix multiplication) του πακέτου MESA.

Γενικές Παρατηρήσεις

Ακολουθούν μερικές γενικές παρατηρήσεις που αφορούν τα παραπάνω γραφήματα για τους διάφορες πυρήνες.

1. Σε μερικούς πυρήνες παρατηρείται πως η "έξυπνη" εξερεύνηση χάνει ορισμένες λύσεις σε σχέση με τη εξαντλητική. Συνήθως αυτές οι λύσεις έχουν μεγάλη επιφάνεια υλικού (τα τελευταία σημεία στη γραφική παράσταση). Αυτό συμβαίνει λόγω του βάθους εξερεύνησης που υλοποιείται (κάτι που εξηγείται στην ενότητα 3.3.2), το οποίο βάθος είναι 5. Στους συγκεκριμένους πυρήνες, η "έξυπνη" εξερεύνηση αντιλαμβάνεται ευθεία με βάθος 5 χωρίς βελτίωση της απόδοσης και έτσι δεν ψάχνει περαιτέρω. Έτσι τυχόν λύσεις που βρίσκονται μετά από μεγάλες ευθείες χωρίς βελτίωση της απόδοσης, δεν εξετάζονται από την "έξυπνη" εξερεύνηση.

2. Ορισμένες λύσεις (όπως για παράδειγμα στον πυρήνα YUV2RGBA) ενώ ήταν αναμενόμενο να εντοπιστούν από την "έξυπνη" εξερεύνηση δεν βρέθηκαν. Αυτό οφείλεται στο γεγονός ότι οι λύσεις αυτές έχουν σαν παράμετρο βαθμό chaining x, αλλά στο διάστημα (interval space) που αυτές υπάρχουν, η "έξυπνη" εκτελέστηκε με όρια για το βαθμό chaining x+1 μέχρι το μέγιστο δυνατό βαθμό. Έτσι λύσεις που υπάρχουν στο διάστημα αυτό, αλλά με μικρότερο βαθμό chaining, δεν εντοπίζονται από την "έξυπνη" εξερεύνηση.

3. Σε αρκετά γραφήματα παρατηρείται απόκλιση στα σημεία-λύσεις που σημειώνονται από την εξαντλητική και την "έξυπνη" εξερεύνηση· η απόκλιση αφορά την απόδοση (κύκλους) στην ίδια επιφάνεια υλικού. Αυτό ίσως οφείλεται σε καθυστέρηση που δημιουργείται από το μηχάνημα (server) στο οποίο γίνεται η εκτέλεση των δύο περιπτώσεων εξερεύνησης. Αφού η εξερεύνηση γίνεται σε διαφορετικούς χρόνους (για να μην υπάρχει αλληλοκαθυστέρηση), το μηχάνημα πιθανώς να είχε διαφορετική χρήση, κάτι που επηρεάζει και τα αποτελέσματα.

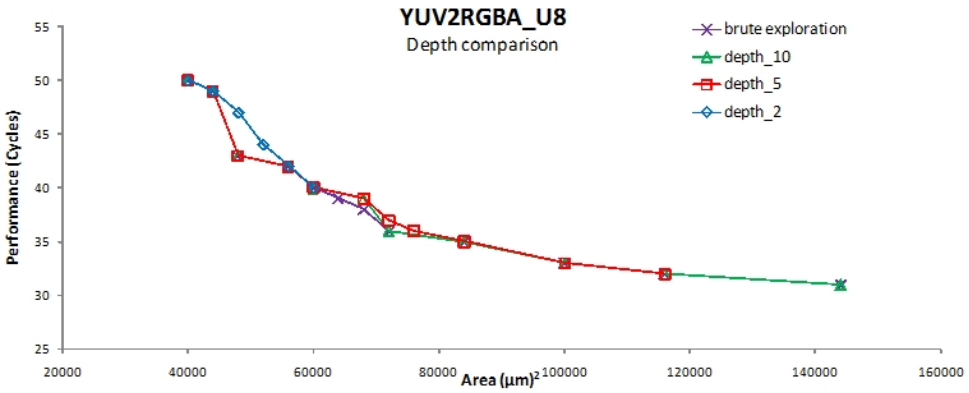
4.1.2 Έλεγχος βάθους αναζήτησης

Από τις πειραματικές μετρήσεις φάνηκε πως έχοντας μικρό "βάθος ελέγχου μηδενικής κλήσης" η εξερεύνηση ολοκληρωνόταν σε μικρό χρονικό διάστημα, χάνοντας όμως σημαντικό αριθμό βέλτιστων λύσεων και έτσι έδινε χαμηλό ποσοστό εύρεσης βέλτιστων λύσεων σε σύγκριση με τη εξαντλητική εξερεύνηση. Στην αντίθετη περίπτωση που χρησιμοποιείται μεγάλο "βάθος ελέγχου μηδενικής κλήσης" υπήρχε σχεδόν εύρεση όλων των βέλτιστων λύσεων αλλά ο χρόνος εκτέλεσης της εξερεύνησης ήταν πολύ μεγάλος, κάτι που δεν εξυπηρετεί ιδιαίτερα, αφού σκοπός είναι μια γρήγορη αλλά και αποδοτική εξερεύνηση. Έτσι καθορίστηκε το "βάθος ελέγχου μηδενικής κλήσης" στο 5 (έχοντας τη δυνατότητα αλλαγής για διάφορους κώδικες).

Παρακάτω παρατίθονται διάφορα παραδείγματα εκτέλεσης πυρήνων με διαφορετικά βάθη αναζήτησης (2,5 και 10), μαζί με την επιτάχυνση που επιτυγχάνεται και το ποσοστό επιτυχίας.

YUV2RGBA

Στο γράφημα 4.8 συγκρίνονται διαφορετικά επίπεδα βάθους αναζήτησης (βάθος 2, 5 και 10) σε ότι αφορά την αποδοτικότητα τους (accurasy) στην εύρεση βέλτιστων λύσεων σε σχέση με την εξαντλητική εξερεύνηση.



Σχήμα 4.8: Γράφημα διαφορετικού βάθους αναζήτησης λύσεων για το πυρήνα YUV2RGBA.

Τύπος εξερεύνησης	Βάθος εξερεύνησης	Χρόνος εκτέλεσης	Speedup	Accuracy
Εξαντλητική	—	0 hours, 59 minutes and 59 seconds (3599 seconds)	—	100%
"Έξυπνη"	2	0 hours, 3 minutes and 46 seconds (226 seconds)	$\simeq$ x 16	33.3% (4/11)
"Έξυπνη"	5	0 hours, 12 minutes and 29 seconds (749 seconds)	$\simeq$ x 5	92% (11/12)
"Έξυπνη"	10	0 hours, 20 minutes and 16 seconds (1216 seconds)	$\simeq$ x 3	100% (12/12)

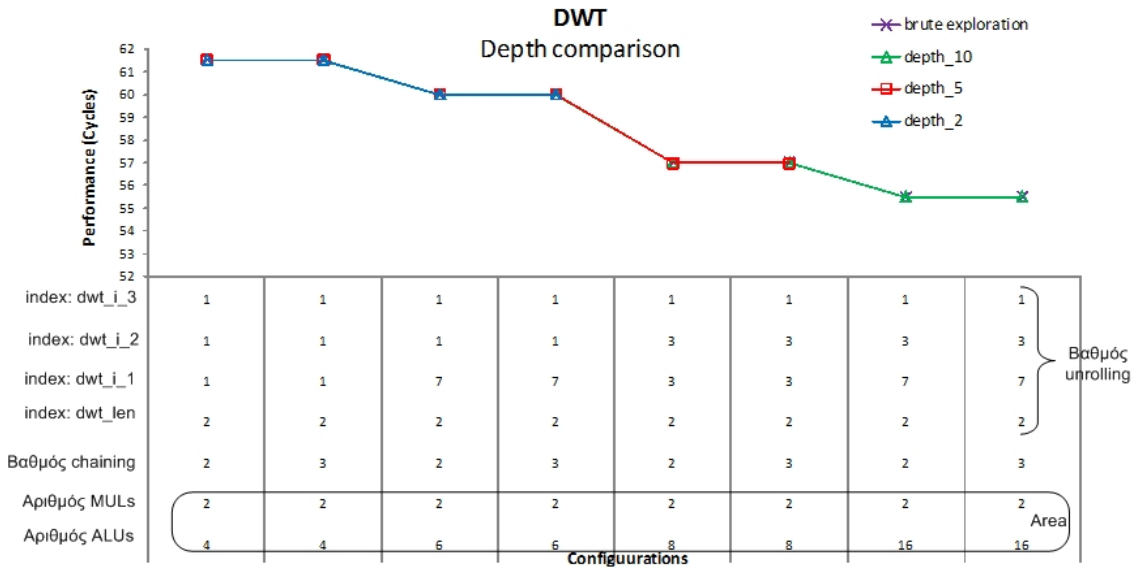
Πίνακας 4.2: Σύγκριση χρόνων εκτέλεσης και απόδοσης της εξαντλητικής εκτέλεσης και της έξυπνης εξερεύνησης με διαφορετικά βάθη αναζήτησης.

Τα αποτελέσματα και η σύγκριση των αποτελεσμάτων για τα διαφορετικά βάθη αναζήτησης για

τον πυρήνα YUVRGBA δίνονται στον Πίνακα 4.2. Φαίνεται ξεκάθαρα ότι με την αύξηση του βαθους αναζήτησης, αυξάνεται και το ποσοστό επιτυχίας, αλλά μειώνεται και η επιτάχυνση της εξερεύνησης.

DWT

Στο Σχήμα 4.9 συγκρίνονται διαφορετικά επίπεδα βάθους αναζήτησης (βάθος 2, 5 και 10) σε ότι αφορά την αποδοτικότητα τους (accurasy) στην εύρεση βέλτιστων λύσεων σε σχέση με την εξαντλητική εξερεύνηση.



Σχήμα 4.9: Γράφημα διαφορετικού βάθους αναζήτησης λύσεων για το πυρήνα DWT.

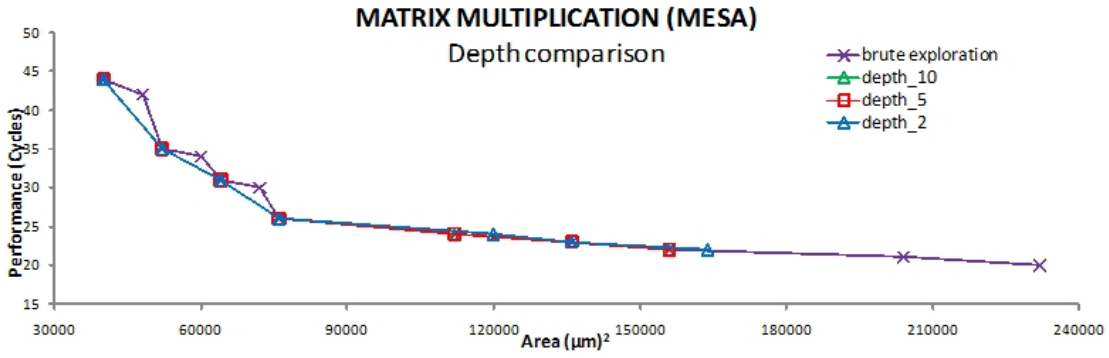
Τα αποτελέσματα και η σύγκριση των αποτελεσμάτων για τα διαφορετικά βάθη αναζήτησης για τον πυρήνα DWT δίνονται στον Πίνακα 4.3. Φαίνεται ξεκάθαρα ότι με την αύξηση του βαθους αναζήτησης, αυξάνεται και το ποσοστό επιτυχίας, αλλά μειώνεται και η επιτάχυνση της εξερεύνησης.

Τύπος εξερεύνησης	Βάθος εξερεύνησης	Χρόνος εκτέλεσης	Speedup	Accuracy
Εξαντλητική	—	27 hours, 54 minutes and 8 seconds (100448 seconds)	—	100%
”Εξυπνη”	2	0 hours, 19 minutes and 45 seconds (1185 seconds)	$\simeq$ x 85	50% (4/8)
”Εξυπνη”	5	1 hour, 0 minutes and 56 seconds (3656 seconds)	$\simeq$ x 27	75% (6/8)
”Εξυπνη”	10	1 hour, 37 minutes and 5 seconds (5825 seconds)	$\simeq$ x 17	100% (8/8)

Πίνακας 4.3: Σύγκριση χρόνων εκτέλεσης και απόδοσης της εξαντλητικής εκτέλεσης και της έξυπνης εξερεύνησης με διαφορετικά βάθη αναζήτησης.

MATRIX MULTIPLICATION (MESA)

Στο Σχήμα 4.10 συγκρίνονται διαφορετικά επίπεδα βάθους αναζήτησης (βάθος 2, 5 και 10) σε ότι αφορά την αποδοτικότητα τους (accurasy) στην εύρεση βέλτιστων λύσεων σε σχέση με την εξαντλητική εξερεύνηση.



Σχήμα 4.10: Γράφημα διαφορετικού βάθους αναζήτησης λύσεων για το πυρήνα MATRIX MULTIPLICATION (MESA).

Τα αποτελέσματα και η σύγκριση των αποτελεσμάτων για τα διαφορετικά βάθη αναζήτησης για τον πυρήνα DWT δίνονται στον Πίνακα 4.4. Φαίνεται ξεκάθαρα ότι με την αύξηση του βαθους αναζήτησης, αυξάνεται και το ποσοστό επιτυχίας, αλλά μειώνεται και η επιτάχυνση της εξερεύνησης.

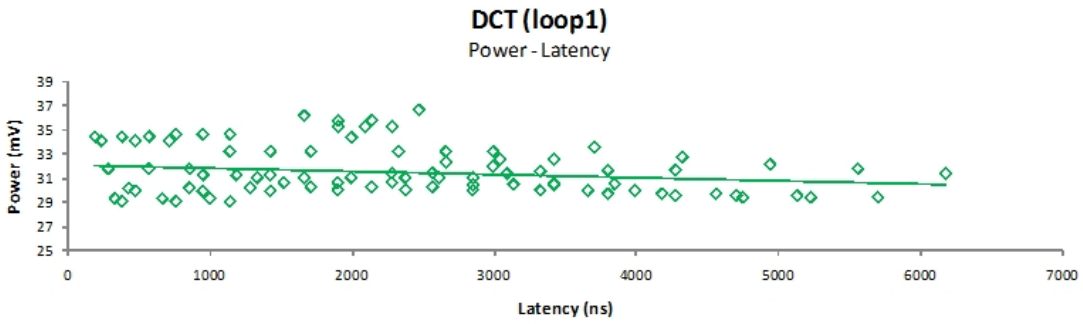
Τύπος εξερεύνησης	Βάθος εξερεύνησης	Χρόνος εκτέλεσης	Speedup	Accuracy
Εξαντλητική	—	0 hours, 30 minutes and 41 seconds (1841 seconds)	—	100%
”Έξυπνη”	2	0 hours, 1 minute and 58 seconds (118 seconds)	$\simeq$ x 16	42% (5/12)
”Έξυπνη”	5	0 hour, 3 minutes and 24 seconds (204 seconds)	$\simeq$ x 9	60% (7/12)
”Έξυπνη”	10	0 hour, 4 minutes and 18 seconds (258 seconds)	$\simeq$ x 7	60% (7/12)

Πίνακας 4.4: Σύγκριση χρόνων εκτέλεσης και απόδοσης της εξαντλητικής εκτέλεσης και της έξυπνης εξερεύνησης με διαφορετικά βάθη αναζήτησης.

4.2 Επίπεδο 2ο: RTL code to Post synthesis code

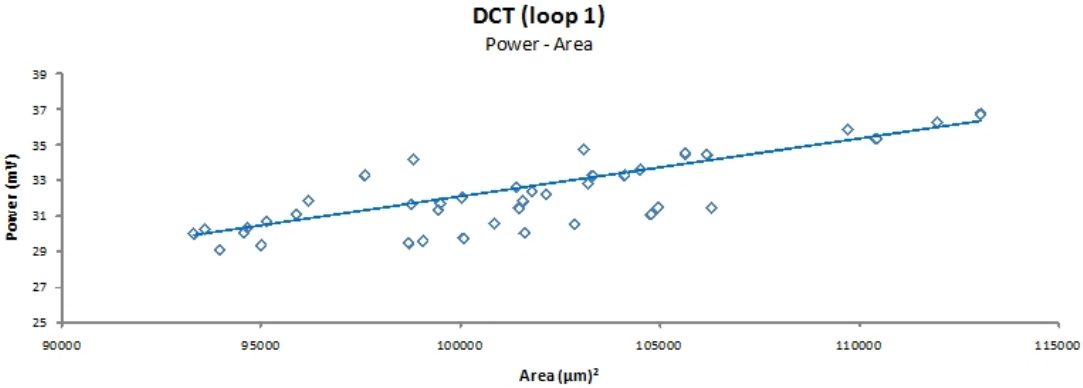
Στο δεύτερο επίπεδο της εξερεύνησης, χρησιμοποιούνται οι λύσεις από το πρώτο επίπεδο και δημιουργείται το αρχείο πυλών (net list) του κυκλώματος με χρήση του εργαλείου σύνθεσης *DC Shell*. Παρακάτω παρουσιάζονται γραφικές παραστάσεις με τα δεδομένα που προκύπτουν από το επιπέδου της σύνθεσης.
Σημειώνεται ότι λόγω του μεγάλου χρόνου που απαιτείται για την εξερεύνηση στο επίπεδο σύνθεσης (δημιουργία net-list) των λύσεων από το επίπεδο του HLS, παρουσιάζονται αποτελέσματα για δύο πυρήνες τα οποία είναι αρκετά για να εξαχθούν ασφαλή συμπεράσματα.

DCT_Loop1 (JPEG)



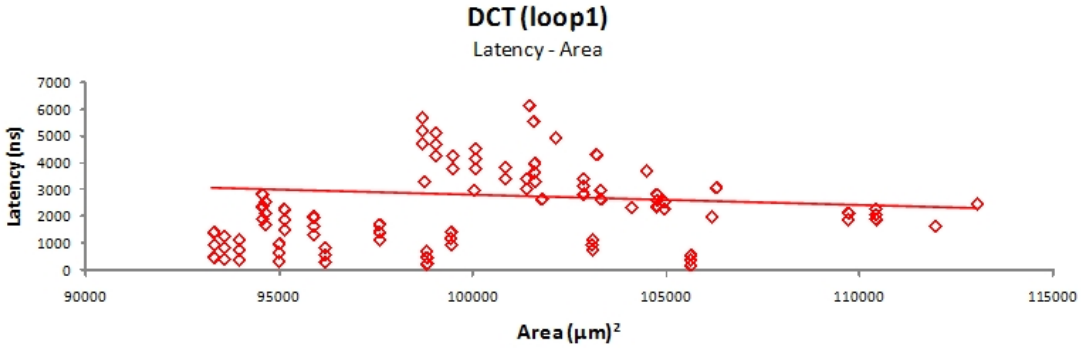
Σχήμα 4.11: Γράφημα της κατανάλωσης ισχύς σε σχέση με το χρόνο εκτέλεσης.

Στο σχήμα 4.11 παρουσιάζεται η εξάρτηση μεταξύ της κατανάλωσης ισχύος και χρόνου εκτέλεσης του κυκλώματος (Latency = Clock * Cycles). Με την ευθεία γραμμή παρουσιάζεται η αντιστρόφως ανάλογη σχέση που έχουν οι δύο παράγοντες, με την αύξηση του χρόνου εκτέλεσης μειώνεται η κατανάλωση ισχύς στο κύκλωμα.



Σχήμα 4.12: Γράφημα της κατανάλωσης ισχύς σε σχέση με το επιφάνειας υλικού.

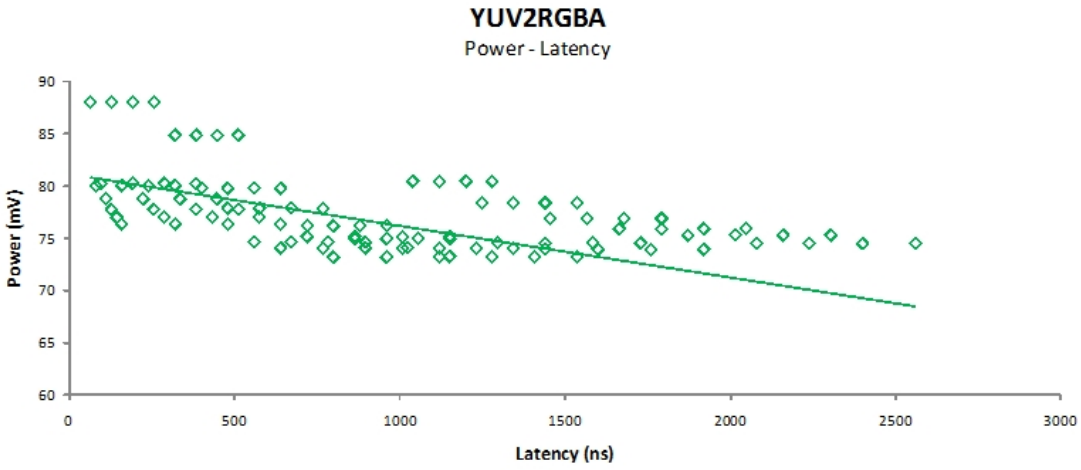
Η σχέση μεταξύ κατανάλωσης ισχύος και επιφάνειας υλικού ενός κυκλώματος φαίνεται στο σχήμα 4.12. Οι δύο παράγοντες είναι ανάλογοι (δηλαδή αύξηση της επιφάνειας υλικού επιφέρει και αύξηση στην κατανάλωση ισχύς στο κύκλωμα), κάτι το οποίο φαίνεται στο σχήμα και με την ευθεία γραμμή που υπάρχει ανάμεσα στα σημεία.



Σχήμα 4.13: Γράφημα του χρόνου εκτέλεσης σε σχέση με την επιφάνεια υλικού.

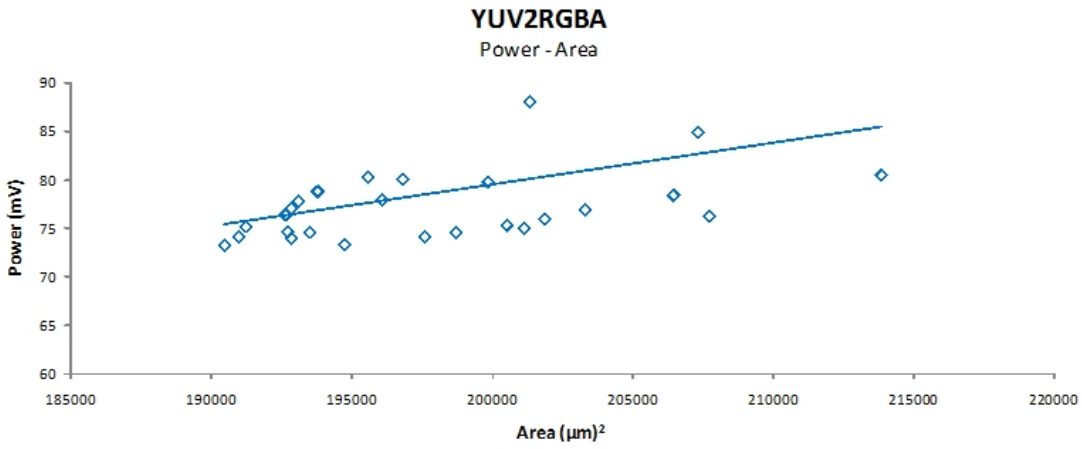
Τέλος στο σχήμα 4.13 παρουσιάζεται το γράφημα για το χρόνο εκτέλεσης σε σχέση με την επιφάνεια υλικού. Στο συγκεκριμένη γραφική παράσταση τα αποτελέσματα δεν είναι συγκεντρωμένα ώστε να εξαχθούν έγκυρα συμπεράσματα, να παρουσιαστεί δηλαδή η αντιστρόφως ανάλογη σχέση μεταξύ των δύο παραγόντων. Εντούτοις η αντιστρόφως ανάλογη σχέση του χρόνου εκτέλεσης και επιφάνειας υλικού φαίνεται ξεκάθαρα με την ευθεία που παρουσιάζεται.

YUV2RGBA

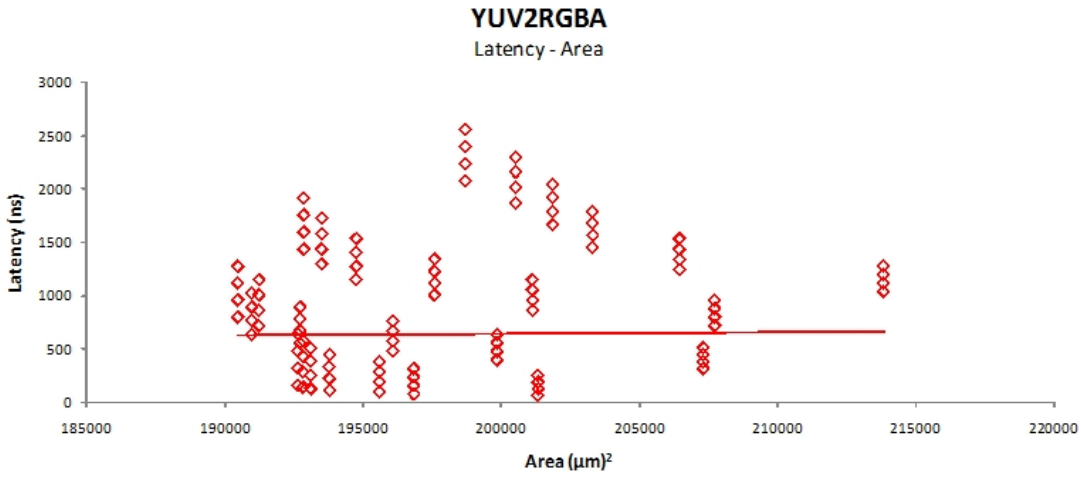


Σχήμα 4.14: Γράφημα της κατανάλωσης ισχύς σε σχέση με το χρόνο εκτέλεσης.

Στο σχήμα 4.14 παρουσιάζεται η γραφική παράσταση για την κατανάλωσης ισχύος και το χρόνου εκτέλεσης του κυκλώματος (Latency = Clock * Cycles). Με την ευθεία γραμμή παρουσιάζεται η αντιστρόφως ανάλογη σχέση που έχουν οι δύο παράγοντες, με την αύξηση του χρόνου εκτέλεσης μειώνεται η κατανάλωση ισχύς στο κύκλωμα.
Η σχέση μεταξύ κατανάλωσης ισχύος και επιφάνειας υλικού ενός κυκλώματος φαίνεται στο σχήμα 4.15. Οι δύο παράγοντες είναι ανάλογοι (δηλαδή αύξηση της επιφάνειας υλικού επιφέρει και αύξηση στην κατανάλωση ισχύς στο κύκλωμα), κάτι το οποίο φαίνεται στο σχήμα και με την ευθεία γραμμή που υπάρχει ανάμεσα στα σημεία.



Σχήμα 4.15: Γράφημα της κατανάλωσης ισχύς σε σχέση με το επιφάνειας υλικού.



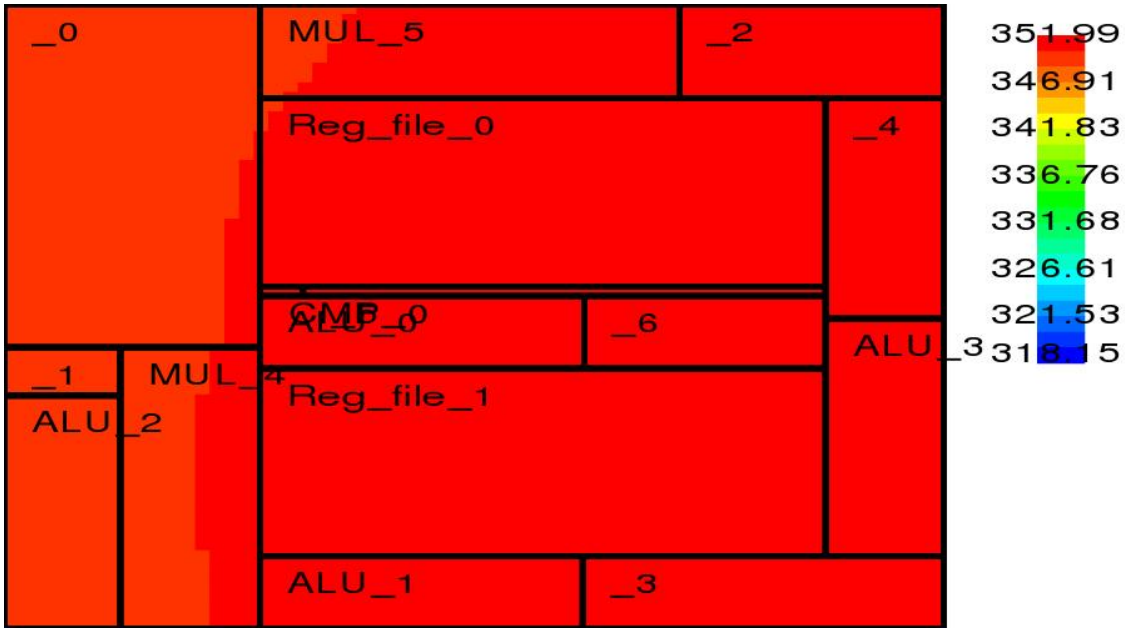
Σχήμα 4.16: Γράφημα του χρόνου εκτέλεσης σε σχέση με την επιφάνεια υλικού.

Τέλος στο σχήμα 4.16 παρουσιάζεται το γράφημα για το χρόνο εκτέλεσης σε σχέση με την επιφάνεια υλικού. Στο συγκεκριμένη γραφική παράσταση τα αποτελέσματα δεν είναι συγκεντρωμένα ώστε να εξαχθούν έγκυρα συμπεράσματα, να παρουσιαστεί δηλαδή η αντιστρόφως ανάλογη σχέση μεταξύ των δύο παραγόντων. Εντούτοις η αντιστρόφως ανάλογη σχέση του χρόνου εκτέλεσης και επιφάνειας υλικού φαίνεται ξεκάθαρα με την ευθεία που παρουσιάζεται.

4.3 Επίπεδο 3ο: Power Data to Thermal Floorplan - Management

Για το τρίτο επίπεδο του Framework, δημιουργήθηκαν οι θερμικοί χάρτες για ενδεικτικές λύσεις με δεδομένα ισχύος και επιφάνειας υλικού από το δεύτερο επίπεδο. Στο συγκεκριμένο στάδιο χρησιμοποιήθηκαν τα εργαλεία HotFloorplan και HotSpot.

DCT_Loop1 (JPEG)

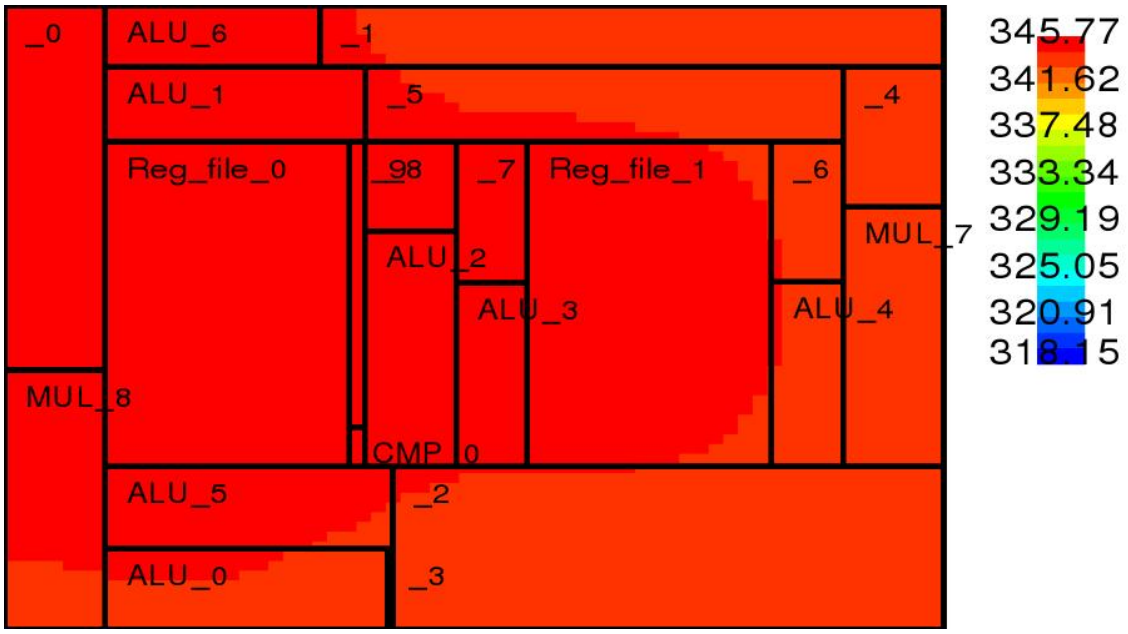


Σχήμα 4.17: Θερμική εικόνα χωροθετημένου σχεδίου για τον πυρήνα DCT(loop1) με αριθμό λειτουργικών μονάδων: 4 ALUs, 2 MULs, 1 CMP, 2 Register Files.

ALUs: 4, MULs: 2, Chaining: 1, Unrolling: 0 / Synthesis Clock: 2. Στο Σχήμα 4.17 παρουσιάζεται η θερμική εικόνα για το πυρήνα DCT_loop1 (JPEG). Τα δεδομένα επιφάνειας υλικού και ισχύος προέρχονται από το σύνολο των βέλτιστων λύσεων στο πυρήνα *DCT (loo1)* με αριθμό αθροιστών 4, αριθμό πολλαπλασιαστών 4, βαθμό chaining 1 και χωρίς unrolling. Για το συγκεκριμένη χωροθέτηση, υπήρχαν λειτουργικές μονάδες 4 ALUs, 2 MULs, 1 CMP και 2 Register Files. Το εργαλείο χωροθέτησης τοποθέτησε επίσης 7 κενά "κομμάτια" στο σχέδιο για να καλύψει την καλωδίωση του κυκλώματος. Η μέγιστη θερμοκρασία που αναπτύσσεται στο κύκλωμα είναι 78.84 °C (351.99 Kelvin), ενώ η μέση θερμοκρασία που επικρατεί στο υπόλοιπο μέρος είναι $\simeq 70\,^{\circ}\mathrm{C}$.

Η συνολική επιφάνεια υλικού για τ συγκεκριμένο κύκλωμα είναι $105646.0\mu m^2$ και η συνολική κατανάλωση ισχύος είναι $34.506mV$.

ALUs: 7, MULs: 2, Chaining: 3, Unrolling: 0 / Synthesis Clock: 2. Στο Σχήμα 4.17 παρουσιάζεται μία δεύτερη θερμική εικόνα για το πυρήνα DCT_loop1 (JPEG). Η λύση από το επίπεδο του HLS καθόριζε αριθμό αθροιστών 7, αριθμό πολλαπλασιαστών 4, βαθμό chaining 3 και χωρίς unrolling. Για το συγκεκριμένη χωροθέτηση, υπήρχαν λειτουργικές μονάδες 7 ALUs, 2 MULs, 1 CMP και 2 Register Files. Το εργαλείο χωροθέτησης τοποθέτησε επίσης 10 κενά "κομμάτια" στο σχέδιο για να καλύψει την καλωδίωση του κυκλώματος. Η μέγιστη θερμοκρασία



Σχήμα 4.18: Θερμική εικόνα χωροθετημένου σχεδίου για τον πυρήνα DCT(loop1) με αριθμό λειτουργικών μονάδων: 4 ALUs, 2 MULs, 1 CMP, 2 Register Files.

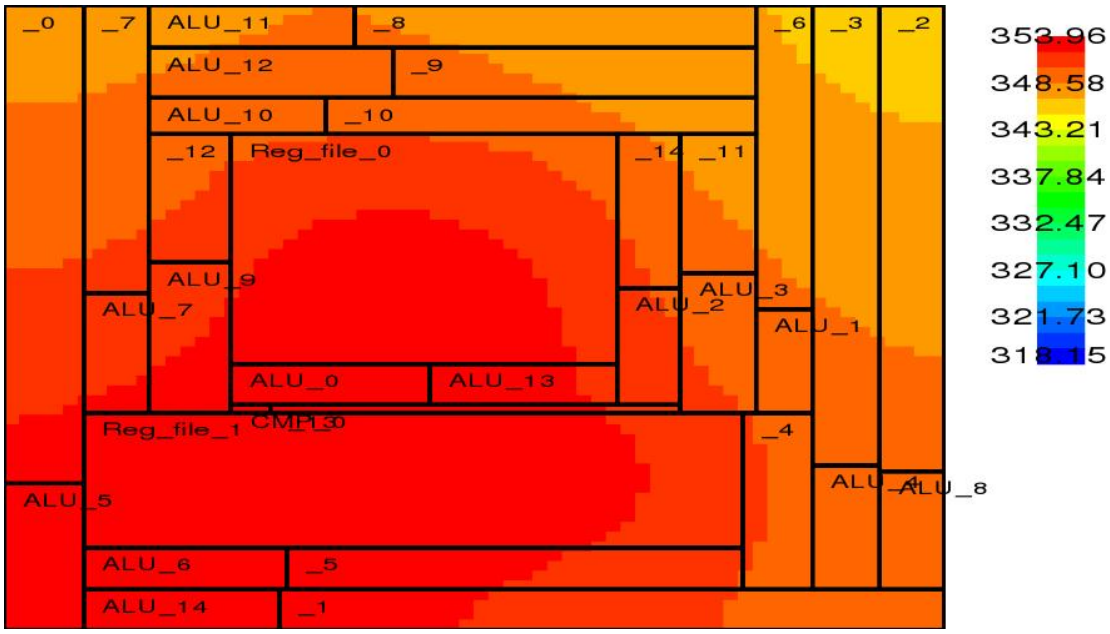
που αναπτύσσεται στο κύκλωμα είναι 72.62 °C (345.77 Kelvin), ενώ η μέση θερμοκρασία που επικρατεί στο υπόλοιπο μέρος είναι $\simeq$ 66 °C.

Η συνολική επιφάνεια υλικού για τ συγκεκριμένο κύκλωμα είναι $110436.0\mu m^2$ και η συνολική κατανάλωση ισχύος είναι $35.346mV$.

YUV2RGBA_U8

ALUs: 15, MULs: 2, Chaining: 3, Unrolling: 7 / Synthesis Clock: 2.5. Η χωροθέτηση του κυκλώματος που αντιστοιχεί με πυρήνα *YUV2RGBA* με τη θερμική εικόνα που παρουσιάζει φαίνεται στο Σχήμα 4.19. Οι παράμετροι που καθορίστηκαν από τη λύση του επιπέδου εξερεύνησης του HLS προϋποθέτει 15 μονάδες αθροιστές, 2 πολλαπλασιαστές, βαθμό chaining 3 και βαθμό unrolling 7 στην παράμετρο του βρόγχου στον πυρήνα. Όπως φαίνεται όμως, στο κύκλωμα δεν υπάρχουν πολλαπλασιαστές. Ο λόγος είναι ότι ο design compiler στο 2ο επίπεδο στη δημιουργία του δικτύου πυλών χρησιμοποίησε τις διαθέσιμες μονάδες αθροιστών για την εκτέλεση των λειτουργιών πολλαπλασιασμού που υπάρχουν στον πυρήνα.

Για το συγκεκριμένη χωροθέτηση, υπήρχαν λειτουργικές μονάδες 15 ALUs, 1 CMP και 2 Register Files. Το εργαλείο χωροθέτησης τοποθέτησε επίσης 15 κενά "κομμάτια" στο σχέδιο για να καλύψει την καλωδίωση του κυκλώματος. Η μέγιστη θερμοκρασία που αναπτύσσεται στο κύκλωμα είναι 80.81 °C (353.96 Kelvin), ενώ η μέση θερμοκρασία που επικρατεί στο υπόλοιπο μέρος είναι $\simeq$ 70 °C.



Σχήμα 4.19: Θερμική εικόνα χωροθετημένου σχεδίου για τον πυρήνα YUV2RGBA με αριθμό λειτουργικών μονάδων: 15 ALUs, 1 CMP, 2 Register Files.

Η συνολική επιφάνεια υλικού για το συγκεκριμένο κύκλωμα είναι $213863.000\mu m^2$ και η συνολική κατανάλωση ισχύος είναι $80.469mV$.

4.3.1 Θερμικοί χάρτες για κοινη επιφάνεια υλικού και διαφορετικά code motions

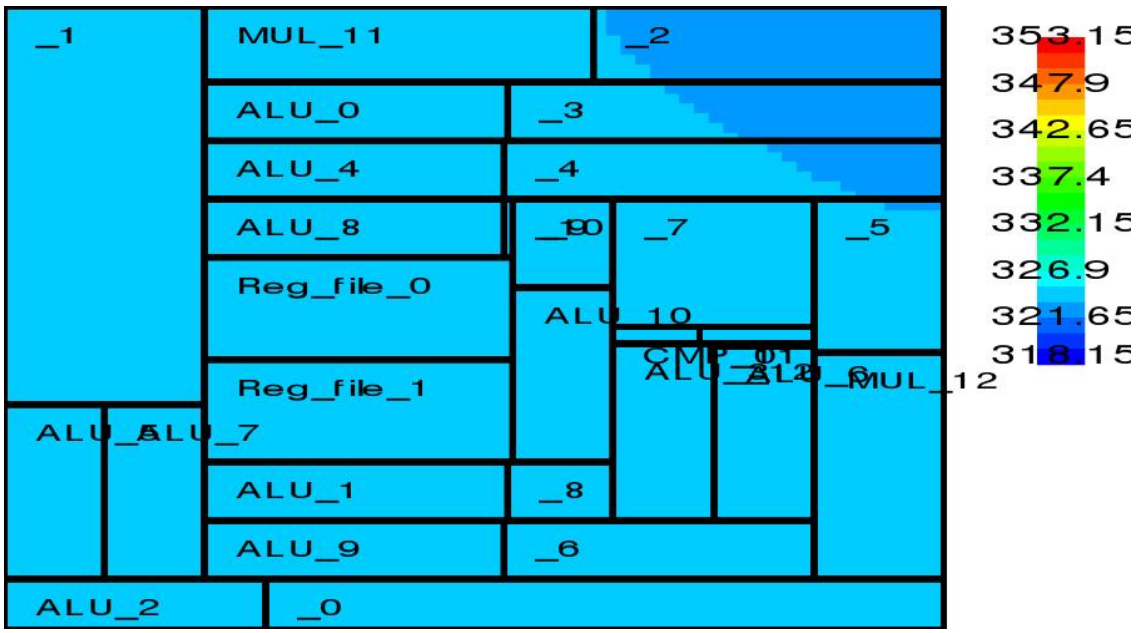
Ενδιαφέρον παρουσιάζει η επίδραση που έχει η εφαρμογή code motions στο επίπεδο του HLS για τη θερμική εικόνα του κυκλώματος που δημιουργείται στο επίπεδο του Floorplanning. Παρουσιάζονται παρακάτω οι θερμικοί χάρτες για τον πυρήνα *yuv2rgba* με κοινή επιφάνεια υλικού (11 ALUs, 2 MULs) στις 4 ακόλουθες περιπτώσεις:

1. Chaining = 0, Unrolling = 0
2. Chaining = 2, Unrolling = 0
3. Chaining = 0, Unrolling = 6
4. Chaining = 2, Unrolling = 6

Σημειώνουμε πως στη σύνθεση του κυκλώματος, το net list και οι αναφορών για επιφάνεια υλικού και κατανάλωση ισχύος δεν περιλάμβαναν πολλαπλασιαστές στην υλοποίηση. Γίνεται αντιληπτό πως ο design compiler χρησιμοποίησε τις διαθέσιμες μονάδες αθροιστών για την εκτέλεση των λειτουργιών πολλαπλασιασμού που υπάρχουν στον πυρήνα.
Για τη δημιουργία των θερμικών χαρτών, προστέθηκαν και οι μονάδες των πολλαπλασιαστών για να μοντελοποιηθεί η αρχιτεκτονική CGRArch. Στις μονάδες που προστέθηκαν σαν κατανάλωση ισχύος δόθηκε το Leakage Power του κυκλώματος.

ALUs: 11, MULs: 2, Chaining: 0, Unrolling: 0 / Synthesis Clock: 5. Η πρώτη θερμική εικόνα που παρουσιάζεται (Σχήμα 4.20) αφορά την περίπτωση με Chaining = 0 και Unrolling

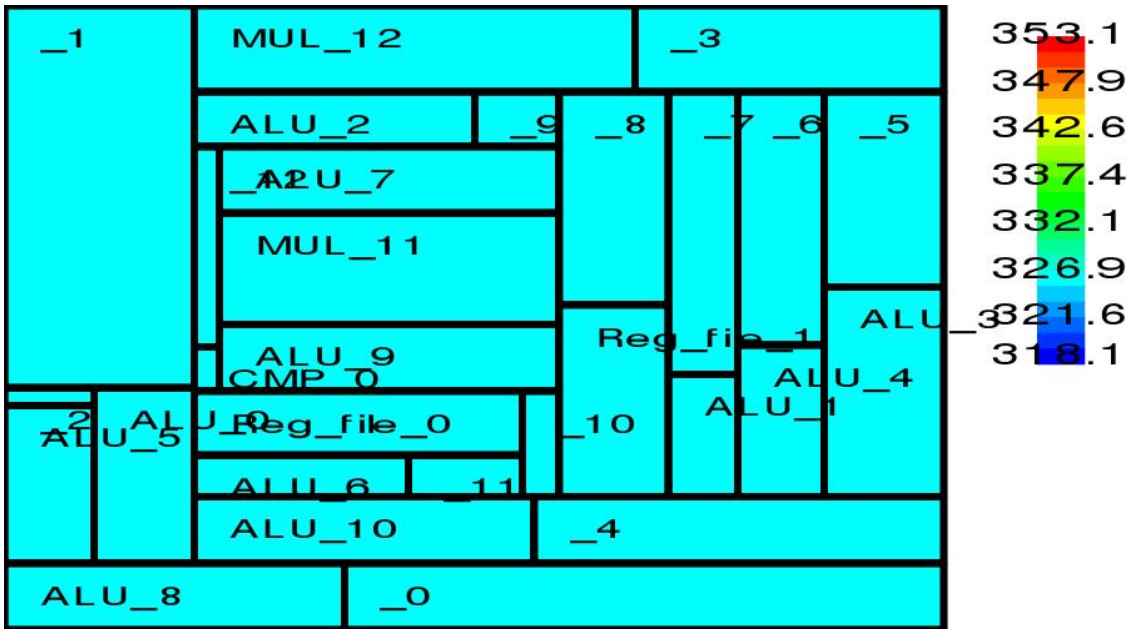
= 0. Τα αποτελέσματα για την επιφάνεια υλικού και την κατανάλωση ισχύος που παράχθηκαν από τη σύνθεση έδειχναν τη χρησιμοποίηση μόνο 4 ALUs από του 11 που ήταν διαθέσιμοι. Όπως αναφέρθηκε σε προηγούμενη παράγραφο, προστέθηκαν και οι υπόλοιπες μονάδες ALU μαζί με τις 2 MUL με κατανάλωση ισχύος τη Leakage Power του κυκλώματος.
Είναι εμφανής η μικρή χρήση των λειτουργικών μονάδων από τη χαμηλή θερμοκρασία που αναπτύσσεται στο κύκλωμα, με μέγιστη θερμοκρασία τους 51.09 °C (324.24 Kelvin), ενώ η μέση θερμοκρασία που επικρατεί στο υπόλοιπο μέρος είναι ≃ 49 °C. Οι κύκλοι της συγκεκριμένης υλοποίησης είναι 57.



Σχήμα 4.20: Θερμική εικόνα χωροθετημένου σχεδίου για τον πυρήνα YUV2RGBA με αριθμό λειτουργικών μονάδων: 11 ALUs, 2 MULs, 1 CMP, 2 Register Files. Chaining = 0, Unrolling = 0.

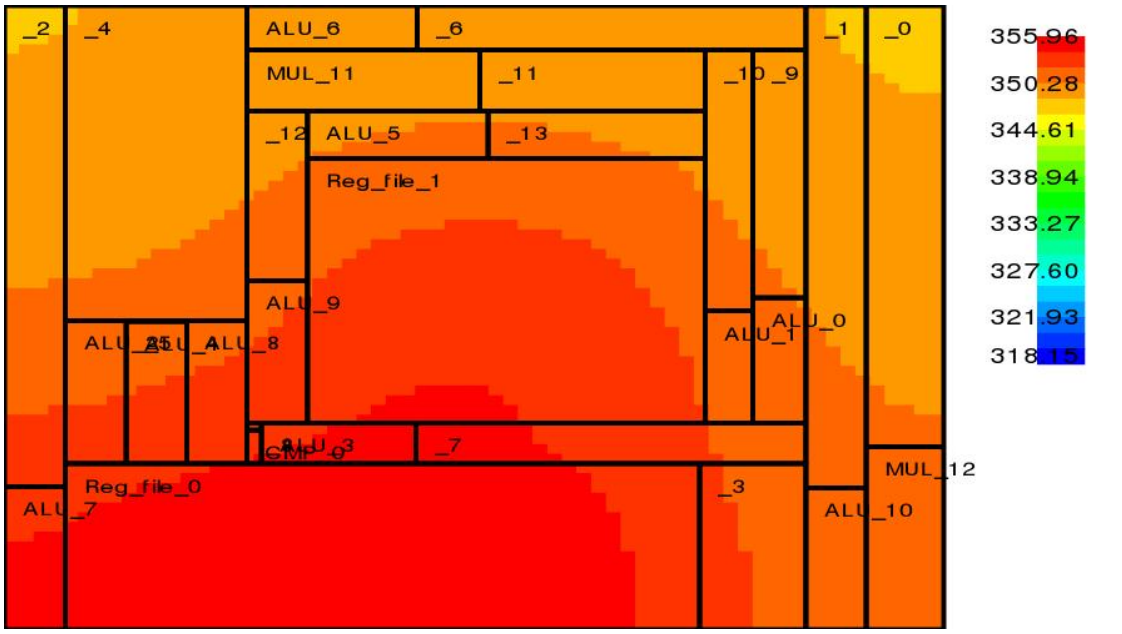
ALUs: 11, MULs: 2, Chaining: 2, Unrolling: 0 / Synthesis Clock: 5. Η θερμική εικόνα που παρουσιάζεται Σχήμα 4.21 αφορά την περίπτωση με Chaining = 2 και Unrolling = 0. Τα αποτελέσματα για την επιφάνεια υλικού και την κατανάλωση ισχύος που παράχθηκαν από τη σύνθεση έδειχναν τη χρησιμοποίηση μόνο 7 ALUs από του 11 που ήταν διαθέσιμοι. Και στην περίπτωση αυτή προστέθηκαν οι επιπλέον μονάδες για τη δημιουργία του θερμικού χάρτη οπώς αναφέρθηκε προηγουμένως.
Και στην συγκεκριμένη περίπτωση είναι εμφανής η μικρή χρήση των λειτουργικών μονάδων από τη χαμηλή θερμοκρασία που αναπτύσσεται στο κύκλωμα, με μέγιστη θερμοκρασία τους 52.95 °C (326.10 Kelvin), ενώ η μέση θερμοκρασία που επικρατεί στο υπόλοιπο μέρος είναι ≃ 51 °C. Οι κύκλοι της συγκεκριμένης υλοποίησης είναι 43.

ALUs: 11, MULs: 2, Chaining: 0, Unrolling: 6 / Synthesis Clock: 5. Η θερμική εικόνα που παρουσιάζεται Σχήμα 4.22 αφορά την περίπτωση με Chaining = 0 και Unrolling = 6. Τα αποτελέσματα για την επιφάνεια υλικού και την κατανάλωση ισχύος που παράχθηκαν από τη σύνθεση έδειχναν τη χρησιμοποίηση και οι 11 αθροιστές που ήταν διαθέσιμοι.
Σ'αυτή την περίπτωση παρουσιάζεται αυξημένη θερμοκρασία στο κύκλωμα, κάτι που δείχνει τη μεγάλη χρήση των λειτουργικών μονάδων. Μέγιστη θερμοκρασία τους 82.81 °C (355.96 Kelvin),



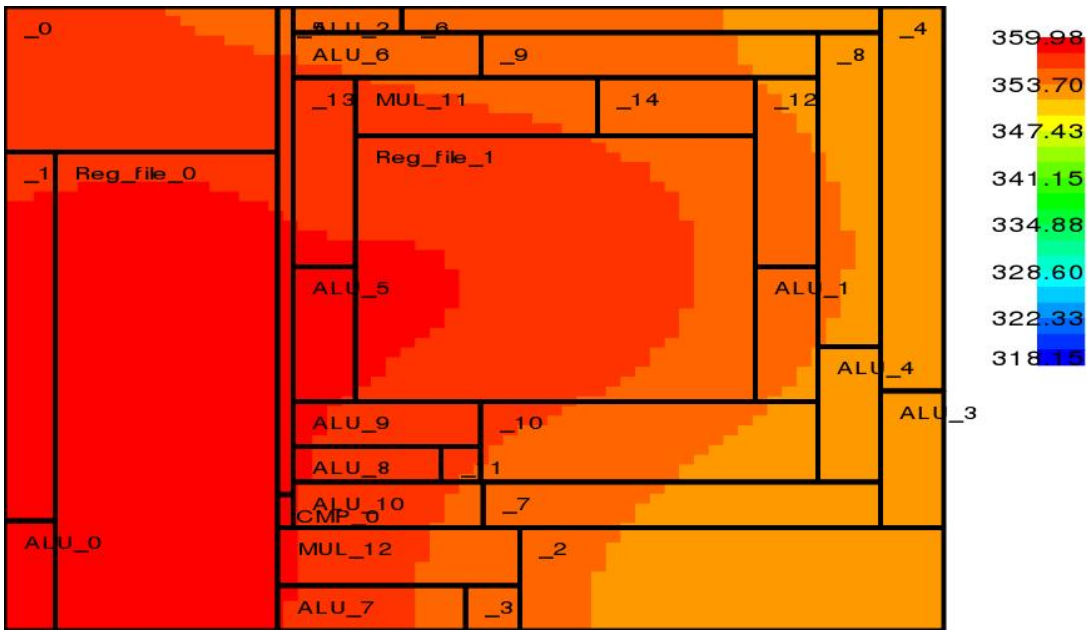
Σχήμα 4.21: Θερμική εικόνα χωροθετημένου σχεδίου για τον πυρήνα YUV2RGBA με αριθμό λειτουργικών μονάδων: 11 ALUs, 2 MULs, 1 CMP, 2 Register Files. Chaining = 2, Unrolling = 0.

ενώ η μέση θερμοκρασία που επικρατεί στο υπόλοιπο μέρος είναι $\simeq 74\,^{\circ}\mathrm{C}$. Οι κύκλοι της συγκεκριμένης υλοποίησης είναι 38.



Σχήμα 4.22: Θερμική εικόνα χωροθετημένου σχεδίου για τον πυρήνα YUV2RGBA με αριθμό λειτουργικών μονάδων: 11 ALUs, 2 MULs, 1 CMP, 2 Register Files. Chaining = 0, Unrolling = 6.

ALUs: 11, MULs: 2, Chaining: 2, Unrolling: 6 / Synthesis Clock: 5. Τέλος στο Σχήμα 4.23 παρουσιάζεται η περίπτωση με Chaining = 2 και Unrolling = 6. Τα αποτελέσματα για την επιφάνεια υλικού και την κατανάλωση ισχύος που παράχθηκαν από τη σύνθεση έδειχναν τη χρησιμοποίηση και οι 11 αθροιστές που ήταν διαθέσιμοι.
Και σ'αυτή την περίπτωση είναι εμφανής η μεγάλη χρήση των λειτουργικών μονάδων από τη υψηλή θερμοκρασία που αναπτύσσεται στο κύκλωμα, με μέγιστη θερμοκρασία τους 86.83 °C (359.98 Kelvin), ενώ η μέση θερμοκρασία που επικρατεί στο υπόλοιπο μέρος είναι ≃ 78 °C. Οι κύκλοι της συγκεκριμένης υλοποίησης είναι 39.

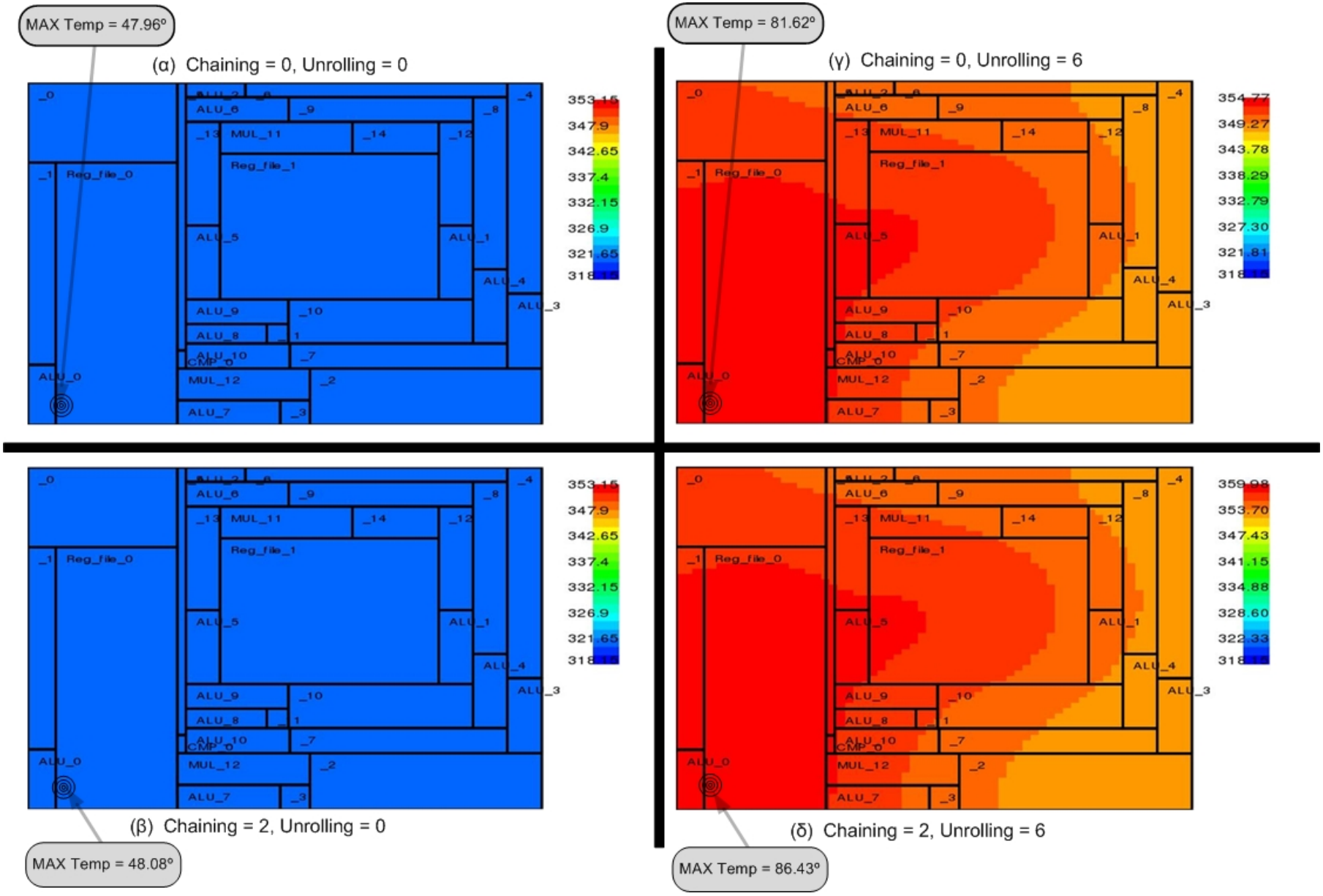


Σχήμα 4.23: Θερμική εικόνα χωροθετημένου σχεδίου για τον πυρήνα YUV2RGBA με αριθμό λειτουργικών μονάδων: 11 ALUs, 2 MULs, 1 CMP, 2 Register Files. Chaining = 2, Unrolling = 6.

Παρατηρώντας τις πάραπάνω θερμικές εικόνες (Σχήματα 4.20 - 4.23) φαίνεται ξεκάθαρα η επίδραση που έχει η εφαρμογή μετασχηματισμών κώδικα στη θερμική κατάσταση του κυκλώματος. Εφαρμόζοντας unrolling στο κώδικα, αυξάνονται οι λειτουργίες, γι' αυτό και γίνεται χρησιμοποίηση όλων των αθροιστών που είναι διαθέσιμοι για τη σύνθεση. Αυτό μειώνει τους κύκλους λειουργίας του σχεδίου (latency) και αυξάνει τη καταναλωση ισχύος στην κάθε μονάδα. Γι'αυτό και υπάρχει αναπτύσσεται μεγαλύτερη θερμότητα στο κύκλωμα.

Κοινό Floorplan

Στις προηγούμενες παραγράφους, (Σχήματα 4.20 - 4.23), το εργαλείο του HotFloorplan δημιουργεί διαφορετικό χωροσχέδιο (floorplan) για κάθε περίπτωση ανάλογα με τα δεδομένα επιφάνειας υλικού και κατανάλωση ισχύος. Στο Σχήμα 4.24 χρησιμοποιείται κοινό floorplan (αύτο τις περίπτωσης με Chaining = 2, Unrolling = 6) ενισχύεται το συμπέρασμα για την επίδραση των code motions (και ιδιαίτερα του unrolling) στη θερμική εικόνα του κυκλώματος.



Σχήμα 4.24: Θερμικοί χάρτες με κοινό floorplan (11 ALUs, 2 MULs) για τις περιπτώσεις (α) Chaining = 0, Unrolling 0 (β) Chaining = 2, Unrolling 0, (γ) Chaining = 0, Unrolling 6 και (δ) Chaining = 2, Unrolling 6.

Κεφάλαιο 5

Συμπεράσματα - Μελλοντικές προεκτάσεις

Η διπλωματική αυτή εργασία παρουσίασε την ανάπτυξη ενός ολοκληρωμένου περιβάλλοντος για τη μοντελοποίηση και αυτοματοποιημένη εξερεύνηση του χώρου σχεδίασης συνεπεξεργαστών υλικού. Το προτεινόμενο περιβάλλον εργασίας προσφέρει αυτοματοποιημένη καταγραφή για όλα τα σύγχρονα σχεδιαστικά κριτήρια (επιφάνεια υλικού, επίδοση, κατανάλωση ισχύος, θερμική συμπεριφορά). Τέσσερις βασικοί άξονες συνθέτουν το σκοπό αυτής της διπλωματικής εργασίας. Συγκεκριμένα *(i)* η σύσταση ροής σχεδίασης από το επίπεδο του High Level Synthesis σε Post-Synthesis Floorplan, δηλαδή ενός HLS Framework, *(ii)* η μοντελοποίηση αρχιτεκτονικών CGRArch, *(iii)* η επέκταση του πλαισίου εργασίας για εξερεύνηση με πολλαπλά αφαιρετικά επίπεδα σχεδίασης και *(iv)* η επέκταση του χώρου σχεδίασης για HLS, μοντελοποίηση και εξερεύνηση. Και οι τέσσερις βασικοί αυτοί άξονες αναλύθηκαν στο κείμενο που προηγήθηκε και οι προτεινόμενες υλοποιήσεις παρουσιάστηκαν μαζί με τα πειραματικά αποτελέσματα που υποστηρίζουν αυτές.

Σύμφωνα με την εξερεύνηση στο επίπεδο HLS (Σχήμα 3.5) αλλά και από τα αποτελέσματα (Κεφάλαιο 4.1) συμπεραίνετε η ύπαρξη καλύτερων λύσεων στον επαυξημένο χώρο του HLS, το χώρο που συμπεριλαμβάνει εκτός από τον αριθμό των λειτουργιών μονάδων (#ALUs και #MULs) το βαθμό chaining και unrolling. Παρατηρείται ακόμα, ότι η εφαρμογή μετασχηματισμών κώδικα στο HLS exploration δίνει καλύτερες λύσεις από ότι η μη εφαρμογή τους.

Επιπλέον σύμφωνα με την υλοποίηση της εξερεύνησης που προτείνεται, το βάθος αναζήτησης (gradient search) για την "έξυπνη" εξερεύνηση πρέπει να είναι τέτοιο ώστε το αποτέλεσμα να ισορροπεί ανάμεσα στην επιτυχή εύρεση βέλτιστων λύσεων (σε σχέση με την εξαντλητική εξερεύνηση) αλλά και στην ταχύτητα ολοκλήρωσης της εξερεύνησης. Το βάθος ορίστηκε σε 5, το οποίο σύμφωνα και με τα αποτελέσματα που παρουσιάζονται στο Κεφάλαιο 4.1.2. αποτελεί δίνει μεγάλα ποσοστά εύρεσης βέλτιστων λύσεων και χαμηλό χρόνο εκτέλεσης της εξερεύνησης (μεγάλο speedup).

Με βάση τους θερμικούς χάρτες που παρουσιάστηκαν στο κεφάλαιο 4.3 παρατηρείται επίδραση των μετασχηματισμών κώδικα (code motions) στη θερμική κατάσταση του κυκλώματος. Εφαρμόζοντας μετασχηματισμούς κώδικα αυξάνονται οι διαθέσιμες λειτουργίες και γίνεται εκμετάλλευση όλων των λειτουργικών μονάδων. Αποτέλεσμα είναι η αυξημένη κατανάλωση ισχύος από τις μονάδες και η άνοδος της θερμοκρασίας του κυκλώματος.

Η παρούσα εργασία μπορεί να αποτελέσει τη βάση για διάφορες μελλοντικές προεκτάσεις, κάποιες από τις οποίες αναφέρονται στη συνέχεια.

Μια προτεινόμενη προέκταση στο υπάρχον HLS Framework που παρουσιάζεται σε αυτή τη διπλωματική εργασία είναι η υλοποίηση ενός modular software design για inter-tool integration. Με αυτό το modular θα είναι δυνατή η χρησιμοποίηση του HLS exploration που υλοποιήθηκε με διάφορα άλλα εργαλεία HLS εκτός από το SPARK που χρησιμοποιήθηκε.

Η HLS εξερεύνηση υιοθετεί τις αρχές της χρονοδρομολόγησης για ελαχιστοποίηση της καθυστέρησης βάσει περιορισμών επιφάνειας (minimum latency resource-constrained scheduling). Περιορισμός της επιφάνειας υλικού (Area constrains) και εύρεση περιορισμού χρόνου (Time constrain) για διάφορους πυρήνες. Η αντιμετώπιση του αντίστοιχου δυαδικού προβλήματος, δηλαδή του περιορισμού χρόνου και εύρεση περιορισμού επιφάνειας υλικού, στην ήδη υπάρχουσα καμπύλη των βέλτιστων λύσεων θα μπορούσε να αποτελέσει μια επιπλέον προέκταση.

Η προσθήκη και του παράγοντα ισχύος (Power) στο επίπεδο της εξερεύνησης HLS με κατάλληλα μοντέλα θα έδινε μια επιπλέον διάσταση στην εύρεση βέλτιστων λύσεων που παράγονται από το 1ο επίπεδο, και η εξερεύνηση γινόταν με τη χρήση 3 παραγόντων, Latency, Area και Power. Αποτέλεσμα θα ήταν μία τρισδιάστατη καμπύλη (η οποία θα χωρίζεται σε 3 δισδιάστατες) και εύρεση βέλτιστων λύσεων με βάση και τους 3 παράγοντες.

Ακόμη, άλλος ένας παράγοντας που θα μπορούσε να εισαχθεί στο εργαλείο του HLS είναι δεδομένα για θερμοκρασία στο κύκλωμα που παράγονται στο 3ο επίπεδο. Μια τέτοια προέκταση μαζί τη δυνατότητα για rescheduing και rebinding στο επίπεδο του HLS, θα βελτίωνε λύσεις που στο επίπεδο της χωροθέτησης παρουσιάζουν λειτουργικές μονάδες με αυξημένη θερμοκρασία.

Άλλη μία προτεινόμενη προέκταση είναι η εφαρμογή Software Pipelining στην μοντελοποίηση και την εξερεύνηση στο επίπεδο της χρονοδρομολόγησης (scheduling) και όχι σ' αυτό των code motions.

Τέλος δεδομένου ότι το Design Space Exploration παράγει pareto points (βέλτιστες λύσεις) της ίδιας συμπεριφοράς, αυτά θα μπορούσαν να χρησιμοποιηθούν σε επίπεδο runtimes scenarios για την αντιμετώπιση προβλημάτων αξιοπιστίας στα μελλοντικά Systems-on-Chips (SOCs).

Βιβλιογραφία

[Gerl] J. Gerlach and W. Rosenstiel, "A scalable methodology for cost estimation in a transformational high-level design space exploration environment", in *DATE*, vol. 98, pp. 226–231.

[Gerl00] J. Gerlach and W. Rosenstiel, "A Methodology and Tool for Automated Transformational High-Level Design Space Exploration", in *ICCD '00: Proceedings of the 2000 IEEE International Conference on Computer Design*, p. 545, Washington, DC, USA, 2000, IEEE Computer Society.

[Gu06] Z.P. Gu, Y. Yang, J. Wang, R.P. Dick and L. Shang, "TAPHS: Thermal-aware unified physical-level and high-level synthesis", in *Proceedings of the 2006 conference on Asia South Pacific design automation*, pp. 879–885, IEEE Press Piscataway, NJ, USA, 2006.

[Gupt04] S. Gupta, *SPARK: a parallelizing approach to the high-level synthesis of digital circuits*, Kluwer Academic Publishers, 2004.

[Han05] Y. Han, I. Koren and C.A. Moritz, "Temperature aware floorplanning", in *Proc. of the Second Workshop on Temperature-Aware Computer Systems*, 2005.

[Huan01] Z. Huang and S. Malik, "Managing dynamic reconfiguration overhead in systems-on-a-chip design using reconfigurable datapaths and optimized interconnection networks", in *Proceedings of the conference on Design, automation and test in Europe*, p. 735, IEEE Press Piscataway, NJ, USA, 2001.

[Huan04] Zhining Huang, Sharad Malik, Nahri Moreano and Guido Araujo, "The design of dynamically reconfigurable datapath coprocessors", *ACM Trans. Embed. Comput. Syst.*, vol. 3, no. 2, pp. 361–384, 2004.

[Huan08] W. Huang, K. Sankaranarayanan, K. Skadron, R.J. Ribando and M.R. Stan, "Accurate, Pre-RTL Temperature-Aware Design Using a Parameterized, Geometric Thermal Model", *IEEE Transactions on Computers*, vol. 57, no. 9, pp. 1277–1288, 2008.

[Karg06] K. Kargupta, B.H. Park and H. Dutta, "Orthogonal decision trees", *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 8, pp. 1028–1042, 2006.

[Mukh06] R. Mukherjee, SO Memik, I. Synopsys and M. View, "An Integrated Approach to Thermal Management in High-Level Synthesis", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 11, pp. 1165–1174, 2006.

[Sank05] K. Sankaranarayanan, S. Velusamy, M. Stan and K. Skadron, "A case for thermal-aware floorplanning at the microarchitectural level", *Journal of Instruction-Level Parallelism*, vol. 7, pp. 1–16, 2005.

[Skad04] K. Skadron, M.R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy and D. Tarjan, "Temperature-aware microarchitecture: Modeling and implementation", *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 1, no. 1, pp. 94–125, 2004.

[Vass07] Stamatis Vassiliadis and Dimitrios Soudris, *Fine- and Coarse-Grain Reconfigurable Computing*, Springer Publishing Company, Incorporated, 2007.

[Wang07] Gang Wang, Wenrui Gong, Brian Derenzi and Ryan Kastner, "Exploring time/resource trade-offs by solving dual scheduling problems with the ant colony optimization", *ACM Trans. Des. Autom. Electron. Syst.*, vol. 12, no. 4, p. 46, 2007.