



Ανάπτυξη Μεθοδολογίας Σχεδιασμού Βέλτιστων Επεξεργαστών Ειδικού Σκοπού

ΠΕΝΕΔ-2003

Τίτλος:	Τεχνικές Γέννησης και Επιλογής Εντολών για Επεξεργαστές Ειδικού Σκοπού Συνόλου Εντολών
Συγγραφείς:	Νικόλαος Καβαβιάδας (Ερευνητής Α.Π.Θ.)
Κωδικός:	Π 4.1
Έκδοση:	0.1
Τύπος:	Παραδοτέο
Εμπιστευτικότητα:	Δημόσια
Ημερομηνία:	Ιούλιος 29, 2007
Πρόγραμμα:	Πρόγραμμα Ενίσχυσης του Ερευνητικού Δυναμικού (ΠΕΝΕΔ 2003)
Λέξεις-Κλειδιά:	επεξεργαστές ειδικού σκοπού, γέννηση εντολών, επιλογή εντολών, λειτουργικές μονάδες, εφαρμογές δοκιμής, εκτίμηση επιδόσεων, πρωτότυπα εργαλεία ανάπτυξης
Πρόλογος:	Αντικείμενο του παραδοτέου αυτού είναι η ανάπτυξη και υλοποίηση τεχνικών γέννησης και επιλογής ειδικών εντολών που μπορούν να αξιοποιηθούν στο πλαίσιο των στοχευόμενων επεξεργαστών ειδικού σκοπού (ΕΕΣ) συνόλου εντολών. Για το σκοπό αυτό επεκτέθηκαν τεχνικές της βιβλιογραφίας για την εξαγωγή ειδικών εντολών με οργάνωση γράφου τύπου MaxMISO και MIMO, οι οποίες τεχνικές εργάζονται στο επίπεδο ενδιάμεσης αναπαράστασης επαναστοχευόμενου μεταγωγτιστή. Οι αντίστοιχοι βασικοί αλγόριθμοι της βιβλιογραφίας ονομάζονται αλγόριθμος εξαγωγής εντολών MaxMISO και αλγόριθμος αναγνώρισης μονής τομής, αντίστοιχα. Οι τεχνικές αυτές υλοποιήθηκαν στα πρωτότυπα εργαλεία IAGF και YARDstick, τα οποία περιγράφονται αντίστοιχα στις ενότητες 3 και 4. Στη διαδικασία αυτή αξιοποιείται όλη η πρότερη υποδομή λογισμικού που αναπτύχθηκε από τον ερευνητή Α.Π.Θ. (κατά τις Φάσεις 2-3), για το χαρακτηρισμό των εφαρμογών δοκιμής για τους ΕΕΣ.

**Ιστορικό**

Ημερομηνία	Έκδοση	Σχόλια
Ιούλιος 17, 2007	0.1	Πρώτη πρόχειρη έκδοση
Ιούλιος 29, 2007	1.0	Επίσημη έκδοση

Πίνακας περιεχομένων

Πίνακας περιεχομένων	3
1. ΕΙΣΑΓΩΓΗ	4
2. ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ ΚΑΙ ΟΡΙΣΜΟΙ ΓΙΑ ΤΙΣ ΕΙΔΙΚΕΣ ΕΝΤΟΛΕΣ	5
2.1. Ορισμοί για τις βασικές δομές γράφων που χρησιμοποιούνται στη γένεση ειδικών εντολών	5
2.1.1. Βασικοί τύποι γράφων και τομών	5
2.1.2. Κατάλληλοι τύποι τομών για ειδικές εντολές	7
2.1.3. Θεμελιώδη προβλήματα γέννησης και επιλογής εντολών	8
2.2. Αλγόριθμοι γέννησης ειδικών εντολών	9
2.2.1. Βασικός αλγόριθμος εξαγωγής τομών MaxMISO	9
2.2.2. Βασικός αλγόριθμος εξαγωγής δομών MIMO (αναγνώριση μονής τομής)	10
3. ΓΕΝΝΗΣΗ ΚΑΙ ΕΠΙΛΟΓΗ ΕΝΤΟΛΩΝ ΜΕ ΤΟ ΠΛΑΙΣΙΟ ΕΡΓΑΣΙΑΣ IAGF	15
3.1. Γενική άποψη του IAGF	15
3.2. Δυνατότητες χρήσης του αλγόριθμου GENMISO εντός του IAGF	19
3.3. Αυτοματοποιημένα σενάρια διερεύνησης πεδίου λύσεων για την επέκταση ενσωματωμένων επεξεργαστών	21
3.3.1. Γέννηση εντολών για διαφορετικούς αριθμούς εισόδων	23
3.3.2. Επιτάχυνση εφαρμογής κάτω από διαφορετικούς περιορισμούς κόμβων	24
3.3.3. Επίδραση βελτιστοποιήσεων πολλαπλασιασμού απλής σταθεράς	27
3.3.4. Επιλογή ειδικών εντολών – Λύση με άπληστη τεχνική υπό περιορισμό αριθμού ειδικών εντολών και διαφορετικών μετρικών προτεραιότητας	27
4. ΓΕΝΝΗΣΗ ΚΑΙ ΕΠΙΛΟΓΗ ΕΝΤΟΛΩΝ ΜΕ ΤΟ YARDSTICK	32
4.1. Πρακτικά ζητήματα στη γέννηση/επιλογή ειδικών εντολών	32
4.2. Στα εσώτερα του YARDstick	34
4.2.1. Ο πυρήνας του YARDstick	34
4.2.2. Δομή ενός περιβάλλοντος YARDstick	37
4.2.3. Χρήση του YARDstick API	38
4.3. Περιπτώσεις διερεύνησης πεδίου λύσεων με το YARDstick	39
4.3.1. Εισαγωγικό παράδειγμα εξαγωγής ειδικής εντολής: Ανίχνευση ακμής με απλή παράγωγο	41
4.3.2. Αποτίμηση των επιδράσεων του μεταγωγτιστή: Περίπτωση μελέτης για πυρήνες εφαρμογών πολυμέσων	42
4.3.3. Περίπτωση μεταχηματισμού σε καταλληλότερο IR	46
4.3.4. Επίδραση του μοντέλου προσπελάσεων μνήμης δεδομένων	48
4.3.5. Άπληστη επιλογή ειδικών εντολών υπό μετρικών προτεραιότητας	50
4.4. Περιβάλλον χρήσης του YARDstick	52
5. ΓΛΩΣΣΑΡΙ ΟΡΩΝ	53
6. ΑΝΑΦΟΡΕΣ	56

1. ΕΙΣΑΓΩΓΗ

Στο πλαίσιο της Φάσης 4, ο ερευνητής Α.Π.Θ. (Νικόλαος Καββαδίας) επέκτεινε τεχνικές γέννησης και επιλογής ειδικών εντολών και γενικότερα ειδικών αρχιτεκτονικών επεκτάσεων υλικού (Application-Specific Hardware Extensions – ASHEs) της διεθνούς βιβλιογραφίας. Οι τεχνικές αυτές εργάζονται στο επίπεδο της ενδιάμεσης αναπαράστασης επαναστοχευόμενου μεταγλωττιστή. Η διαδικασία της γέννησης και επιλογής εντολών είναι ιδιαίτερης σημασίας σε μια ροή σχεδιασμού ΕΕΣ προκειμένου την επιλογή των πλεονεκτικότερων ASHEs οι οποίες ικανοποιούν συγκεκριμένα κριτήρια αναφορικά με τις επιδόσεις του επεξεργαστή (ταχύτητα επεξεργασίας, κατανάλωση ισχύος, κατάληψη επιφάνειας ολοκληρωμένου).

Συγκεκριμένα, ο ερευνητής Α.Π.Θ. (Νικόλαος Καββαδίας) επέκτεινε δύο διαφορετικές τεχνικές παραγωγής εντολών:

- Τον αλγόριθμο εξαγωγής εντολών MaxMISO [Poz00]. Οι υπογράφοι MaxMISO διαθέτουν ένα κόμβο εξόδου και αντιπροσωπεύουν εντολές με N_{in} ορίσματα ανάγνωσης (εισόδου) και ένα το πολύ όρισμα καταχωρητή εγγραφής (εξόδου).
- Τον αλγόριθμο αναγνώρισης μονής τομής (Single-Cut Identification – SCI) για την εξαγωγή εντολών MIMO [Poz06]. Οι υπογράφοι MIMO διαθέτουν N_{in} ορίσματα εισόδου και N_{out} ορίσματα εξόδου σε εντολές γενικής μορφής. Οι εντολές με δομή MaxMISO αποτελούν υποπερίπτωση των MIMO.

Οι υλοποιήσεις των αλγορίθμων από τον ερευνητή Α.Π.Θ., και στις δύο περιπτώσεις, προσθέτουν εκτενή παραμετροποίηση με τον ορισμό νέων περιορισμών χρήστη για την καλύτερη κατεύθυνση της διαδικασίας εξαγωγής εντολών ώστε την αποδοτικότερη κάλυψη και διερεύνηση του πεδίου λύσεων. Ιδιαίτερα για την τεχνική MIMO η οποία είναι εκθετικής υπολογιστικής πολυπλοκότητας, ο ερευνητής Α.Π.Θ. όρισε ευρεστική προϋπόθεση μονοτονικής συμπεριφοράς (περισσότερα στην ενότητα 4) για την σημαντική επιτάχυνση της διαδικασίας εξαγωγής εντολών. Στην συντριπτική πλειοψηφία των περιπτώσεων χρήσης του ευρεστικής τεχνικής, τα αποτελέσματα είναι ίδια με αυτά του βέλτιστου SCI.

Οι δύο τεχνικές εξαγωγής MaxMISO και MIMO εντολών ενσωματώθηκαν σε πρωτότυπα εργαλεία γέννησης ειδικών εντολών. Για τις εντολές MaxMISO συντάχθηκε το εργαλείο IAGF στις γλώσσες προγραμματισμού C/C++, το οποίο περιγράφεται εκτενώς στο κεφ. 3, και του οποίου η ανάπτυξη απαιτήσε περί τα 2 ανθρωποέτη (Σεπτ. 2004-Ιούνιος 2006). Για τις εντολές MIMO συντάχθηκε το εργαλείο YARDstick στις γλώσσες προγραμματισμού C/C++/Tcl-Tk, το οποίο περιγράφεται εκτενώς στο κεφ. 4, και του οποίου η ανάπτυξη έχει απαιτήσει μέχρι στιγμής (θα βρίσκεται σε διαρκή ανάπτυξη για αρκετό καιρό ακόμη) 10 ανθρωπομήνες εργασίας (Οκτ. 2006-Ιούλιος 2007).

Τα παραπάνω εργαλεία ενσωματώνουν και μηχανισμούς επιλογής των ειδικών εντολών με χρήστη άπληστης επιλογής (greedy selection) κάτω από συναρτήσεις προτεραιότητας (κόστους). Οι συναρτήσεις αυτές λαμβάνουν υπόψη κυρίως τους απαιτούμενους κύκλους εκτέλεσής τους για δεδομένο υλικό, τη χρονική διάρκεια του κύκλου, το μέγεθος του απαιτούμενου υλικού και την κατανάλωση ενέργειας.

Το εργαλείο YARDstick επίσης προσφέρει τη δυνατότητα εξαγωγής των CDFG των ειδικών εντολών προκειμένου την δρομολόγησή τους είτε χωρίς περιορισμούς είτε υπό περιορισμό πόρων υλικού (CDFG toolset [CDFGtool] και SALTO [SALTO]).

Επίσης, τα εργαλεία IAGF και YARDstick δεν είναι δεσμευμένα προς χρήση με ένα συγκεκριμένο μεταγλωττιστή. Το δε YARDstick είναι πλήρως επαναστοχευόμενο ως προς την ενδιάμεση αναπαράσταση μεταγλωττιστή προκειμένου να μπορούν να αξιοποιηθούν και τρίτοι μεταγλωττιστές μέσω εξωτερικής διεπαφής (ενδεικτικά αναφέρονται οι: GCC [GCC], COINS [COINS]).

2. ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ ΚΑΙ ΟΡΙΣΜΟΙ ΓΙΑ ΤΙΣ ΕΙΔΙΚΕΣ ΕΝΤΟΛΕΣ

2.1. Ορισμοί για τις βασικές δομές γράφων που χρησιμοποιούνται στη γένεση ειδικών εντολών

2.1.1. Βασικοί τύποι γράφων και τομών

Στη συνήθη πρακτική της εξαγωγής ειδικών εντολών, τα βασικά μπλοκ των συναρτήσεων των εφαρμογών (στο επίπεδο τοπικής οργάνωσης ενδιάμεσης αναπαράστασης μεταγλωττιστή) αναπαρίστανται υπό μορφή DAG. Ένα DAG καταγράφει τις εξαρτήσεις δεδομένων μεταξύ των στοιχειωδών λειτουργιών (βασικών εντολών) εντός ενός βασικού μπλοκ. Για αυτά τα DAG ισχύει ο Ορισμός 2.1.

Ορισμός 2.1 (DAG βασικού μπλοκ). Καλούνται $G(V,E)$ οι κατευθυνόμενοι άκυκλοι γράφοι (DAGs) που αναπαριστούν τη ροή (και τις εξαρτήσεις) δεδομένων σε κάθε βασικό μπλοκ: οι κόμβοι V αναπαριστούν στοιχειώδεις λειτουργίες και οι ακμές E αναπαριστούν εξαρτήσεις δεδομένων. Κάθε γράφος G συνδέεται με ένα γράφο $G^+(V \cup V^+, E \cup E^+)$, ο οποίος περιλαμβάνει επιπρόσθετους κόμβους V^+ και επιπρόσθετες ακμές E^+ . Οι επιπρόσθετοι κόμβοι V^+ αναπαριστούν μεταβλητές (έντελα/ορίσματα) εισόδου και εξόδου από το βασικό μπλοκ. Οι επιπρόσθετες ακμές E^+ συνδέουν κόμβους του V^+ με κόμβους του V , και κόμβους V με τους V^+ .

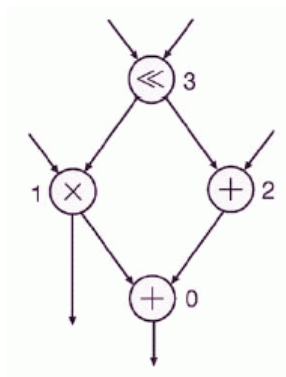
Μία υποψήφια ειδική εντολή (αναγνωρίσιμη εντός βασικού μπλοκ) αποτελεί τομή (cut) του γράφου G . Έτσι έχουμε:

Ορισμός 2.2 (Τομή του DAG βασικού μπλοκ). Μία τομή S είναι ένας παρακινούμενος (induced) υπογράφος του G : $S \subseteq G$.

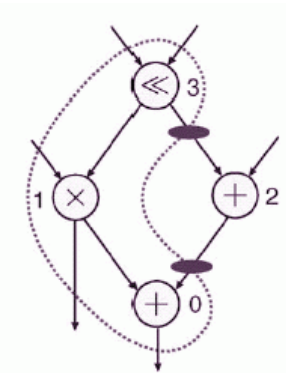
Υπάρχουν $2^{|V|}$ δυνατές τομές, όπου $|V|$ είναι ο αριθμός των κόμβων του G . Μία αυθαίρετη συνάρτηση $M(S)$ καταμετρά την αξία (merit) μιας τομής S . Η έκφραση $M(S)$ αποτελεί την αντικειμενική συνάρτηση του προβλήματος βελτιστοποίησης το οποίο στη γενική του εκδοχή αποτελεί το πρόβλημα γέννησης των βέλτιστων ειδικών εντολών υπό περιορισμούς. Η $M(S)$ αναπαριστά τυπικά μια εκτίμηση της επιτυγχανόμενης επιτάχυνσης εφαρμογής λόγω της υλοποίησης της τομής S από ειδική εντολή.

Ονομάζουμε $IN(S)$ τον αριθμό των προηγμένων κόμβων των οποίων οι ακμές εισέρχονται στην τομή S από το υπόλοιπο του γράφου G^+ . Αναπαριστούν τον αριθμό των μεταβλητών εισόδου που χρησιμοποιούνται (διαβάζονται) από τις λειτουργίες στην S . Παρόμοια, $OUT(S)$ είναι ο αριθμός των προηγμένων κόμβων στην S , των ακμών που εξέρχονται από την S . Αναπαριστούν τον αριθμό των τιμών που παράγονται στην S και χρησιμοποιούνται από άλλες λειτουργίες, είτε στον G είτε σε άλλα βασικά μπλοκ.

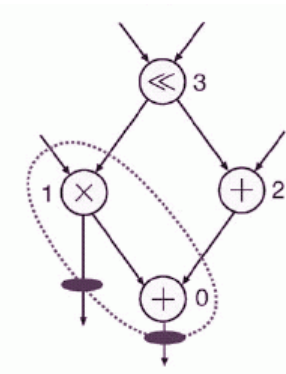
Ορισμός 2.3 (Κυρτότητα τομής). Μία τομή S καλείται κυρτή (convex) όταν δεν υφίσταται μονοπάτι από ένα κόμβο $u \in S$ προς άλλο κόμβο $v \in S$ το οποίο να περιλαμβάνει κόμβο $w \notin S$. Ένα παράδειγμα μη κυρτής τομής δίνεται στο Σχ. 2.1.β.



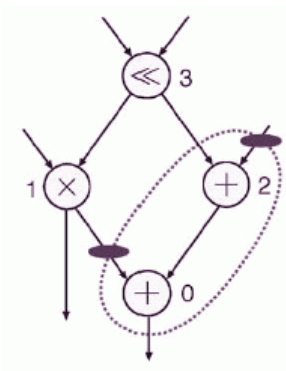
(α)



(β)



(γ)



(δ)

Σχήμα 2.1: (α) Τοπολογικά ταξινομημένος γράφος. (β) Παράδειγμα μη-κυρτής τομής. (γ) Μια τομή που παραβιάζει τον έλεγχο αριθμού εξόδων για $N_{out}=1$. (δ) Μία τομή που παραβιάζει

τον έλεγχο του αριθμού των μόνιμων εισόδων για $N_{in}=1$.

Επίσης, οι τιμές N_{in} , N_{out} εκφράζουν κάποια χαρακτηριστικά (ιδιότητες) της μικροαρχιτεκτονικής και πιο συγκεκριμένα δίνουν τον αριθμό θυρών ανάγνωσης και εγγραφής, αντίστοιχα, του αρχείου καταχωρητών (του υποτιθέμενου επεξεργαστή) οι οποίες μπορούν να χρησιμοποιηθούν από τις ειδικές εντολές.

Ακόμη, λόγω μικροαρχιτεκτονικών περιορισμών, υπάρχει η περίπτωση κάποιοι τύποι λειτουργιών να μην επιτρέπεται να συμπεριλαμβάνονται σε μια ειδική εντολή, πάντα σε εξάρτηση με την υποτιθέμενη οργάνωση της μικροαρχιτεκτονικής. Αυτό μπορεί για παράδειγμα να αντικατοπτρίζει την παρουσία ή απουσία θυρών μνήμης δεδομένων που να έχουν άμεση συνδεσιμότητα με τις λειτουργικές μονάδες. Σε τέτοια περίπτωση θα πρέπει οι εντολές φόρτωσης και αποθήκευσης (`lod` και `str`, αντίστοιχα, για το SUIFvm IR) να αποκλείονται από την συμπερίληψή τους σε πιθανές τομές. Ονομάζουμε F (με $F \subseteq V$) το σύνολο των απαγορευμένων κόμβων οι οποίοι δεν πρέπει να αποτελέσουν σε καμία περίπτωση μέρος της τομής S .

2.1.2. Κατάλληλοι τύποι τομών για ειδικές εντολές

Με βάση τους παραπάνω ορισμούς, μπορούμε να ορίσουμε τους κατάλληλους τύπους τομών γράφων που δύνανται να αντιπροσωπεύουν τους γράφους εξάρτησης δεδομένων (γενικά και ελέγχου) ειδικών εντολών. Οι παρακάτω ορισμοί περιορίζονται σε ειδικές εντολές αναγνωρίσιμες εντός βασικών μπλοκ.

Η κατηγοριοποίηση μιας τομής προσδιορίζεται από τις δυνατές τιμές των χαρακτηριστικών $IN(S)$, $OUT(S)$ για τις μεταβλητές εισόδου και εξόδου. Έτσι μπορούμε να διακρίνουμε τις εξής τέσσερις κατηγορίες:

- τομές MISO ($N_{out}=1$)
- τομές MIMO
- μέγιστες τομές MISO (Ορ. 2.5) ή MaxMISO
- μέγιστες τομές MIMO ή MaxMIMO

Ακολουθεί ο επίσημος ορισμός για τους υπογράφους MISO:

Ορισμός 2.4 (Τομές MISO). Έστω ο υπογράφος $G^i = \langle V^i, E^i \rangle$ όπου V^i είναι το σύνολο των κόμβων στον G^i και E^i το σύνολο όλων των ακμών που εξέρχονται από αυτούς τους κόμβους. Μία ακμή $e^i \in E^i$ αναγνωρίζεται από τον κόμβο πηγής της ($v_k^i \in V^i$) και τον κόμβο προορισμού v_l^i και σημειώνεται με $e^i(k,l)$. Αν για όλους τους $v_k^i \in V^i$ με εξαίρεση τον ένα κόμβο v_o^i είναι αληθές ότι:

$$\forall e^i(k,l) \in E^i, v_l^i \in V^i$$

τότε ο G αποτελεί MISO. Σημειώνουμε έναν MISO ως M^i με τον v_o^i να είναι ο κόμβος εξόδου του. Σημειώνεται επίσης ότι κάθε κόμβος του V^i μπορεί να έχει εισερχόμενες ακμές που προέρχονται από κόμβους που δεν ανήκουν στον V^i .

Από τον ορισμό του υπογράφου MISO (Ορ. 2.4) προκύπτει άμεσα ο ορισμός του υπογράφου MaxMISO:

Ορισμός 2.5 (Τομές MaxMISO). Ένας υπογράφος MaxMISO MM^i είναι ένας MISO ο οποίος

δεν περιλαμβάνεται πλήρως σε οποιονδήποτε άλλον MISO.

Το Θεώρημα 2.1 αφορά το αδύνατο της μερικής ή ολικής αλληλεπικάλυψης δύο υπογράφων MaxMISO και έχει αποδειχτεί στη βιβλιογραφία [Ρoz00].

Θεώρημα 2.1. Δύο MaxMISO MM^i , MM^j δεν μπορεί να αλληλεπικαλύπτονται μερικώς (άρα και ολικώς).

Ο επίσημος ορισμός για ένα γράφο MIMO είναι απλούστερος λόγω της έλλειψης περιοριστικών συνθηκών. Ένας γράφος MIMO ορίζεται κατά τον Ορ. 2.2 ως κυρτή τομή S του γράφου G (DDG DAG ενός βασικού μπλοκ) και χαρακτηρίζεται από τις ποσότητες $IN(S)$ και $OUT(S)$ οι οποίες μπορούν να είναι και μεγαλύτερες του 1.

Πρέπει εδώ να σημειωθεί ότι πέραν του μέγιστου αριθμού εξόδων, οι γράφοι MISO και MIMO διαφέρουν ως προς το γεγονός ότι ένας MIMO μπορεί να ασύνδετος (disjoint) ενώ οι MISO είναι συνδεδεμένοι (αποδεικνύεται ως πόρισμα).

Παρόμοια με την τομή MaxMISO ορίζεται και τομές MaxMIMO:

Ορισμός 2.6 (Τομές MaxMIMO). Ένας υπογράφος MaxMIMO MMM^i είναι ένας MIMO ο οποίος δεν περιλαμβάνεται πλήρως σε οποιονδήποτε άλλον MIMO.

2.1.3. Θεμελιώδη προβλήματα γέννησης και επιλογής εντολών

Θεωρώντας κάθε βασικό μπλοκ ξεχωριστά, το πρόβλημα της αναγνώρισης μιας τομής επιλέξιμης για υλοποίηση ως ειδική εντολή δηλώνεται ως εξής:

Ορισμός 2.6 (Πρόβλημα της αναγνώρισης μονής τομής). Δοθέντος ενός γράφου G^+ και των μικροαρχιτεκτονικών χαρακτηριστικών N_{in} , N_{out} και του συνόλου F , ζητείται η ανεύρεση της τομής S η οποία μεγιστοποιεί την συνάρτηση αξίας $M(S)$ κάτω από τους παρακάτω περιορισμούς:

- 1) $IN(S) \leq N_{in}$
- 2) $OUT(S) \leq N_{out}$
- 3) $F \cap S = \emptyset$
- 4) Η S είναι κυρτή.

Ο περιορισμός κυρτότητας (4) προσφέρει έλεγχο εγκυρότητας της τομής S και χρειάζεται για την διασφάλιση του εφικτού της δρομολόγησης των λειτουργιών της S : όπως δείχνει το Σχ. 2.1.γ, αν όλες οι εισοδοί μιας ειδικής εντολής υποτίθεται ότι πρέπει να είναι διαθέσιμες κατά το χρόνο έκδοσης και όλα τα αποτελέσματα παράγονται στο τέλος της εκτέλεσης της ειδικής εντολής, δεν υπάρχει εφικτή δρομολόγηση η οποία να μπορεί να σεβαστεί τις εξαρτήσεις του γράφου αυτού εφόσον η S καταρρεύσει σε ένα κόμβο (σύνθετης) εντολής.

Το πρόβλημα της επιλογής των τελικών ειδικών εντολών από το σύνολο των υποψηφίων ειδικών εντολών ορίζεται ως εξής:

Ορισμός 2.7 (Πρόβλημα της επιλογής ειδικών εντολών). Δοθέντος των γράφων G_i^+ όλων των βασικών μπλοκ και των μικροαρχιτεκτονικών χαρακτηριστικών N_{in} , N_{out} , και F , ζητείται η εύρεση έως N_{instr} τομών S_j ώστε να μεγιστοποιείται το $\sum_j M(S)_j$ κάτω από τους ίδιους περιορισμούς με τον Ορ. 2.6 για κάθε τομή S_j .

2.2. Αλγόριθμοι γέννησης ειδικών εντολών

2.2.1. Βασικός αλγόριθμος εξαγωγής τομών MaxMISO

Ο βασικός αλγόριθμος για την εξαγωγή μέγιστων MISO υπογράφων (MaxMISO) εργάζεται σε κατευθυντικούς άκυκλους γράφους (DAG). Κατά τη συνήθη πρακτική της εξαγωγής ειδικών εντολών, ο αλγόριθμος εφαρμόζεται σε IR μεταγλωττιστή η οποία είναι τοπικά οργανώσιμη υπό μορφή DAG. Ένα DAG αναπαριστά τις εξαρτήσεις δεδομένων εντός ενός βασικού μπλοκ. Στο επίπεδο αυτού του DAG δεν υφίσταται θεώρηση καταχωρητών και δρομολόγησης σε βήματα ελέγχου, για αυτό μπορεί να θεωρηθεί ως μια μορφή γράφου ροής δεδομένων (DFG).

```
∀Node ∈ Nodes_to_be_analysed do
{
  Generate_MAXMISO(Node)
  Nodes_to_be_analysed -- = Nodes_in_MAXMISO
}

Generate_MAXMISO(Node)
{
  ∀Parent_of_Node do
  {
    if (fanout == 1) {
      include(Parent_of_Node)
      Generate_MAXMISO(Parent_of_Node)
    }else
      fanout --
  }
}
```

Σχήμα 2.2: Ψευδοκώδικας του αλγορίθμου για την εξαγωγή όλων των MaxMISO από ένα DAG.

Το Σχ. 2.2 απεικονίζει τον ψευδοκώδικα για τη γέννηση όλων των MaxMISO εντός ενός DAG βασικού μπλοκ. Ο αλγόριθμος εργάζεται σε δύο βήματα: πρώτα, επιλέγεται ένας κόμβος ώστε να είναι ο κόμβος εξόδου, και έπειτα το πρόγραμμα ενεργοποιεί μια συνάρτηση η οποία χτίζει την MaxMISO που σχετίζεται (για την ακρίβεια έχει ως ρίζα) με τον κόμβο εξόδου. Οι κόμβοι εξόδου επιλέγονται από κάτω προς τα πάνω, δηλαδή εκκινώντας από τις εξόδους (κόμβοι χωρίς διάδοχους) του DAG. Αρχικά, το σύνολο των Nodes_to_be_analysed συμπίπτει με το σύνολο των κόμβων του DAG· εν συνεχεία, όταν παράγεται μία MaxMISO, οι συνιστώντες κόμβοι της απομακρύνονται από το σύνολο Nodes_to_be_analysed. Η συνάρτηση Generate_MAXMISO αρχίζει από τον επιλεγμένο κόμβο εξόδου και αναδρομικά προσπαθεί να συμπεριλάβει τους γονείς του στην MaxMISO υπό γέννηση. Η αναδρομή ολοκληρώνεται όταν ο κόμβος που απαντάται δεν είναι έγκυρος (π.χ. είναι μια εντολή φόρτωσης που είναι απαγορευμένη) ή δεν έχει επανασυγκλίνον φορτίο εξόδου (fan-out).

Προκειμένου την βελτιστοποίηση του κώδικα που υλοποιεί τον αλγόριθμο γέννησης MaxMISO, η αναγνώριση μιας περίπτωσης στην οποία συναντάται ένα μη-επανασυγκλίνον φορτίο εξόδου έχει υλοποιηθεί ως εξής: Κάθε φορά που η συνάρτηση επισκέπτεται ένα

κόμβο που έχει φορτίο εξόδου μεγαλύτερο του ενός, η τιμή της μεταβλητής fanout μειώνεται κατά 1. Τώρα, αν και μόνο αν η συνάρτηση επισκέπτεται τον κόμβο fanout φορές, (και έτσι μία κλήση συνάρτησης επισκέπτεται τον κόμβο όταν η τιμή της fanout έχει μειωθεί στο ένα) ο κόμβος συμπεριλαμβάνεται στην MaxMISO (καθώς αυτό υπονοεί ότι επιτυγχάνεται σύγκλιση). Όταν ο κόμβος έχει αριθμό επισκέψεων μικρότερο του fanout, τότε μπορεί μόνο να είναι κόμβος εξόδου κάποιας άλλης MaxMISO, και δεν μπορεί να συμπεριληφθεί στην τρέχουσα. Ο αλγόριθμος δείχνει μια υπολογιστική πολυπλοκότητα που είναι γραμμική με τον αριθμό των κόμβων εντολών στο εξετασθέν DAG. Αυτό προκύπτει από το Θεωρ. 2.1: καθώς οι MaxMISO δεν μπορούν να αλληλεπικαλύπτονται, κάθε κόμβος διαβαίνεται ως υποψήφιος κόμβος εξόδου μιας MaxMISO μόνο μία φορά.

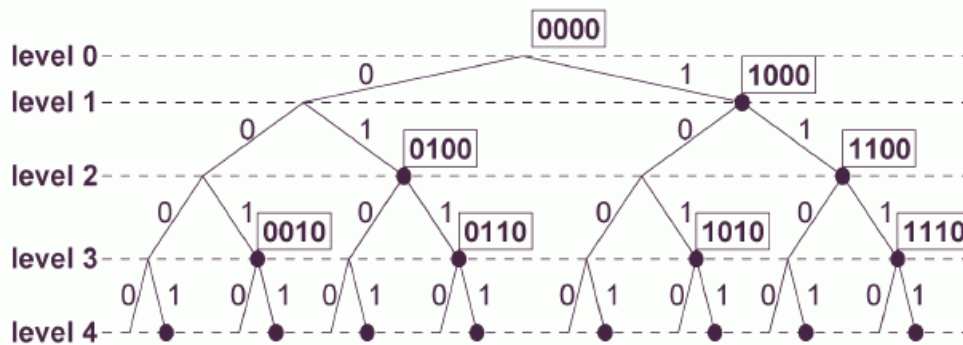
2.2.2. Βασικός αλγόριθμος εξαγωγής δομών MIMO (αναγνώριση μονής τομής)

Ο βασικός αλγόριθμος αναγνώρισης MIMO (ήτοι πολλαπλών εισόδων-πολλαπλών εξόδων) τομών [Poz06] ονομάζεται αλγόριθμος αναγνώρισης μονής τομής (single-cut identification). Σε μία εφαρμογή του αλγορίθμου, είναι δυνατό να αναγνωρισθεί μέχρι μία MIMO ειδική εντολή. Ολοκληρώνοντας την τρέχουσα επανάληψη της διαδικασίας αναγνώρισης, η MIMO εντολή που τυχόν αναγνωρίστηκε στο DAG του βασικού μπλοκ υπό εξέταση, αντικαθίσταται με σύνθετο κόμβο και η διαδικασία μπορεί να επαναληφθεί, θεωρώντας τον σύνθετο κόμβο ως μη επισκέψιμο (απαγορευμένο).

Η διεξοδική απαρίθμηση όλων των πιθανών τομών εντός ενός βασικού μπλοκ δεν είναι υπολογιστικά εφικτή (ιδιαίτερα για μεγάλα βασικά μπλοκ) καθώς η αντίστοιχη υπολογιστική πολυπλοκότητα είναι εκθετική με τον αριθμό των κόμβων του DAG. Στο σημείο αυτό περιγράφεται ένας ακριβής (exact) αλγόριθμος ο οποίος εξερευνά το πεδίο αναζήτησης σε όλη την έκτασή του μεν αλλά ανιχνεύει με επάρκεια και απορρίπτει υποπεριοχές του πεδίου αναζήτησης οι οποίες αποδεδειγμένα δεν οδηγούν σε εφικτές λύσεις. Ο αλγόριθμος λύνει ακριβώς το πρόβλημα που τίθεται από τον Ορ. 2.6 (Πρόβλημα της αναγνώρισης μονής τομής).

Ο αλγόριθμος εκκινά με τοπολογική ταξινόμηση πάνω στον γράφο G : Οι κόμβοι του G ταξινομούνται έτσι ώστε αν ο G περιέχει μία ακμή (u, v) τότε ο u εμφανίζεται μετά τον v στην κατάταξη. Η σχέση αυτή μεταξύ των κόμβων γράφεται ως $u > v$. Το Σχ. 2.1.α δείχνει ένα τέτοιο τοπολογικά ταξινομημένο γράφο. Ο αλγόριθμος χρησιμοποιεί μία αναδρομική συνάρτηση αναζήτησης βασισμένη σε αυτή την κατάταξη προκειμένου τη εξερεύνηση ενός αφηρημένου δένδρου αναζήτησης.

Το δένδρο αναζήτησης είναι ένα δυαδικό δένδρο με κόμβους που αναπαριστούν πιθανές τομές. Το δένδρο δομείται ξεκινώντας από ένα κόμβο-ρίζα που αναπαριστά την κενή τομή, και κάθε ζεύγος κλάδων που σημειώνεται με 1 και 0 στο επίπεδο i αναπαριστά την προσθήκη ή όχι του κόμβου στον G που έχει τοπολογική κατάταξη i , στην (τρέχουσα) τομή που αντιπροσωπεύεται από τον γονεϊκό κόμβο. Οι κόμβοι του δένδρου αναζήτησης που ακολουθούν άμεσα μιας διακλάδωσης με μηδενική τιμή αναπαριστούν την ίδια τομή με τον άμεσο γονιό τους και μπορούν να αγνοηθούν στην διαδικασία της αναζήτησης. Το Σχ. 2.3 απεικονίζει το δένδρο αναζήτησης για τον γράφο G του Σχ. 2.1.α με κάποιους κόμβους να φέρουν επιγραφή με τις τιμές τομής τους. Η αναζήτηση προχωρά ως διάβαση προκατάταξης (preorder traversal) του δένδρου αναζήτησης. Μπορεί να δειχθεί ότι σε ορισμένες περιπτώσεις δεν υπάρχει ανάγκη διακλάδωσης προς τα χαμηλότερα επίπεδα, και έτσι επιτυγχάνεται η σύντμηση του πεδίου αναζήτησης.



Σχήμα 2.3: Δένδρο αναζήτησης τομών για τον γράφο του Σχήματος 2.1.α.

Μία τετριμμένη περίπτωση, για την οποία είναι περιττή η εξερεύνηση του υποδένδρου που εκπορεύεται από ένα συγκεκριμένο κόμβο του δένδρου αναζήτησης, προκύπτει όταν ένας τέτοιος κόμβος αναπαριστά μία τομή S , η οποία περιέχει ένα απαγορευμένο κόμβο (κόμβος που ανήκει στο σύνολο F). Είναι προφανές ότι στην περίπτωση αυτή η S δεν μπορεί να αποτελεί λύση στο πρόβλημα του Ορ. 2.6 αλλά ούτε μπορεί να υπάρξει οποιαδήποτε νέα λύση η οποία συντίθεται με περαιτέρω προσθήκη κόμβων.

Η πραγματική χρησιμότητα αυτής της διάταξης διάβασης μπορεί να γίνει κατανοητή από την ακόλουθη συζήτηση. Ας υποθεθεί για παράδειγμα ότι ο περιορισμός ορισμάτων εξόδου έχει ήδη παραβιαστεί από την τομή που καθορίζεται από ένα συγκεκριμένο κόμβο δένδρου. Στην περίπτωση αυτή η πρόσθεση κόμβων οι οποίοι εμφανίζονται σε επόμενες θέσεις της τοπολογικής κατάταξης δεν μπορούν να ελαττώσουν τον αριθμό των εξόδων από την τομή. Ένα παράδειγμα αυτού δίνεται στο Σχ. 2.1.γ όπου ένας περιορισμός ορισμάτων εξόδου ($N_o=1$) παραβιάζεται με την συμπερίληψη του κόμβου με αύξοντα αριθμό 1 και έτσι η ικανοποίηση του περιορισμού αυτού για την τομή δεν μπορεί να ανακτηθεί. Παρόμοια, αν ο περιορισμός κυρτότητας παραβιάζεται σε ένα συγκεκριμένο κόμβο δένδρου, δεν υπάρχει τρόπος ώστε να ανακτηθεί η ιδιότητα της κυρτότητας με την εισαγωγή επιπλέον κόμβων του G οι οποίοι εμφανίζονται σε επόμενες θέσεις της τοπολογικής κατάταξης. Αν θεωρήσουμε τον γράφο του Σχ. 2.1.β μετά τη συμπερίληψη του κόμβου 3, οι μόνοι τρόποι για την ανάκτηση της κυρτότητας είναι είτε να συμπεριληφθεί και ο κόμβος 2 είτε να αφαιρεθούν οι κόμβοι 0 ή 3 από την τομή. Εξαιτίας της χρήσης της τοπολογικής ταξινόμησης, και οι δύο λύσεις είναι ανέφικτες σε ένα βήμα της διαδικασίας αναζήτησης μετά την εισαγωγή του κόμβου 3. Ως συνέπεια αυτού, όταν οι περιορισμοί ορισμάτων εξόδου ή κυρτότητας παραβιάζονται κατά τη διάβαση από ένα συγκεκριμένο κόμβο του δένδρου αναζήτησης, το υποδένδρο που έχει ρίζα τον κόμβο αυτό μπορεί να μην λαμβάνεται υπ' όψη στο πεδίο αναζήτησης.

Η παραπάνω συζήτηση για τις επιμέρους ιδιότητες των έγκυρων τομών που προκύπτουν με τη διαδικασία αναζήτησης μπορεί να τυποποιηθεί με δύο απλά θεωρήματα (πλήρεις αποδείξεις βρίσκονται στο [Poz06]) τα οποία δικαιολογούν τη χρήση της προαναφερθείσας διάταξης διάβασης.

Δοθέντων δύο τομών S_1 και S_2 του G , σημειώνουμε ως $S_1 \cup S_2$ τον παρακινούμενο γράφο του G , ο οποίος περιέχει όλους τους κόμβους των S_1 και S_2 .

Θεώρημα 2.2 (Μονοτονικότητα του αριθμού ορισμάτων εξόδου). Έστω S_1 και S_2 δύο διαχωρισμένες τομές του G έτσι ώστε για κάθε κόμβο $u_1 \in S_1$ και $u_2 \in S_2$ να είναι $u_2 \succ u_1$. Τότε είναι: $OUT(S_1 + S_2) \geq OUT(S_1)$.

Θεώρημα 2.3 (Μονοτονικότητα της κυρτότητας). Έστω S_1 και S_2 δύο διαχωρισμένες τομές του G έτσι ώστε για κάθε κόμβο $u_1 \in S_1$ και $u_2 \in S_2$ να είναι $u_2 \succ u_1$. Αν η τομή S_1 είναι

μη-κυρτή (non-convex), τότε και η $S_1 \cup S_2$ είναι μη-κυρτή.

Επιπρόσθετα, η έκταση του πεδίου αναζήτησης μειώνεται όταν υφίστανται παραβιάσεις του περιορισμού ορισμάτων εισόδου. Για αυτό σημειώνουμε ότι υπάρχουν περιπτώσεις που οι ακμές που εισέρχονται σε μία τομή S από τον γράφο G^+ δεν μπορούν να αφαιρεθούν από το σύνολο $IN(S)$ με περαιτέρω προσθήκη κόμβων τομής του G οι οποίοι εμφανίζονται αργότερα στην τοπολογική κατάταξη. Ο αριθμός των προηγμένων κόμβων των ακμών αυτών που εισέρχονται στην S από το υπόλοιπο του G^+ ονομάζεται $IN_f(S)$ και: 1) ανήκουν στο V^+ ή στο F (δηλαδή είτε προέρχονται από πρωτογενείς μεταβλητές εισόδου είτε από απαγορευμένους κόμβους) ή 2) είναι κόμβοι που έχουν ήδη θεωρηθεί κατά τη διάβαση δένδρου και τους έχει ανατεθεί μηδενικό στην τομή, δηλαδή έχουν εξαιρεθεί από την συμπερίληψή τους στην τομή. Στην πρώτη περίπτωση, δεν υπάρχει κόμβος στο υπόλοιπο του G ο οποίος μπορεί να απομακρύνει αυτή την τιμή εισόδου. Στην δεύτερη περίπτωση, κόμβοι οι οποίοι θα μπορούσαν να απομακρύνουν τις εισόδους υπάρχουν, αλλά έχουν ήδη αφαιρεθεί από την τομή. Προφανώς, ισχύει πάντα ότι $IN_f(S) \leq IN(S)$ και οι εισοδοί αυτές ονομάζονται μόνιμες (permanent) εισοδοί.

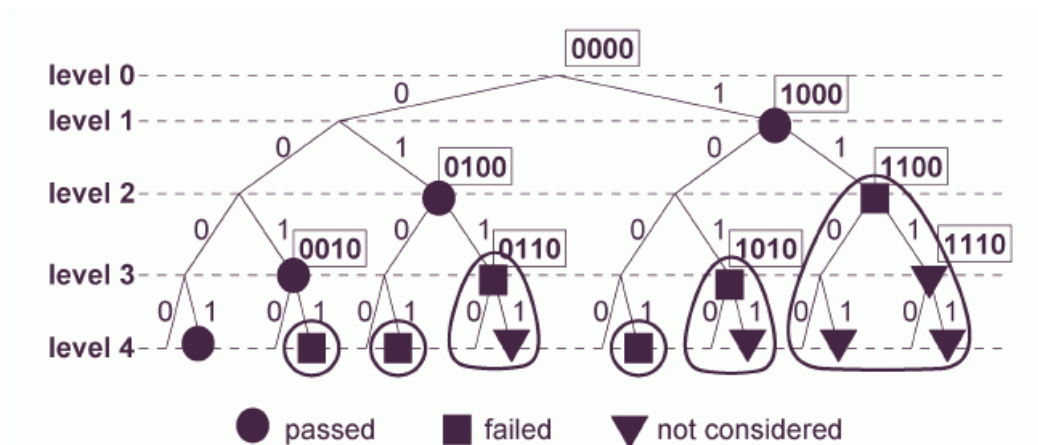
Ως παράδειγμα, θεωρούμε τον γράφο του Σχ. 2.1.δ. Για τον γράφο αυτό με την συμπερίληψη του κόμβου 2, η τομή έχει συσσωρεύσει δύο μόνιμες εισόδους: από αυτές η μία είναι η εξωτερική είσοδος του κόμβου 2 και η άλλη είναι η είσοδος του κόμβου 0, η οποία μετατράπηκε σε μόνιμη όταν ο κόμβος 1 εξαιρέθηκε από την τομή. Παρόμοια με ότι συμβαίνει για τους περιορισμούς θυρών εξόδου και κυρτότητας, όταν ο αριθμός των μόνιμων εισόδων $IN_f(S)$ παραβιάζει τον περιορισμό θυρών εισόδου όταν απαντάται (κατά τη διάβαση) ένας συγκεκριμένος κόμβος στο δένδρο αναζήτησης, το υποδένδρο που έχει αυτόν τον κόμβο ως ρίζο μπορεί να αφαιρεθεί από το πεδίο αναζήτησης.

Το Σχ. 2.4 δίνει τον αλγόριθμο της αναγνώρισης μονής τομής σε ψευδοκώδικα παρόμοιο με την C. Το δένδρο αναζήτησης κατασκευάζεται έμμεσα με την χρήση της αναδρομικής συνάρτησης search. Η παράμετρος current_choice ορίζει την κατεύθυνση της διακλάδωσης, και η παράμετρος current_index ορίζει τον δείκτη του κόμβου γράφου και το επίπεδο του δένδρου στο οποίο λαμβάνεται η διακλάδωση. Η συνάρτηση forbidden επιστρέφει ΑΛΗΘΕΣ όταν η S περιέχει τουλάχιστον ένα κόμβο του F . Οι συναρτήσεις input_port_check, permanent_input_port_check, και output_port_check επιστρέφουν αληθή τιμή όταν, αντίστοιχα, $IN(S) \leq N_{in}$, $IN_f(S) \leq N_{in}$, και $OUT(S) \leq N_{out}$. Η συνάρτηση convexity_check επιστρέφει αληθή τιμή όταν η S είναι κυρτή. Όταν αποτυγχάνει κάποιος από τους ελέγχους εγκυρότητας (αριθμού εξόδων, αριθμού μόνιμων εισόδων και κυρτότητας), ο αλγόριθμος οπισθοδρομεί (backtracking). Η βέλτιστη λύση ενημερώνεται μόνο όταν όλοι οι περιορισμοί έχουν ικανοποιηθεί για την τρέχουσα τομή.

```
identification() {  
    for (i = 0; i < NODES; i++) cut[i] = 0;  
    topological_sort();  
    search(1, 0);  
    search(0, 0);  
}  
  
search(current_choice, current_index) {  
    cut[current_index] = current_choice;  
    if (current_choice == 1) {  
        if (forbidden()) return;  
        if (!output_port_check()) return;  
        if (!permanent_input_port_check()) return;  
        if (!convexity_check()) return;  
        if (input_port_check()) {  
            calculate_speedup();  
            update_best_solution();  
        }  
    }  
    if ((current_index + 1) == NODES) return;  
    current_index = current_index + 1;  
    search(1, current_index);  
    search(0, current_index);  
}
```

Σχήμα 2.4: Ο αλγόριθμος εξαγωγής MIMO εντολών με αναγνώριση μονής τομής.

Το Σχ. 2.5 απεικονίζει την εφαρμογή του αλγορίθμου στον γράφο που δίνεται στο Σχ. 2.1.α για $N_{out}=1$ και $N_{in}=1$. Μόνο τέσσερις τομές περνούν επιτυχώς όλους τους ελέγχους περιορισμών, ενώ έξι τομές βρίσκεται να παραβιάζουν τουλάχιστον ένα περιορισμό, κάτι που έχει ως αποτέλεσμα την εξαίρεση πέντε τομών επιπλέον. Ανάμεσα στις 16 πιθανές τομές, μόνο 10 από αυτές χρειάζεται να εξεταστούν από τον αλγόριθμο (η κενή τομή δεν εξετάζεται).



Σχήμα 2.5: Ίχνος εκτέλεσης του αλγορίθμου αναγνώρισης μονής τομής για τον γράφο του Σχ. 2.1.α, για $N_{out}=1$ και $N_{in}=1$. Να σημειωθεί ότι η τομή 1100 που αντιστοιχεί στον υπογράφο που απεικονίζεται στο Σχ. 2.1.γ παραβιάζει τον περιορισμό εξόδου, και η τομή 1010 που αντιστοιχεί στον υπογράφο του Σχ. 2.1.δ αποτυγχάνει στον έλεγχο αριθμού μόνιμων εισόδων.

Οι κόμβοι γράφων περιέχουν $O(1)$ καταχωρήσεις στην λίστα γεινιάσής τους κατά μέσο όρο, καθώς ο αριθμός των εισόδων ενός κόμβου γράφου είναι περιορισμένος σε κάθε πρακτική περίπτωση (εξαίρεση αυτού αποτελούν οι βαριαδικοί κόμβοι *cal* και *mbr* του SUIF_{vm} IR που είναι μεταβλητού αριθμού έντελων εισόδου και πιθανόν αυτό συμβαίνει αντίστοιχα και για άλλα IR μεταγλωττιστή). Οι συναρτήσεις *input_port_check*, *permanent_input_port_check*,



output_port_check, convexity_check και calculate_speedup, σε συνδυασμό με την εισαγωγή ενός κόμβου ανά βήμα του αλγορίθμου μπορούν να υλοποιηθούν σε χρόνο $O(1)$ χρησιμοποιώντας κατάλληλες δομές δεδομένων. Η συνολική πολυπλοκότητα του αλγορίθμου είναι της τάξης του $O(2^M)$ όμως παρ' όλο που είναι εκθετική στη θεωρία, στην πράξη για πολλές εφαρμογές δοκιμής [Poz06] επιδεικνύει ασυμπτωτικά πολυωνυμική πολυπλοκότητα.

3. ΓΕΝΝΗΣΗ ΚΑΙ ΕΠΙΛΟΓΗ ΕΝΤΟΛΩΝ ΜΕ ΤΟ ΠΛΑΙΣΙΟ ΕΡΓΑΣΙΑΣ IAGF

3.1. Γενική άποψη του IAGF

Αποτελεί συχνό φαινόμενο στα πρώτα στάδια του σχεδιασμού ενός επεξεργαστή, να μην είναι διαθέσιμα τα κατάλληλα εργαλεία μεταγλώττισης και προσομοίωσης για την διερεύνηση του πεδίου λύσεων που απαρτίζεται από τις εφαρμόσιμες (μικρο-)αρχιτεκτονικές για τον θεωρούμενο επεξεργαστή. Ως κοινή αρχιτεκτονική πλατφόρμα, ώστε τα αποτελέσματα χαρακτηρισμού εφαρμογών να είναι χρήσιμα στο φάσμα των πιθανών μικροαρχιτεκτονικών, μπορεί να χρησιμοποιηθεί η ενδιάμεση αναπαράσταση χαμηλού επιπέδου (low-level IR) ενός μεταγλωττιστή. Τέτοιου είδους IR αναπαριστούν πληροφορία εξαρτήσεων δεδομένων και ελέγχου σε γενικευμένους/συμβολικούς RISC επεξεργαστές ώστε να μην είναι πολωμένοι ως προς συγκεκριμένους υπάρχοντες επεξεργαστές. Ανάμεσα στα διάφορα υπολογιστικά μοντέλα (computational models) που χρησιμοποιούνται σε μεταγλωττιστές και εργαλεία HLS για την αποτύπωση της συμπεριφοράς της εφαρμογής σε ενδιάμεση μορφή για επεξεργασία, κατάλληλες κρίνονται οι αναπαραστάσεις τύπου CDFG. Παραδείγματα IR μεταγλωττιστή που είναι οργανώσιμο σε μορφές CDFG έχουν ήδη αναφερθεί (SSA [Cyt91], SSI [Sin04], Π2.1-κεφ. 2) ενώ αντίστοιχα είναι τα IR που χρησιμοποιούνται στα γνωστά εργαλεία HLS όπως το SPARK [SPARK] (HCDFG ιεραρχικής δομής). Η δομή CDFG επιτρέπει την μετάφραση επιμέρους δομών της εφαρμογής για τη σημασιολογία της σύνθεσης υψηλού επιπέδου για επιταχυντές υλικού καθώς και για τη γέννηση κώδικα για προγραμματιζόμενους επεξεργαστές.

Το πλαίσιο εργασίας IAGF (Instruction/AFU Generation Framework) χρησιμοποιεί CDFG IR και παρουσιάζεται στο Σχ. 3.1. Αν και θα μπορούσαν να χρησιμοποιηθούν διαφορετικά IR (κάποιες δοκιμές έγιναν για το three-address-code C IR του LANCE [LANCE] και νέο IR του μεταγλωττιστή GCC ονόματι GIMPLE [GCC] ενώ κάτι τέτοιο θα ήταν εφικτό και για τον LLVM [LLVM]), στην πράξη το IAGF χρησιμοποιείται για το SUIF_{vm} IR της υποδομής μεταγλωττιστή SUIF/Machine-SUIF. Η ροή γέννησης εντολών του σχήματος χρησιμοποιεί μία βελτιωμένη έκδοση του περιβάλλοντος χαρακτηρισμού εφαρμογών που έχει παρουσιαστεί σε διεθνές συνέδριο με κριτές [Kan04]. Πρώτα, γίνεται επεξεργασία του C/C++ κώδικα της εφαρμογής από το SUIF frontend, διαδικασία που συμπεριλαμβάνει τη κατασκευή του AST με κόμβους SUIF, και μετασχηματισμούς του SUIF για την αποδόμηση εκφράσεων της C/C++ σε απλούστερες. Έπειτα, η εξαγόμενη αναπαράσταση εισάγεται στο πέρασμα s2m για την εκπομπή SUIF_{vm} IR. Ο κώδικας στο επίπεδο του IR δεν έχει υποστεί δρομολόγηση ενώ δεν περιλαμβάνει πλήρεις ακολουθίες εντολών για την είσοδο και έξοδο από ρουτίνες καθώς ο κώδικας για την διάταξη πλαισίου στοίβας (stack frame layout) εξαρτάται από τον τελικά στοχευόμενο επεξεργαστή. Γενικά θεωρείται ότι δεν είναι σκόπιμη η αναζήτηση υποψηφίων ειδικών εντολών σε ακολουθίες κώδικα χειρισμού στοίβας, λόγω των λανθανόντων εξαρτήσεων δεδομένων που ανακύπτουν στους αντίστοιχους γράφους σε επίπεδο βασικού μπλοκ [Cla03]. Παρά το γεγονός αυτό, υφίστανται πολλές περιπτώσεις που καθιστούν σκόπιμη την εξαγωγή ειδικών εντολών από κώδικα συμβολομεταφραστή: μετατροπή κώδικα επιπέδου μηχανής για εκτέλεση σε ανανεωμένες αρχιτεκτονικές επεξεργαστών που διαθέτουν εξειδικευμένα χαρακτηριστικά ή για τη σύνθεση ενός RTOS υποβοηθούμενο από υλικό (hardware-assisted RTOS) για δοθέντα ενσωματωμένο επεξεργαστή¹. Πάνω στο SUIF_{vm} IR μπορούν να εφαρμοστούν περάσματα μεταγλωττιστή για το Machine-SUIF (Π 2.1, κεφ. 3.2.1) που είναι ανεξάρτητα της αρχιτεκτονικής όπως βελτιστοποίηση κλειδαρότρυπας (peephole optimization), διάδοση σταθεράς, εξουδετέρωση νεκρού κώδικα, μείωση ισχύος τελεστή, τοπική εξουδετέρωση κοινής υποεκφράσεως (LCSE) [EPFL-passes] για την βελτιστοποίησή του.

¹ Στο εργαλείο YARDstick του ερευνητή Α.Π.Θ. (κεφ. 4) είναι εφικτή η γέννηση ειδικών εντολών από κώδικα με λανθάνουσες εξαρτήσεις (π.χ. κώδικας συμβολομεταφραστή ή αντικείμενος κώδικας) για μια εφαρμογή.

Η πληροφορία του στατικού και δυναμικού προφίλ της εφαρμογής λαμβάνεται αυτόματα με τη βοήθεια περασμάτων ανάλυσης που δέχονται ως είσοδο SUIFvm IR σε μορφή cfg. Η λήψη των αποτελεσμάτων (π.χ. συχνότητες εκτέλεσης βασικών μπλοκ) δυναμικού χαρακτηρισμού γίνεται με εκτέλεση κώδικα C τριών διευθύνσεων που προκύπτει από την μετατροπή του SUIFvm με το πέρασμα m2c ή με μετατροπή του κώδικα συμβολομεταφραστή για την αρχιτεκτονική (όταν αυτό είναι εφικτό) σε αντικείμενο κώδικα και εκτέλεση σε προσομοιωτή ακρίβειας εντολής.

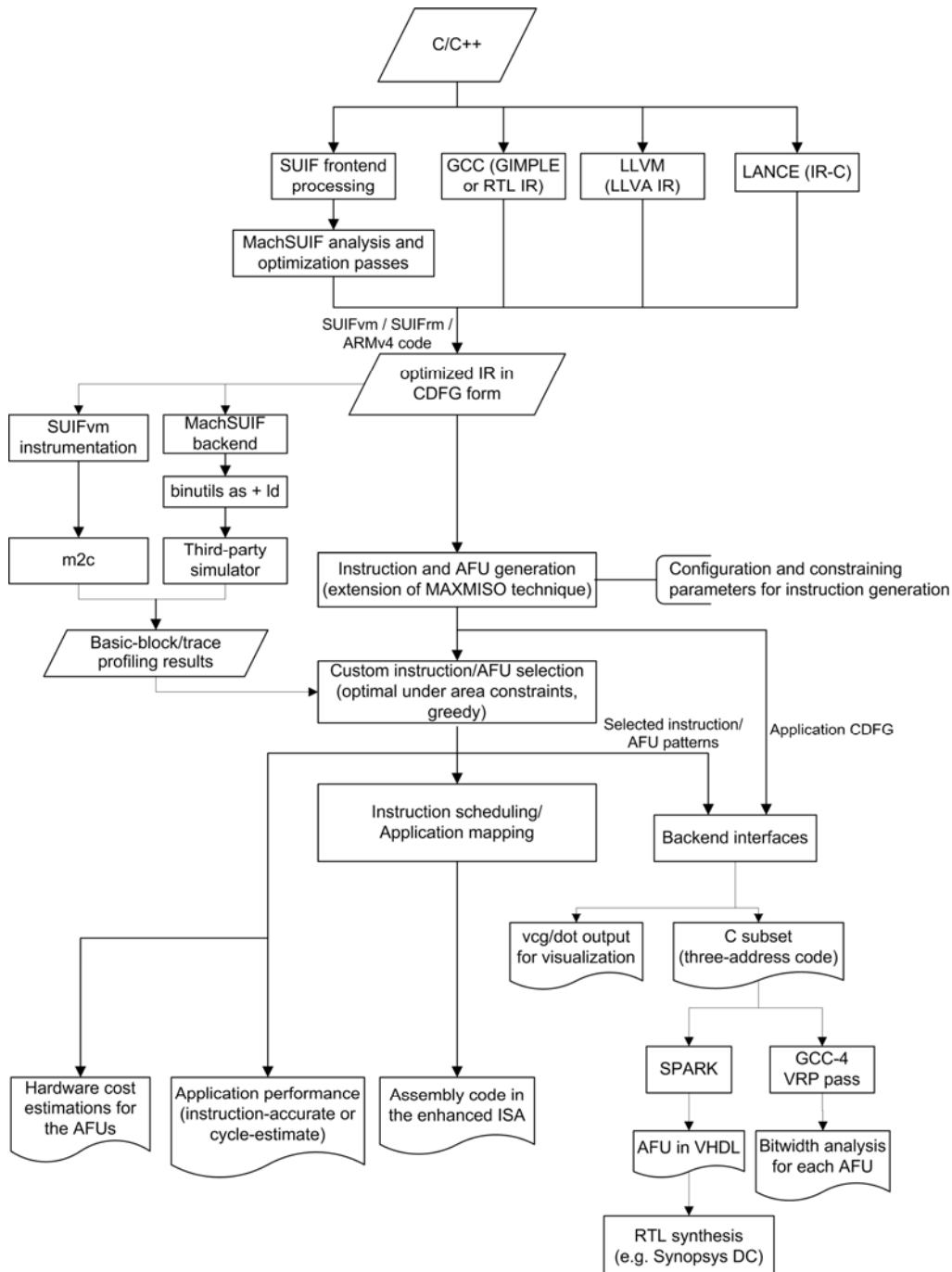


Figure 3.1: Ροή ανάλυσης εφαρμογών και παραμετροποιημένης γέννησης εντολών στο πλαίσιο εργασίας IAGF.

Η διαδικασία αναγνώρισης μοτίβων λαμβάνει χώρα στο επόμενο βήμα, ακολουθούμενη από την επιλογή ειδικών εντολών, η οποία έχει επίσης υλοποιηθεί στο IAGF. Η μηχανή

αναγνώρισης μοτίβων και γένεσης ειδικών εντολών υλοποιεί παραμετροποιημένη παραλλαγή του αλγορίθμου MaxMISO [Poz00], η οποία έχει σχεδιαστεί από τον ερευνητή Α.Π.Θ. και ανακοινωθεί στο [Kan05]. Ο αλγόριθμος MaxMISO (κεφ. 2) αναγνωρίζει τους μέγιστους μη-αλληλεπικαλυπτόμενους συνδεδεμένους υπογράφους του κατευθυνόμενου άκυκλου γράφου εξάρτησης δεδομένων (DDG ή data-dependence DAG) που παράγουν ένα υπολογιστικό αποτέλεσμα (έχουν ένα κόμβο εξόδου που παράγει ένα όρισμα εξόδου). Ο MaxMISO στην αρχική του μορφή δέχεται ως περιορισμούς χρήστη μόνο τον μέγιστο αριθμό ορισμάτων εισόδου (τα οποία μπορούν να παραληφθούν από την ΕΛΜ), αλλά η υλοποίηση του IAGF, ονόματι **GENMISO**, είναι ενισχυμένη με περισσότερες δυνατότητες παραμετροποίησης ώστε να είναι καταλληλότερη για την διαδικασία της γέννησης εντολών σε ένα πρακτικό περιβάλλον. Αυτές οι επιπρόσθετες παράμετροι στον MaxMISO συμπεριλαμβάνουν:

- 1) Το μέγιστο αριθμό κόμβων βασικών εντολών που μπορούν να συμπεριληφθούν σε μια MISO.
- 2) Την καθιέρωση δύο τύπων περιορισμών κόμβων οι οποίοι είναι εφαρμόσιμοι σε οποιαδήποτε κατηγορία εντολών:
 - a. Περιορισμός κόμβου τύπου-A. Ο περιορισμός κόμβου τύπου-A ή συνοριακού κόμβου [Kan05b] απαγορεύει την ανάπτυξη μιας τομής γράφου που απαρτίζεται από στοιχειώδεις εντολές (π.χ. SUIFvm) πέραν της σημειωθείσας εντολής. Έχει παρατηρηθεί ότι η εφαρμογή του σε εντολές μεταφοράς δεδομένων (φόρτωση, αποθήκευση, αντιγραφή από μνήμη σε μνήμη) ευνοεί την γένεση ειδικών εντολών που υλοποιούν σύνθετους τρόπους διευθυνσιοδότησης.
 - b. Περιορισμός κόμβου τύπου-B. Ο περιορισμός κόμβου τύπου-B ή συμπερίληψης κόμβου [Kan05b] δεν επιτρέπει την συμπερίληψη της σημειωθείσας εντολής στην υποψήφια ειδική εντολή που βρίσκεται υπό αναγνώριση. Έχει εφαρμοστεί από τον ερευνητή Α.Π.Θ. σε αλγόριθμο γένεσης εντολών μονής εξόδου (MISO). Συνήθως εφαρμόζεται σε εντολές μεταφοράς ελέγχου (CTI) όπως είναι οι διακλαδώσεις υπό και χωρίς συνθήκη και τα άλματα, και οι λειτουργίες κλήσης και επιστροφής από υποσυνάρτηση.

Στην πράξη, ο περιορισμός κόμβου τύπου-A βοηθά στην αναγνώριση τρόπων διευθυνσιοδότησης για την σύνθεση ειδικών εντολών μεταφοράς δεδομένων. Επίσης, στην πλειονότητα των επεκτάσιμων επεξεργαστών (όπως οι ARC [ARC] και Xtensa [Tensilica]) δεν επιτρέπεται η μεταβολή των μηχανισμών μεταφοράς ελέγχου, όπως αυτό θα συνέβαινε με την προσθήκη ειδικών εντολών στο βασικό ρεπερτόριο εντολών που τροποποιούν τις λειτουργίες ανάκλησης εντολής. Αυτό ισχύει καθώς η επίδρασή τους στη μέγιστη διάρκεια του κύκλου ρολογιού για τον επεξεργαστή είναι λιγότερο προβλέψιμη από ότι στην περίπτωση που οι αρχιτεκτονικές παρεμβάσεις λόγω των ειδικών εντολών τοποθετούνται στα κύρια στάδια διοχέτευσης εκτέλεσης του επεξεργαστή (όπως είναι το στάδιο EX στον επεξεργαστή MIPS-I/R3000 και σε τυπικές υλοποιήσεις της αρχιτεκτονικής DLX).

- 3) Την εφαρμογή ενός ορίου για το μέγιστο αριθμό κύκλων μηχανής που απαιτούνται για την εκτέλεση μιας (ειδικής) εντολής. Οι εντολές μονού κύκλου απαιτούν μικρές μόνο τροποποιήσεις στην κύρια λογική αποκωδικοποίησης εντολών ενώ οι εντολές πολλαπλών κύκλων απαιτούν επιπρόσθετες λειτουργίες ελέγχου (πιθανόν υλοποιημένες με FSM) και είναι ενδεχόμενη η χρήση καταχωρητών κατάστασης χρήστη (user-state registers) [Cla03],[Bis04],[NIOS-II-CI].
- 4) Επιτρέπει την ακριβή εκτίμηση επιδόσεων όταν γίνεται χρήση καταχωρητών κατάστασης χρήστη ή/και για την περίπτωση μονάδων με ιδανική οργάνωση διοχέτευσης (στο τελευταίο θεωρείται ότι οι ΕΛΜ ολοκληρώνουν μία ειδική εντολή ανά

κύκλο και δεν σημειώνονται απώλειες κύκλων στην αρχιτεκτονική διοχέτευσης). Όταν ενεργοποιείται αυτή η επιλογή, το εύρος bit για τη μνήμη δεδομένων θεωρείται ότι είναι μία θύρα ανάγνωσης/μία θύρα εγγραφής ενώ οι ταυτόχρονες αναγνώσεις από το αρχείο καταχωρητών περιορίζονται σε δύο και οι εγγραφές σε μία, για κάθε κύκλο μηχανής. Προκειμένου την εκτέλεση CI πολλαπλών κύκλων, χρησιμοποιούνται προσωρινοί (scratch) καταχωρητές για την μεταφορά των ορισμάτων από/προς τους πόρους αποθήκευσης (αρχείο καταχωρητών, μνήμη δεδομένων) προς/από αυτούς τους καταχωρητές. Για την επιλογή της ιδανικής διοχέτευσης υποτίθεται ότι ο αριθμός των σταδίων εκτέλεσης για την δεδομένη ΕΛΜ που εξυπηρετεί την ειδική εντολή επαρκεί ώστε να συμβαίνει CPI=1, υποθέτοντας ότι η αρχιτεκτονική διοχέτευσης διαθέτει πληρότητα για τους κύκλους μηχανής για τους οποίους η συγκεκριμένη CI είναι υπό εξυπηρέτηση από τον επεξεργαστή.

- 5) Γίνεται εκτίμηση επιδόσεων για βελτιστοποιήσεις του διαδρόμου ελέγχου που σχετίζονται με την εξουδετέρωση των επιβαρύνσεων για την εκτέλεση των λειτουργιών βρόχου. Δυνατότητες εκτίμησης κύκλων μηχανής έχουν υλοποιηθεί τόσο για βρόχους μηδενικής επιβάρυνσης όπως απαντώνται σε γνωστούς DSP επεξεργαστές αλλά και για την ερευνητική αρχιτεκτονική ZOLC (zero-overhead loop controller [Kav05a]).
- 6) Δυνατότητες βέλτιστης επιλογής εντολών κάτω από περιορισμούς επιφάνειας υλικού, καθώς και επιλογή εντολών υπό προκαθορισμένα μετρικά προτεραιότητας. Το πρόβλημα της βέλτιστης επιλογής εντολών υπό περιορισμούς διατυπώνεται ως ένα πρόβλημα 0-1 knapsack το οποίο μπορεί να λυθεί βέλτιστα σε ψευδο-πολυωνυμικό χρόνο με τεχνική δυναμικού προγραμματισμού. Στο συγκεκριμένο πρόβλημα, οι απόλυτες τιμές του κέρδους κύκλων μηχανής και της επιφάνειας υλικού εκφράζονται σε ακέραια πολλαπλάσια του τελεστή λογικού NOT, και αντιστοιχούν στην αξία και βάρος ενός εξεταζόμενου αντικειμένου, ενώ το συνολικό βάρος του «σάκου» (knapsack) δεν μπορεί να ξεπερνά ακέραιο πολλαπλάσιο ενός προκαθορισμένου τελεστή (δηλαδή ακέραιο πολλαπλάσιο μιας αναφοράς επιφάνειας). Για το IAGF οι εξωτερικοί περιορισμοί επιφάνειας εκφράζονται σε όρους της κατάληψης επιφάνειας ενός 32x32-bit πολλαπλασιαστή μονού κύκλου με αποτέλεσμα συντετημημένο στα 32-bit.

Επιπρόσθετα χαρακτηριστικά του IAGF συμπεριλαμβάνουν:

- 1) Υποστήριξη της αρχιτεκτονικής SUIFrm και του συνόλου εντολών ARMv4 (γνωστή υλοποίηση του είναι ο επεξεργαστής ARM7TDMI). Η SUIFrm επιτρέπει τη γέννηση κώδικα με κατανομή καταχωρητών (βασισμένη στην τεχνική της επαναλαμβανόμενης συνάσπισης καταχωρητών [Geo96]).
- 2) Βελτιστοποίηση πολλαπλασιασμού με απλή σταθερά σύμφωνα με τον αλγόριθμο του Bernstein [Ber86],[Bri94] με την διαφορά ότι χρησιμοποιούνται ακριβή κόστη χρόνων καθυστέρησης διάδοσης (θετικοί πραγματικοί αριθμοί) αντί για κόστη σε κύκλους μηχανής (ακέραιοι αριθμοί).
- 3) Βελτιστοποιήσεις άθροισης πολλαπλών ορισμάτων [Beu04] οι οποίες εφαρμόζονται χειρωνακτικά (κατ' εξαίρεση) μετά την κύρια διαδικασία γένεσης εντολών.
- 4) Εφαρμογή αλγορίθμων ισομορφισμού γράφου και γράφου-υπογράφου, υλοποιημένες στη βιβλιοθήκη ταύτισης γράφων VFLib2 [Fog01]. Η εφαρμογή του ισομορφισμού γράφου, π.χ. με χρήση του αλγορίθμου VF2 [Fog01] αναγνωρίζει τα μοναδικά μοτίβα γράφων των ειδικών εντολών, ενώ ισομορφισμοί γράφου-υπογράφου εφαρμόζονται για την ταυτοποίηση εκείνων των μοτίβων που αντιστοιχούν σε μοναδικές ΕΛΜ, οι οποίες εξυπηρετούν ένα υποσύνολο από τη εξαγόμενη βιβλιοθήκη μοτίβων (π.χ. με το πέρασ της γέννησης εντολών). Μια δομή δεδομένων απλά συνδεδεμένης λίστας χρησιμοποιείται για την διατήρηση της πληροφορίας συγγένειας των μοτίβων για τα

οποία διατηρούνται τελικώς στατικές καταχωρήσεις για τα μοτίβα που εξυπηρετούνται από την ίδια ΕΛΜ. Αυτή η πληροφορία αξιοποιείται επίσης για τη δρομολόγηση εντολών στην αρχιτεκτονική του ειδικού επεξεργαστή (π.χ. SUIFvm RISC με επιπρόσθετες ΕΛΜ).

- 5) Υποστήριξη για ρύθμιση ρεπερτορίου εντολών με κατηγοριοποίηση πόρων όπως αυτή που χρησιμοποιείται στο [Goo03]. Ένας τελεστής ή πόρος αποθήκευσης μπορεί να αντιστοιχηθεί σε μια κατηγορία πόρων σύμφωνα με συγκεκριμένες ιδιότητές του. Το IAGF διαθέτει 3 βασικές κατηγορίες: κατηγορία opcode (κωδικού λειτουργίας) όπου κάθε opcode καθορίζει διαφορετική κατηγορία, κατηγορία λειτουργικής ομάδας στην οποία κατατάσσονται εντολές με κοινά χαρακτηριστικά, κατηγορία επεξεργαστικής ομάδας όπου μπορούν να καταταχθούν εντολές με μικρότερες συγγένειες. Τυπικές κατηγορίες λειτουργικής ομάδας καθορίζονται από τις εντολές ALU, πολλαπλασιασμού, διακλάδωσης, φόρτωσης/αποθήκευσης, ολίσθησης κ.λ.π.. Διαφορετικές εντολές οι οποίες διαθέτουν opcode που ανήκουν στην ίδια κατηγορία πόρων ταυτοποιούνται θεωρώντας ότι θα υλοποιηθούν από ένα κοινό πόρο ή λειτουργική μονάδα. Τέτοιες μονάδες συναντώνται σε VLIW επεξεργαστές στους οποίους συνήθως θεωρείται ότι καταλαμβάνουν μία θυρίδα εκτέλεσης (execution slot).
- 6) Το IAGF παρέχει backends για τη μετάφραση των εξαγόμενων ειδικών εντολών/ΕΛΜ για σκοπούς οπτικοποίησης και εκτίμησης υλικού. Οι διαμορφώσεις που υποστηρίζονται για την οπτικοποίηση γράφων είναι οι VCG [VCG] και dot [Graphviz] για την αναπαράσταση του CFG οποιασδήποτε οντότητας της βιβλιοθήκης μοτίβων, ενώ υφίσταται και πειραματικό C backend για λόγους προσομοίωσης ακρίβειας εντολής. Υπάρχει επίσης η δυνατότητα γέννησης κώδικα VHDL είτε από την C περιγραφή (π.χ. με χρήση του SPARK) είτε μέσω μετατροπής στη διαμόρφωση CFG που υποστηρίζεται από δωρεάν διαθέσιμο εργαλείο ανοικτού κώδικα (CFG toolset [CFGtool]).

3.2. Δυνατότητες χρήσης του αλγόριθμου GENMISO εντός του IAGF

Η διαδικασία γέννησης ειδικών εντολών ελέγχεται από ένα σύνολο παραμέτρων ρύθμισης (configuration parameters) και περιορισμούς (constraints). Αυτές διαιρούνται σε τέσσερις κατηγορίες: α) παράμετροι μέγιστης τιμής/περιοχής τιμών, β) παράμετροι μικροαρχιτεκτονικού μοτίβου, και γ) ρυθμίσεις για την ενεργοποίηση αριθμητικών βελτιστοποιήσεων, του αρχιτεκτονικού περιγράμματος (π.χ. ρεπερτόριο εντολών) ή για την αποτίμηση άλλων αποφάσεων (όπως για ρυθμίσεις για την επιλογή ειδικών εντολών). Στον Πίνακα 3.1 παρουσιάζεται η σημειογραφία και οι περιγραφές των παραμέτρων ρύθμισης οι οποίες είναι διαθέσιμες στον χρήστη του IAGF.

Πίνακας 3.1: Οι σημαντικότερες παράμετροι ρύθμισης του IAGF.

	Παράμετρος ρύθμισης	Περιγραφή
Παράμετροι μέγιστης τιμής/ περιοχής τιμών	ni	Μέγιστος αριθμός μοναδικών ορισμάτων εισόδου για έναν υπογράφο εξάρτησης δεδομένων
	nn	Μέγιστος αριθμός κόμβων βασικών εντολών που απαρτίζουν έναν υπογράφο εξάρτησης δεδομένων
	limit-area	Όριο συνολικής επιφάνειας για τις παραγόμενες ειδικές εντολές/ΕΛΜ
	limit-instrs	Μέγιστος αριθμός μοναδικών ειδικών εντολών
	limit-cycles	Όριο του αριθμού κύκλων μηχανής για κάθε MISO

Παράμετροι μικροαρχιτεκτονικού μοτίβου	constraint-[a b] <list>	Περιορισμοί κόμβων τύπου-A (συμπερίληψη κόμβου) ή τύπου-B (συνοριακού κόμβου) για δοθείσα λίστα opcode βασικών εντολών (επιτρέπεται η ταυτόχρονη εφαρμογή περιορισμών και των δύο τύπων)
	user-state-regs	Αποτίμηση της επίδρασης λειτουργιών φόρτωσης/αποθήκευσης από/προς καταχωρητές κατάστασης χρήστη στις επιδόσεις σε κύκλους μηχανής των ειδικών επεξεργαστών
	opt-pipelining	Γίνεται υπόθεση χρήσης των σταδίων διοχέτευσης εκτέλεσης ώστε για όλες τις εντολές να είναι CPI=1
	isa-context	Ρύθμιση του βασικού ρεπερτορίου εντολών (μία επιλογή από SUIFvm, SUIFrm and ARMv4 [ARM])
	zorc	Υπόθεση ότι έχουν εφαρμοστεί μετασχηματισμοί για μηδενική επιβάρυνση βρόχων σύμφωνα με την αρχή ZOLC
Διάφορες ρυθμίσεις	opt-const-mul	Ενεργοποίηση βελτιστοποίησης πολλαπλασιασμού με απλή σταθερά κατά τον αλγόριθμο του Bernstein (υλοποίηση Briggs-Harvey)
	isomorph-<case>-<subcase>-<subsubcase>	Εφαρμογή αλγόριθμου ισομορφισμού γράφου (ορισμένος από <case>) ή ισομορφισμού γράφου-υπογράφου (ορισμένος από <subcase>) στα παραχθέντα μοτίβα υποψηφίων εντολών. Οι ιδιότητες που μπορούν να ταυτοποιηθούν (κατηγοριοποίηση πόρων) καθορίζονται από το πεδίο <subsubcase>.
	iselect[-<priority function>]-<method>	Επιλογή ειδικών εντολών κάτω από δοθείσα συνάρτηση προτεραιότητας και συγκεκριμένη πολιτική (π.χ. άπληστη)

Επιπλέον, το πλαίσιο εργασίας IAGF επιτρέπει την υλοποίηση διαφόρων σχημάτων γένεσης ειδικών εντολών που έχουν περιγραφεί στην διεθνή βιβλιογραφία. Έτσι, επιπρόσθετα με την τροποποίηση GENMISO του αρχικού αλγορίθμου MaxMISO, είναι άμεση η επινόηση κατάλληλων ρυθμίσεων για την υλοποίηση τέτοιων τεχνικών όπως π.χ. είναι μια στατική εκδοχή του αλγορίθμου γένεσης εντολών που βασίζεται σε μινι-γράφους ροής δεδομένων (data-flow mini-graphs [Bra04]). Άλλες μέθοδοι που υποστηρίζονται είναι:

- η τεχνική “bbgrow” [Cas04]
- μια απλοποιημένη εκδοχή του αλγορίθμου επιλογής μοτίβων του [Con04] για μη-αλληλεπικαλυπτόμενα μοτίβα
- ο αλγόριθμος εξαγωγής επικρατούντων μοτίβων (prevalent dataflow pattern extraction) [Sas04]
- η γέννηση αλυσιδωτών λειτουργιών [Goo03]
- γέννηση μοτίβων για την επέκταση ενός RISC επεξεργαστή με επαναπροσδιορίσιμη λειτουργική μονάδα [Vas05].

Ο Πίνακας 3.2 συνοψίζει τις επιλογές ρυθμίσεων που απαιτούνται για την υλοποίηση των προαναφερθεισών μεθόδων γέννησης ειδικών εντολών.

Πίνακας 3.2: Επιλογές ρυθμίσεων για την υποστήριξη τεχνικών γέννησης μοτίβων από τη σχετική βιβλιογραφία.

Μέθοδος γέννησης ειδικών εντολών	Επιλογές ρυθμίσεων						
	#είσοδοι	#κόμβοι	#σταθερές	#κύκλοι μηχανής	Περιορισμός κόμβων τύπου-A	Περιορισμός κόμβων τύπου-B	Ταύτιση τελεστή
MaxMISO [Poz00]	N	–	–	–	καμία	καμία	Απλά orcodes
Instruction generation on data-flow minigraphs [Bra04]	2	–	–	–	cti, φορτώσεις, αποθηκεύσεις	καμία	Απλά orcodes
“bbgrow [Cas04]	–	–	–	–	καμία	καμία	Απλά orcodes
Extension instruction generation for ReRISC [Vas05]	4	8	4	–	cti, μεταφορά μνήμη-μνήμη	φορτώσεις, αποθηκεύσεις	Απλά orcodes
Non-overlapping version of Cong’s algorithm [Con04]	N (=4)	–	–	M (=4)	καμία	καμία	Απλά orcodes
Prevalent dataflow patterns extraction algorithm [Sas04]	8	–	–	–	καμία	καμία	Τύποι λειτουργικής μονάδας
Part of AutoTIE algorithm (generation of fused operations) [Goo03]	2	–	–	1	cti, φορτώσεις, αποθηκεύσεις	καμία	Τύποι λειτουργικής μονάδας

3.3. Αυτοματοποιημένα σενάρια διερεύνησης πεδίου λύσεων για την επέκταση ενσωματωμένων επεξεργαστών

Στο σημείο αυτό, αποτιμάται η προταθείσα προσέγγιση για την επέκταση συνόλου εντολών (με την γέννηση ειδικών εντολών) για ενσωματωμένους επεξεργαστές. Για τα επόμενα πειράματα έγινε αποτίμηση ενός συνόλου γνωστών ενσωματωμένων εφαρμογών που αποτελείται από 4 κρυπτογραφικές εφαρμογές (*crc*, *gost*, *rc5*, *sha*), 9 εφαρμογές που χρησιμοποιούνται στο πλαίσιο των πολυμέσων (*3ss*, *adpcm_dec*, *adpcm_enc*, *fs*, *jpeg_decode*, *mpeg4_senc*, *susan*, *wsq_compress*, *wsq_decompress*), και 7 οι οποίες λυριάρχονται από λειτουργίες ελέγχου (*compress*, *dijkstra*, *engine*, *g3fax*, *patricia*, *procsag*, *stringsearch*). Αυτές οι εφαρμογές δοκιμής έχουν συλλεχθεί από διάφορες πηγές: α) την σουίτα εφαρμογών MiBench [MiBench], β) την σουίτα PowerStone [Lee99], και γ) από πηγές του δημόσιου πεδίου. Δεν κατέστη εφικτή η χρήση όλου του φάσματος του πηγαίου κώδικα των εφαρμογών από ολόκληρες σουίτες (όπως η MiBench) λόγω ασυμβατοτήτων κάποιων από τις εφαρμογές με το frontend του Machine-SUIF (ένας λόγος είναι η αυστηρότητα του c2s SUIF frontend όσον αφορά τις GNU C επεκτάσεις που χρησιμοποιούν οι περισσότερες εφαρμογές μεγάλης έκτασης). Ένας πιο σημαντικός λόγος ήταν η ανεπάρκεια του περάσματος m2c για την γέννηση C κώδικα χαμηλού επιπέδου για την εξαγωγή του προφίλ των εφαρμογών. Ο Πίνακας 3.3 παρέχει σύντομες περιγραφές για την κάθε εφαρμογή δοκιμής.

Πίνακας 3.3: Σύνοψη των εφαρμογών δοκιμής.

Συντόμηση	Περιγραφή εφαρμογής	Αναφορά	Δυναμικές εντολές
Εφαρμογές δοκιμής κρυπτογράφησης (ασφάλειας)			
<i>crc</i>	Cyclic redundancy check	Powerstone	24,276
<i>gost</i>	GOST encryption algorithm	[Cha94]	88,697,924
<i>rc5</i>	The RC5 encryption algorithm	[Riv95]	111,736,947
<i>sha</i>	Secure Hash Algorithm producing an 160-bit message digest for a given input	MiBench	27,501,214
Εφαρμογές πολυμέσων			
<i>3ss</i>	Three-step search block-matching motion estimation	[Kog81]	134,061,502
<i>adpcm_dec</i>	Adaptive Differential Pulse Code Modulation (ADPCM) decoder	MiBench	334,849,208

<i>adpcm_enc</i>	Adaptive Differential Pulse Code Modulation (ADPCM) encoder	MiBench	558,065,396
<i>fs</i>	Full-search block-matching motion estimation	--	1,113,677,881
<i>jpeg_decode</i>	JPEG 24-bit image decompression standard	Powerstone	1,264,008
<i>mpeg4_senc</i>	MPEG-4 context-based shape encoder	[MPEG4senc]	2,794,785,603
<i>susan_smoothing</i>	SUSAN image recognition package	MiBench	856,211,074
<i>wsq_compress</i>	Wavelet/Scalar Quantization image compression	[Kum00]	39,594,690
<i>wsq_decompress</i>	Wavelet/Scalar Quantization image decompression	[Kum00]	64,349,266
Ενσωματωμένες (κυριαρχούμενες από λειτουργίες ελέγχου) εφαρμογές			
<i>compress</i>	The UNIX "compress" utility	Powerstone	73,860
<i>dijkstra</i>	Shortest path calculation between node pairs Dijkstra's algorithm	MiBench	383,438,928
<i>engine</i>	Engine control application	Powerstone	245,031
<i>g3fax</i>	Group 3 FAX decode	Powerstone	965,013
<i>patricia</i>	Searching routing tables in network applications represented as Patricia tries data structures.	MiBench	10,489,292
<i>pocsag</i>	POCSAG communication protocol for paging applications	Powerstone	29,625
<i>stringsearch</i>	Case-insensitive string matching application	MiBench	228,519

Τα επόμενα σχήματα (Σχ. 3.2-3.6) παρουσιάζουν τα πειραματικά αποτελέσματα για τις μελέτες διερεύνησης πεδίου λύσεων σχετικά με τη γέννηση ειδικών εντολών και ΕΛΜ κάτω από διαφορετικούς περιορισμούς. Συλλέχθηκαν κύκλοι εκτέλεσης και αντίστοιχες επιταχύνσεις εφαρμογών, κέρδη κύκλων βασικών μπλοκ, και μετρήσεις επιφάνειας ΕΛΜ για ένα σύνολο 4 βασικών σεναρίων διερεύνησης:

- γέννηση ειδικών εντολών για διαφορετικό αριθμό εισόδων
- γέννηση ειδικών εντολών για περιορισμούς που σχετίζονται με τους επιλεγόμενους κόμβους εντολών
- εκμετάλλευση πολλαπλασιαστών απλής σταθεράς για την βελτιστοποίηση της χρονικής καθυστέρησης και επιφάνειας των ΕΛΜ
- άπληστη επιλογή εντολών κάτω από διαφορετικές συναρτήσεις προτεραιότητας

Τα παραπάνω σεναρία καταγράφονται στον Πίνακα 3.4.

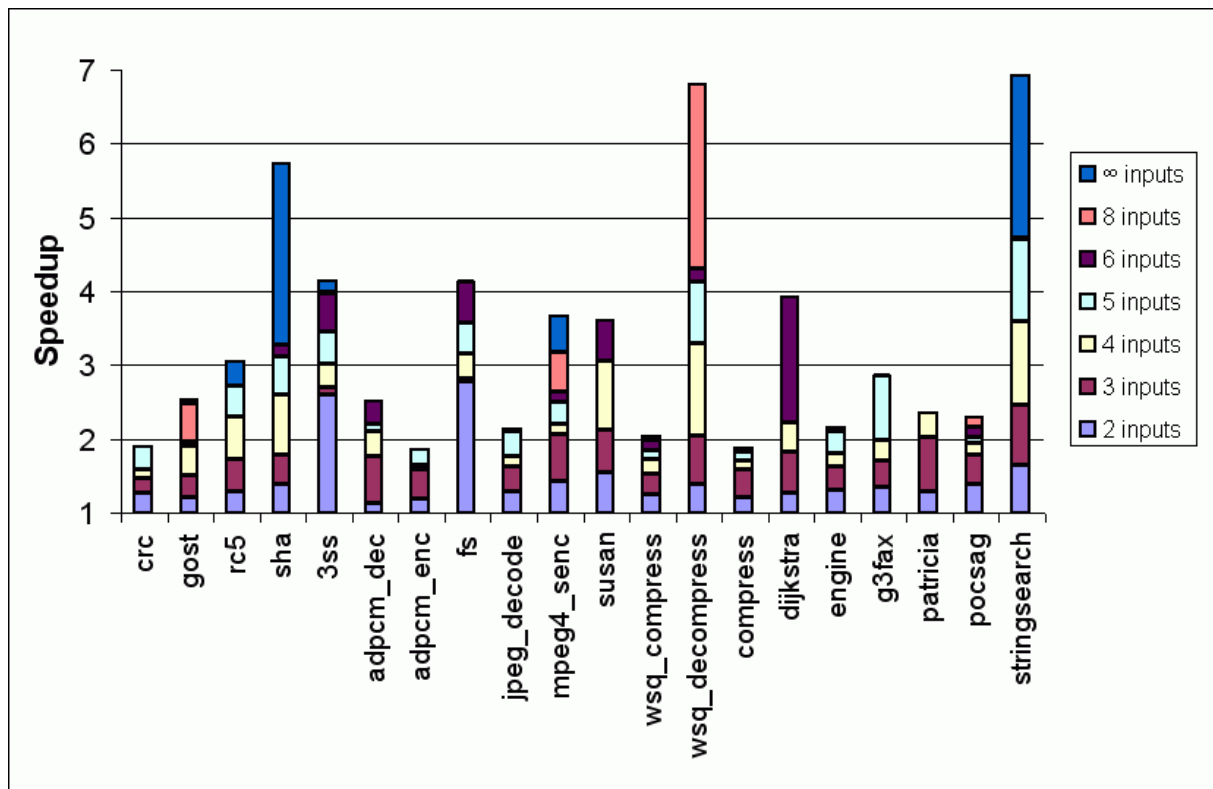
Πίνακας 3.4: Σεναρία διερεύνησης πεδίου σχεδιασμού.

Επιλογή διερεύνησης	Περιγραφή
1	Γέννηση εντολών για διαφορετικούς αριθμούς εισόδων
1A	Βασικό σύνολο επιλογών για την προεπιλογή των περιορισμών κόμβων
1B	Καταχωρητές κατάστασης χρήστη
1C	Καταχωρητές κατάστασης χρήστη και ιδανικές συνθήκες διοχέτευσης
2	Διαφορετικοί περιορισμοί κόμβου
2A	Τύπου-A: καμία / Τύπου-B: καμία
2B	Τύπου-A: εντολές cti / Τύπου-B: καμία
2C	Τύπου-A: cal, ret / Τύπου-B: memcpy
2D	Τύπου-A: εντολές cti / Type-B: εντολές αποθήκευσης
2E	Τύπου-A: εντολές cti / Τύπου-B: εντολές φόρτωσης και αποθήκευσης
2F	Τύπου-A: εντολές cti, φόρτωσης και αποθήκευσης / Τύπου-B: εντολές cti, φόρτωσης και αποθήκευσης
3	Εκμετάλλευση πολλαπλασιαστών απλής σταθεράς για διαφορετικούς αριθμούς ορισμάτων εισόδου και περιορισμούς

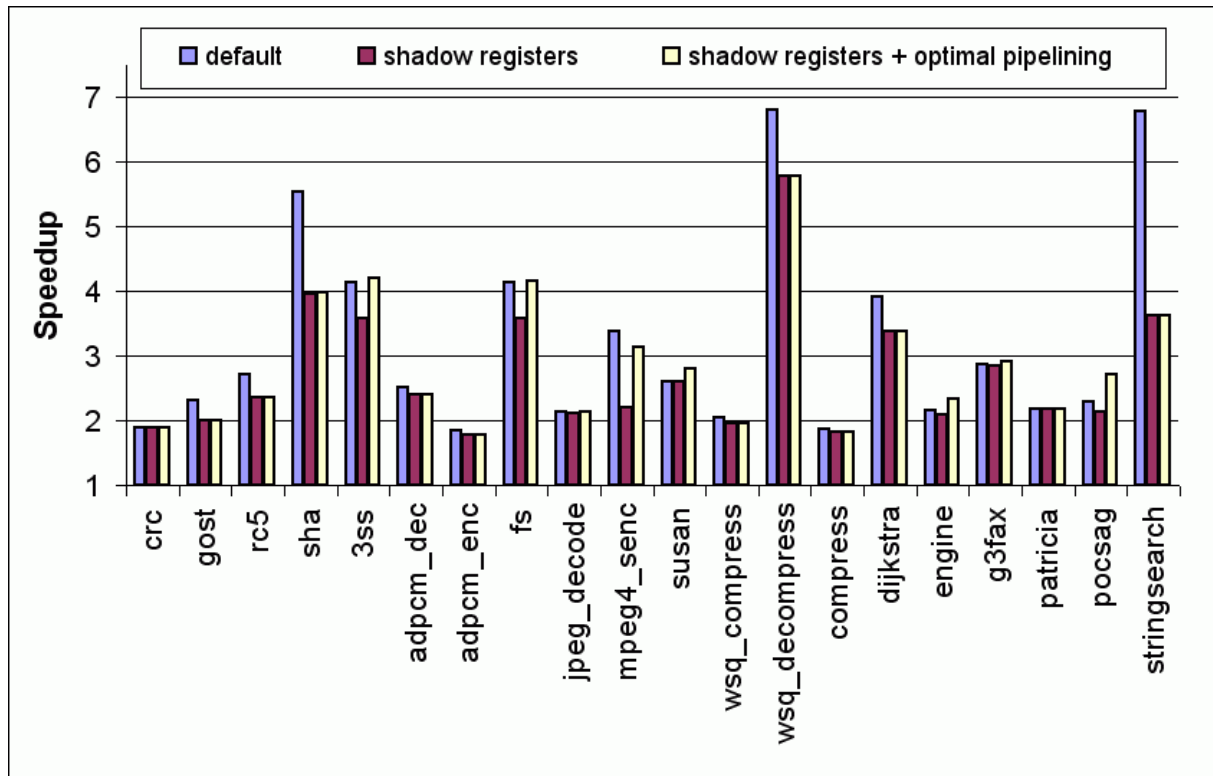
	κόμβων
4	Άπληστη επιλογή εντολών κάτω από δοθείσα συνάρτηση προτεραιότητας
4A	Συνάρτηση προτεραιότητας: Cycle gain
4B	Συνάρτηση προτεραιότητας: Cycle gain per area

3.3.1. Γέννηση εντολών για διαφορετικούς αριθμούς εισόδων

Το Σχ. 3.2 δείχνει τις εκτιμώμενες επιταχύνσεις εφαρμογών για τις εξετασθείσες εφαρμογές για το εξής σύνολο διαφορετικών αριθμών ορισμάτων εισόδου: $n_i = \{2, 3, 4, 5, 6, 8, \infty\}$. Θα πρέπει να σημειωθεί ότι οι επιταχύνσεις έχουν ληφθεί με εκτίμηση κύκλων εκτέλεσης για ένα θεωρητικό RISC επεξεργαστή με n στάδια διοχέτευσης, ρυθμιζόμενη απαίτηση κύκλων του πολλαπλασιαστή (1 ως 4 κύκλοι) και τις περισσότερες λειτουργίες ροής ελέγχου να έχουν επιβάρυνση ενός επιπρόσθετου κύκλου (όπως συμβαίνει στον MIPS-I R3000 επεξεργαστή όταν γίνεται πλήρης εκμετάλλευση των θυρίδων καθυστέρησης). Για τις αναφερόμενες επιταχύνσεις, δεν έχουν ληφθεί υπ' όψη τυχόν περιορισμοί στην συνολική επιφάνεια της θεωρούμενης διάταξης υλοποίησης του επεξεργαστή (π.χ. ολοκληρωμένου τύπου ASIC ή στοιχείο FPGA) και έτσι κάθε φορά όλες οι επιλεγόμενες εντολές λαμβάνονται υπ' όψη στις τελικές εκτιμήσεις. Αυτή η περίπτωση προσομοιάζει υποθετικό επαναπροσδιορισμό υλικό, το οποίο μπορεί να υποστηρίξει μεγάλο αριθμό ειδικών εντολών, και περιορίζεται μόνο από τις απαιτήσεις για τη φόρτωση νέας ρύθμισης (configuration). Για την δυνατότητα αυτή υποτίθεται ότι είναι εφικτός ο μερικός επαναπροσδιορισμός (partial reconfiguration) του υλικού σε πολύ λίγους κύκλους μηχανής (ιδανικά σε έναν). Αυτό θα μπορούσε να επιτευχθεί με επιβολή της κατάλληλης δρομολόγησης εντολών και ρυθμίσεων υλικού τόσο για τις βασικές λειτουργίες του επεξεργαστή όσο και για τις ανάγκες των ειδικών εντολών.



(α)



(β)

Σχήμα 3.2: (α) Επιτάχυνση εφαρμογής για διαφορετικούς αριθμούς ορισμάτων εισόδου. (β) Μέγιστη επιτάχυνση σε σχέση με τη χρήση καταχωρητών κατάστασης χρήστη (καταχωρητές «σκιάς») και ιδανική διοχέτευση.

Με παρατήρηση του Σχ. 3.2.α εξάγεται ότι η σχέση μεταξύ των επιτυγχανόμενων επιταχύνσεων εφαρμογής και την αύξηση του ορίου για τα ορίσματα εισόδου είναι μονότονη αλλά το ποσόν της αυξητικής επιτάχυνσης εξαρτάται από την εκάστοτε περίπτωση εφαρμογής. Το συμπέρασμα αυτό υποστηρίζει την αναγκαιότητα για λεπτομερή διερεύνηση ώστε την εξαγωγή του περιθωρίου διερεύνησης για την κάθε εφαρμογή.

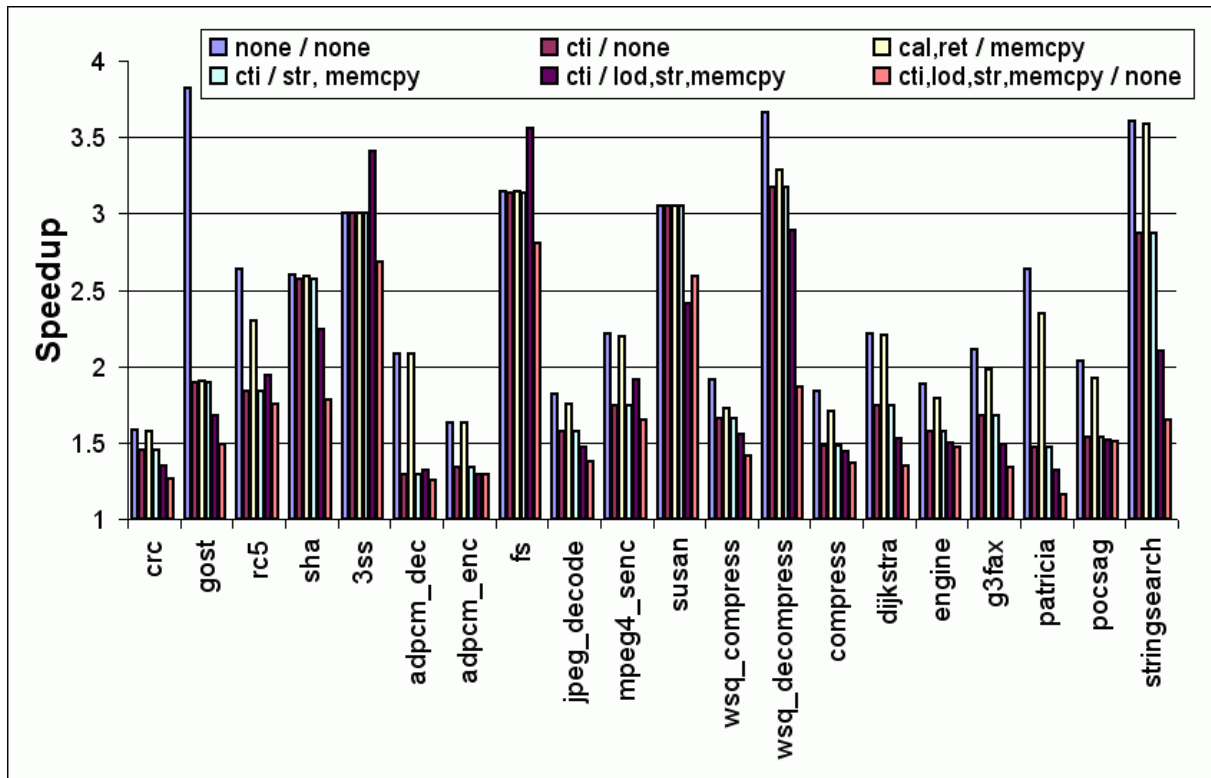
Το Σχ. 3.2.β αναλύει την επίδραση της εισαγωγής των καταχωρητών κατάστασης χρήστη και της θεώρησης για ιδανική διοχέτευση (για τα στάδια διοχέτευσης EXI είναι έτσι ώστε: $CPI=1$) για την εκτέλεση ειδικών εντολών χωρίς περιορισμούς για τα ορίσματα εισόδου. Εξάγεται ότι οι περιορισμένες θύρες του αρχείου καταχωρητών (2 θύρες ανάγνωσης/ 1 θύρα εγγραφής) επιδρούν αρνητικά στην επιτάχυνση εφαρμογής της περίπτωσης 1Α του Πίνακα 3.4 κατά 8.5%, η οποία επιβάρυνση μπορεί να θεωρηθεί αποδεκτή για την υποστήριξη ειδικών εντολών με 3 ή περισσότερα ορίσματα εισόδου. Για συγκεκριμένες εφαρμογές (*3ss*, *fs*, *susan*, *g3fax*, *pocsag*), ο συνδυασμός της χρήσης καταχωρητών κατάστασης χρήστη και ιδανικής διοχέτευσης (σημειώνεται ότι δεν γίνεται συνεκτίμηση των κινδύνων δεδομένων) παρέχει επιταχύνσεις στο ίδιο επίπεδο ή και καλύτερες της περίπτωσης 1Α (για απεριόριστο εύρος δεδομένων από και προς τις ΕΛΜ και με επίτρεψη εκτέλεσης πολλαπλών κύκλων). Αυτό δικαιολογείται από το γεγονός ότι σε αυτές τις εφαρμογές οι εντολές πολλαπλών κύκλων συνεισφέρουν σημαντικά στην συνολική επιτάχυνση εφαρμογής και ότι τέτοιες υψηλές επιταχύνσεις μπορούν να προσεγγιστούν με ένα σχετικά μέτριο αριθμό ορισμάτων εισόδου.

3.3.2. Επιτάχυνση εφαρμογής κάτω από διαφορετικούς περιορισμούς κόμβων

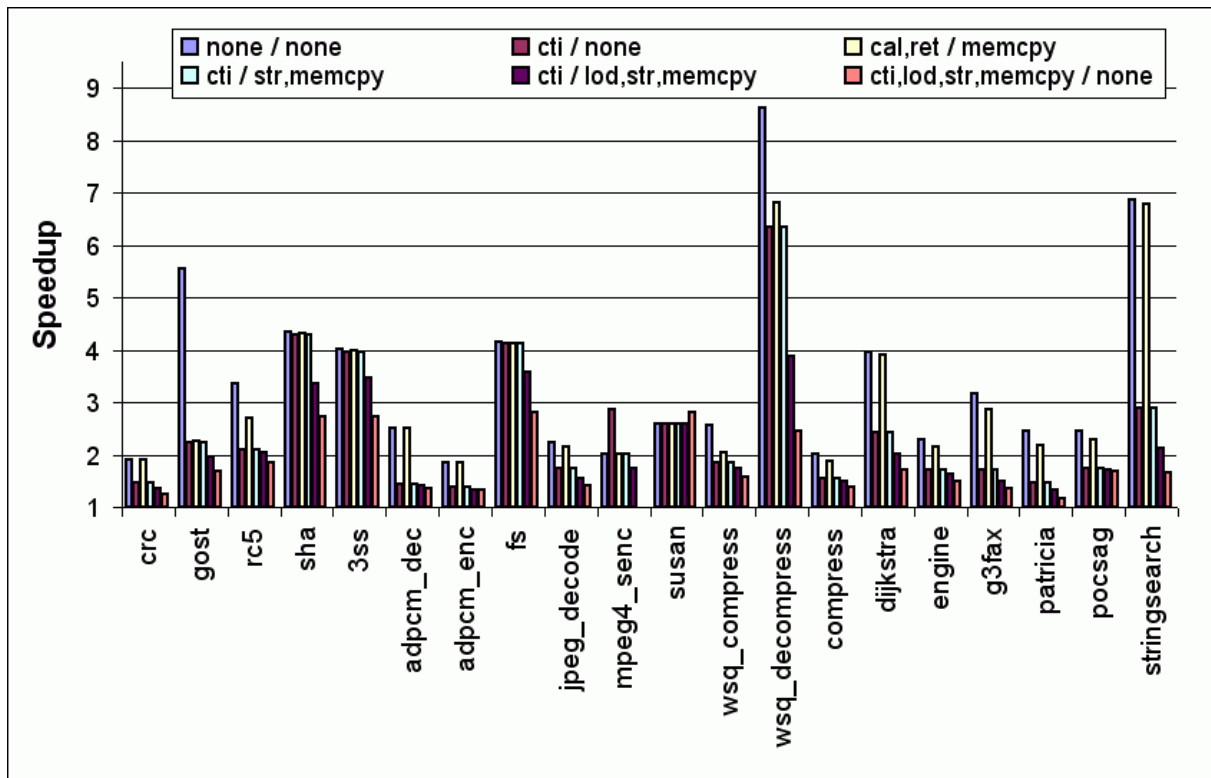
Η διερεύνηση των διαφορετικών περιορισμών κόμβων αναφέρεται στη ρύθμιση των

περιορισμών συμπερίληψης κόμβου καθώς και συνοριακού κόμβου όπως αυτοί καθορίστηκαν στην ενότητα 3.1. Στη σχετική βιβλιογραφία [Poz01] έχει αναφερθεί ότι η αύξηση του αριθμού θυρών της μνήμης δεδομένων οι οποίες είναι προσβάσιμες από τις ΕΛΜ έχει περιορισμένη σημασία. Πιο συγκεκριμένα, υποστηρίζεται ότι οι λειτουργίες φόρτωσης/αποθήκευσης μπορούν να μετακινηθούν σε ένα πρώιμο βήμα δρομολόγησης μέσα στο αντίστοιχο βασικό μπλοκ ώστε περισσότερο σημαντικοί υπολογισμοί σε δεδομένα εντός υπογράφων λαμβάνουν χώρα σε μεταγενέστερα βήματα δρομολόγησης τα οποία συνήθως αντιστοιχούν σε λίγους κύκλους μηχανής.

Στο Σχ. 3.3 παρουσιάζονται αποτελέσματα από τη μελέτη πλήθους συνδυασμών περιορισμών κόμβων αναφορικά με περιορισμούς σε λειτουργίες μεταφοράς δεδομένων και ροής ελέγχου για τις εξετασθείσες εφαρμογές δοκιμής. Η καθιέρωση τέτοιων περιορισμών για εντολές ροής ελέγχου παρέχει έναν επιπρόσθετο βαθμό ελευθερίας στη διαδικασία διερεύνησης του πεδίου σχεδιασμού σε σύγκριση με την συμβατική περίπτωση που εκτιμάται στην πλειονότητα των επεκτάσιμων επεξεργαστών της βιβλιογραφίας, για τους οποίους δεν επιτρέπεται η τροποποίηση των μηχανισμών μεταφοράς ελέγχου για λόγους που έχουν αναφερθεί προηγουμένως. Στο Σχ. 3.3 παρατηρείται ότι η περίπτωση άνευ περιορισμών έχει καλύτερες επιδόσεις από τις υπόλοιπες εναλλακτικές κατά μία εκτιμώμενη μέση τιμή του 15.3%. Με την εξαίρεση των εφαρμογών *3ss* and *fs*, για τις οποίες τα βέλτιστα μετρικά επιτάχυνσης λαμβάνονται για την περίπτωση 2E, καλύτερα αποτελέσματα επιτυγχάνονται για χαλαρότερους περιορισμούς. Η έκβαση αυτή μπορεί να ερμηνευθεί από το γεγονός ότι οι εντολές ροής ελέγχου έχουν χαμηλή προτεραιότητα σε μια κατάταξη εντολών με τοπολογική ταξινόμηση σε ένα βασικό μπλοκ. Έτσι, είναι συχνό να έχουν αναπτυχθεί ειδικές εντολές τύπου MISO με μία *cti* εντολή ως κόμβο εξόδου, με περισσότερες εντολές μεταφοράς δεδομένων να έχουν απορριφθεί από την MISO εξαιτίας περιορισμού στον αριθμό ορισμάτων εισόδου. Όταν η *cti* εξαιρείται από συμπερίληψη, είναι πιθανό, παρά το γεγονός ότι η ακριβής έκταση της επίδρασης κυμαίνεται κατά περίπτωση, ότι ο αλγόριθμος θα επιλέξει υπολογιστικές δομές μεγαλύτερων διαστάσεων για το ίδιο όριο στον αριθμό ορισμάτων εισόδου. Επίσης, η εισαγωγή σύνθετων εντολών ροής ελέγχου (περίπτωση 2A) προσφέρει βελτίωση στην επιτάχυνση εφαρμογής κατά 15.5% σε σχέση με την περίπτωση 2B. Παρ' όλα αυτά πρέπει να σημειωθεί ότι η συμπερίληψη εντολών κλήσης και επιστροφής από ρουτίνα μπορεί να επισύρει σημαντική επιβάρυνση στις επιδόσεις σε όρους πόρων αποθήκευσης (π.χ. στοίβα PC, παράθυρα καταχωρητών) η οποία δεν μπορεί να εκτιμηθεί από την τρέχουσα έκδοση του IAGF.



(α)



(β)

Σχήμα 3.3: Επιτάχυνση εφαρμογής για εναλλακτικούς περιορισμούς κόμβων: (α) $n_{i,MAX} = 4$, (β) $n_{i,MAX} = 8$.

3.3.3. Επίδραση βελτιστοποιήσεων πολλαπλασιασμού απλής σταθεράς

Ο σκοπός αυτού του σεναρίου διερεύνησης είναι η ποσοτικοποίηση της επίδρασης των βελτιστοποιήσεων πολλαπλασιασμού απλής σταθεράς στην επιτάχυνση εφαρμογής. Από τα αντίστοιχα αποτελέσματα προκύπτει ότι μόνο 8 από τις 20 εφαρμογές που έχουν εξεταστεί μπορούν να ωφεληθούν από την αριθμητική βελτιστοποίηση αυτού του τύπου. Οι εφαρμογές αυτές είναι οι: *3ss*, *adpcm_dec_ifc* (ADPCM αποκωδικοποιητής στον οποίο έχει εφαρμοστεί βελτιστοποίηση if-conversion), *dijkstra*, *fs*, *mpeg4_senc*, *rocsag*, *wsq_compress*, και *wsq_decompress*. Βρέθηκε ότι η άμεση εφαρμογή αυτής της βελτιστοποίησης προσφέρει μόνο 2.84% επιπρόσθετης επιτάχυνσης κατά μέσο όρο για τις προαναφερθείσες 8 εφαρμογές με μέγιστο του 5.2% για την *rocsag*. Η αιτιολόγηση για το αποτέλεσμα αυτό είναι: α) το γεγονός ότι ο πολλαπλασιασμός μεταβλητών (συνδυαστική καθυστέρηση διάδοσης: 2.164 ns) είναι μόνο 3.16 φορές πιο αργός από την τοπολογία αθροιστή που χρησιμοποιείται από το εργαλείο σύνθεσης για την εκτίμηση των μετρικών βασικών τελεστών (LeonardoSpectrum), β) η έλλειψη βελτιστοποιήσεων εύρους bit οι οποίες θα μπορούσαν να μειώσουν περαιτέρω τόσο την χρονική καθυστέρηση όσο και την επιφάνεια των ΕΛΜ. Αυτές οι τιμές βελτιώνονται ελαφρώς όταν έχει χρησιμοποιηθεί (κατά την εξαγωγή του SUIFvm IR) το πέρασμα *strength_reduct* για την απλοποίηση περιπτώσεων διαίρεσης με σταθερά και πολλαπλασιασμούς που ανάγονται σε δυνάμεις του 2, αλλά παραμένουν κάτω του 8% της αθροιστικής επιτάχυνσης εφαρμογής.

3.3.4. Επιλογή ειδικών εντολών – Λύση με άπληστη τεχνική υπό περιορισμό αριθμού ειδικών εντολών και διαφορετικών μετρικών προτεραιότητας

Ενώ η επιλογή εντολών με τεχνική δυναμικού ή ακέραιου γραμμικού προγραμματισμού θα μπορούσε να εφαρμοστεί, η υψηλή υπολογιστική πολυπλοκότητα έχει σημαντικό αντίκτυπο στο χρόνο εκτέλεσης της αντίστοιχης λειτουργίας του IAGF λόγω του μεγέθους του σχετικού πεδίου λύσεων. Για την υλοποίηση ενός άπληστου επιλογέα εντολών, υιοθετήθηκε ο ευρεστικός αλγόριθμος που περιγράφεται στο [Yu04] για την επιτάχυνση της όλης διαδικασίας. Η βασική ιδέα είναι η ανάθεση προτεραιοτήτων στα μοτίβα των ειδικών εντολών. Τότε επιλέγονται οι περισσότερες ωφέλιμες εντολές αρχίζοντας από αυτές με την υψηλότερη προτεραιότητα. Για το σκοπό αυτό ορίστηκαν δύο συναρτήσεις προτεραιότητας:

$$\text{Cycle gain (κέρδος κύκλων): } \text{Priority} \left(\sum_j C_{i,j} \right) = \sum_j \{ P_{i,j} \times f_{i,j} \} \quad (4.1)$$

η οποία επιβάλλει τις βέλτιστες επιδόσεις σε ταχύτητα εκτέλεσης ανεξάρτητα των απαιτήσεων σε επιφάνεια των ΕΛΜ και:

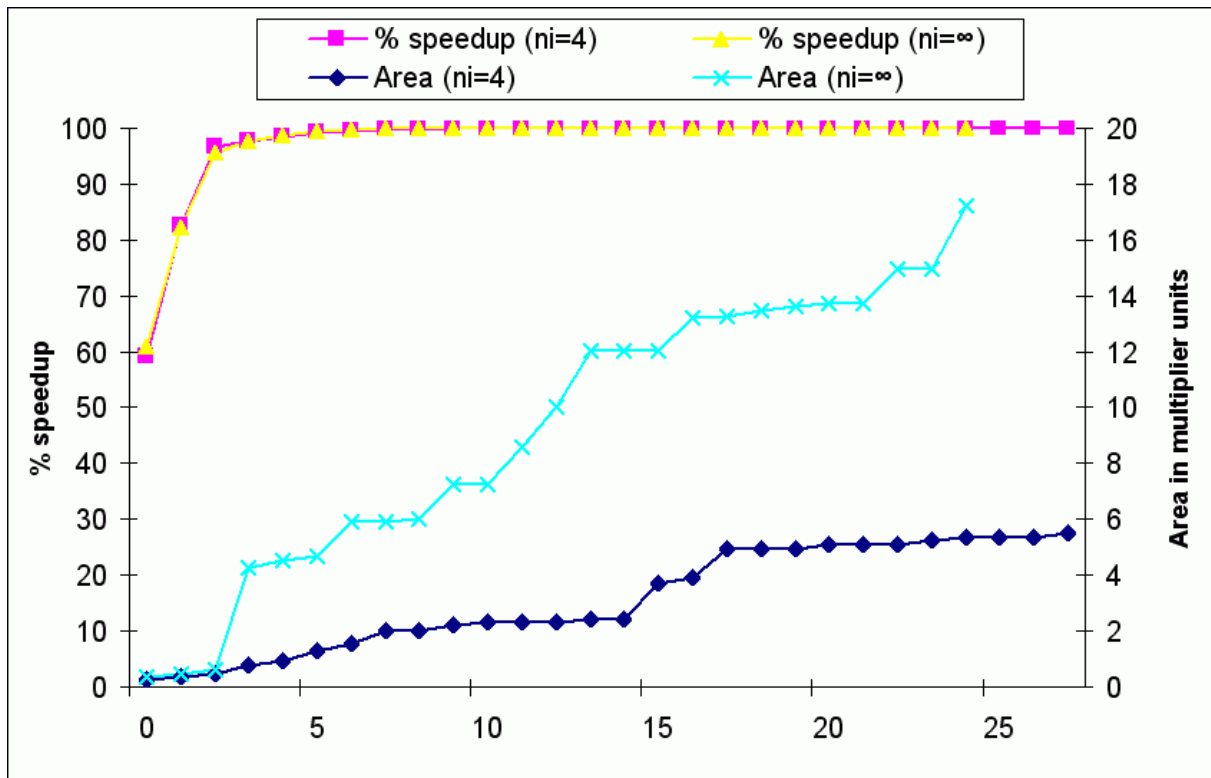
$$\text{Cycle gain/Area (κέρδος κύκλων ανά επιφάνεια): } \text{Priority} \left(\sum_j C_{i,j} \right) = \sum_j \{ P_{i,j} \times f_{i,j} \} / A_i \quad (4.2)$$

όπου $C_{i,j}$ σημειώνει την i -th υποψήφια ειδική εντολή με j διαφορετικά αντίτυπα (εμφανίσεις) σε ολόκληρο το πρόγραμμα εφαρμογής, $f_{i,j}$ είναι η συχνότητα εκτέλεσης βασικού μπλοκ που συνδέεται με το συγκεκριμένο αντίτυπο και A_i το κόστος επιφάνειας για την υποψήφια CI. Αυτές οι συναρτήσεις προτεραιότητας επιβάλλουν διαφορετικά αντικείμενα: η εξίσωση (4.1) μεγιστοποιεί το κέρδος σε επιδόσεις για κάθε ισομορφική υποψήφια CI στην εφαρμογή όταν η επιφάνεια υλικού δεν τίθεται ως ζήτημα ενώ η εξίσωση (4.2) ποσοτικοποιεί και την αξιοποίηση της επιφάνειας υλικού.

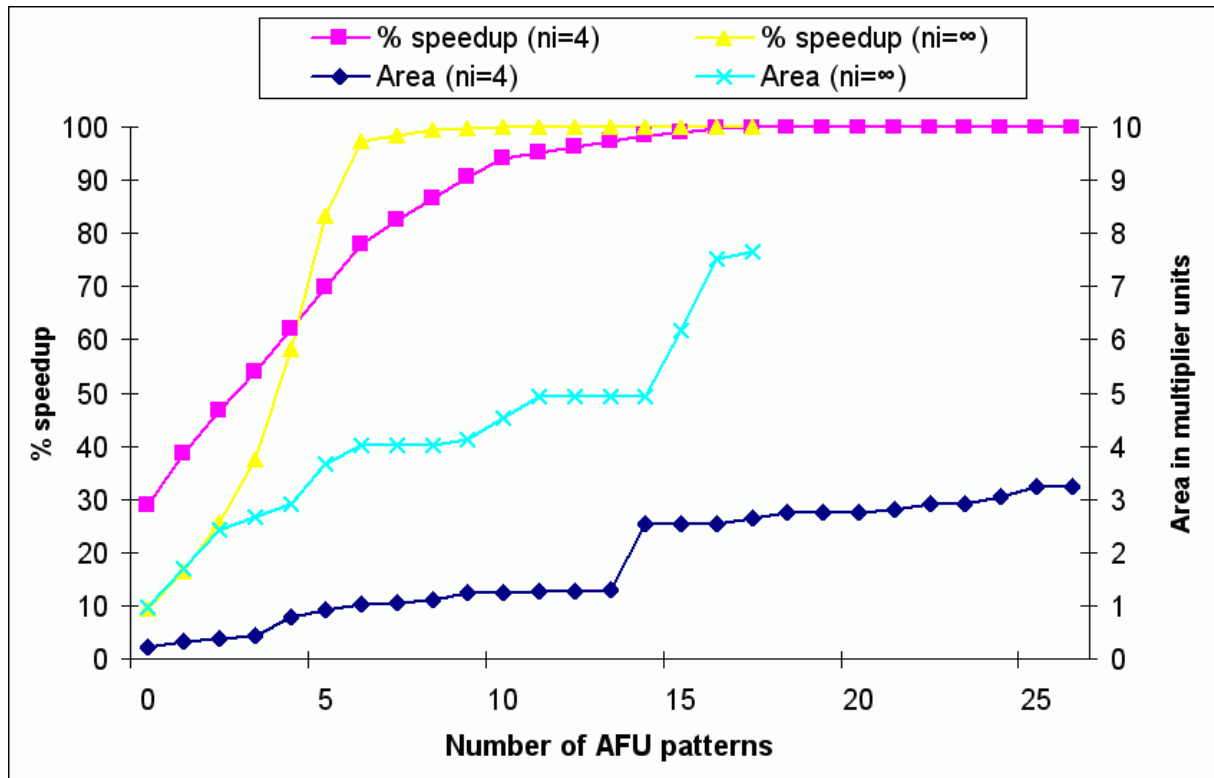
Το Σχ. 3.4 παρουσιάζει εκτενή αποτελέσματα για δύο τυπικές εφαρμογές από τα πεδία των πολυμέσων και της ασφάλειας, ονόματι *fs* και *sha* για δύο διαφορετικές τιμές του περιορισμού αριθμού ορισμάτων εισόδου: $n_i = \{4, \infty\}$. Παρ' όλο που αναγνωρίζονται δεκάδες

υποψήφιες εντολές για αυτές τις εφαρμογές, μόνο λίγες (4 και 12 αντίστοιχα για την κάλυψη του 95% της μέγιστης επιτάχυνσης εφαρμογής) συνεισφέρουν σημαντικά στο χρόνο εκτέλεσης της αντίστοιχης εφαρμογής για οποιαδήποτε συνάρτηση προτεραιότητας από τις δύο. Ο αριθμός των απαιτούμενων ΕΛΜ για την επίτευξη των προαναφερθέντων επιπέδων επιτάχυνσης κυμαίνεται από 3 (*fs*) ως 22 (*wsq_decompress*), ενώ η απαίτηση επιφάνειας για την υλοποίηση ειδικών εντολών σε αντίστοιχες ΕΛΜ είναι μικρότερη των 5 πολλαπλασίων της επιφάνειας πολλαπλασιαστή για όλες τις εφαρμογές. Μια σύνοψη των αποτελεσμάτων για το σύνολο εφαρμογών δοκιμής δίνεται στον Πίνακα 3.5.

Τέλος το Σχ. 3.5 συγκρίνει άμεσα τα υπέρ και κατά των συναρτήσεων προτεραιότητας που χρησιμοποιούνται στη διαδικασία επιλογής ειδικών εντολών. Για την εφαρμογή *fs*, το μετρικό επιτάχυνσης κλιμακώνει περίπου με τον ίδιο τρόπο για τα δύο διαφορετικά αντικείμενα, αλλά μπορεί να φανεί ότι το 'Cycle gain/Area' μπορεί να θεωρηθεί το καλύτερο, καθώς για την ίδια περίπου επιτάχυνση, οι απαιτήσεις επιφάνειας είναι σημαντικά μειωμένες. Τέτοια δεν είναι η περίπτωση και του *sha*, καθώς υφίσταται εμφανές όφελος στην επιτάχυνση για συνάρτηση προτεραιότητας 'Cycle gain' έναντι της 'Cycle gain/Area', μιας και οποιεσδήποτε διαφορές στην κατάληψη επιφάνειας είναι ελαφρώς μικρότερες. Αυτό συμβαίνει καθώς 2 λιγότερες εντολές πρέπει να επιλέγονται για την επίτευξη του 95% της συνολικής επιτάχυνσης εφαρμογής για το προαναφερθέν αντικείμενο.

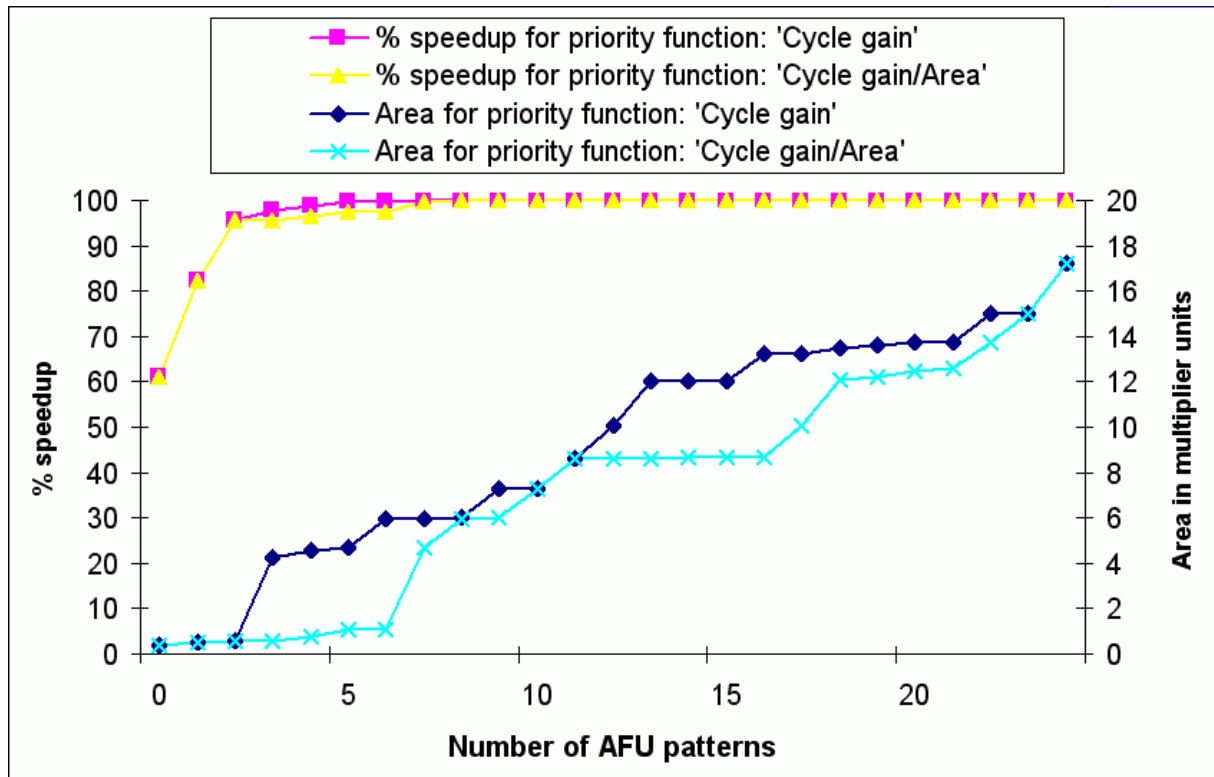


(α)

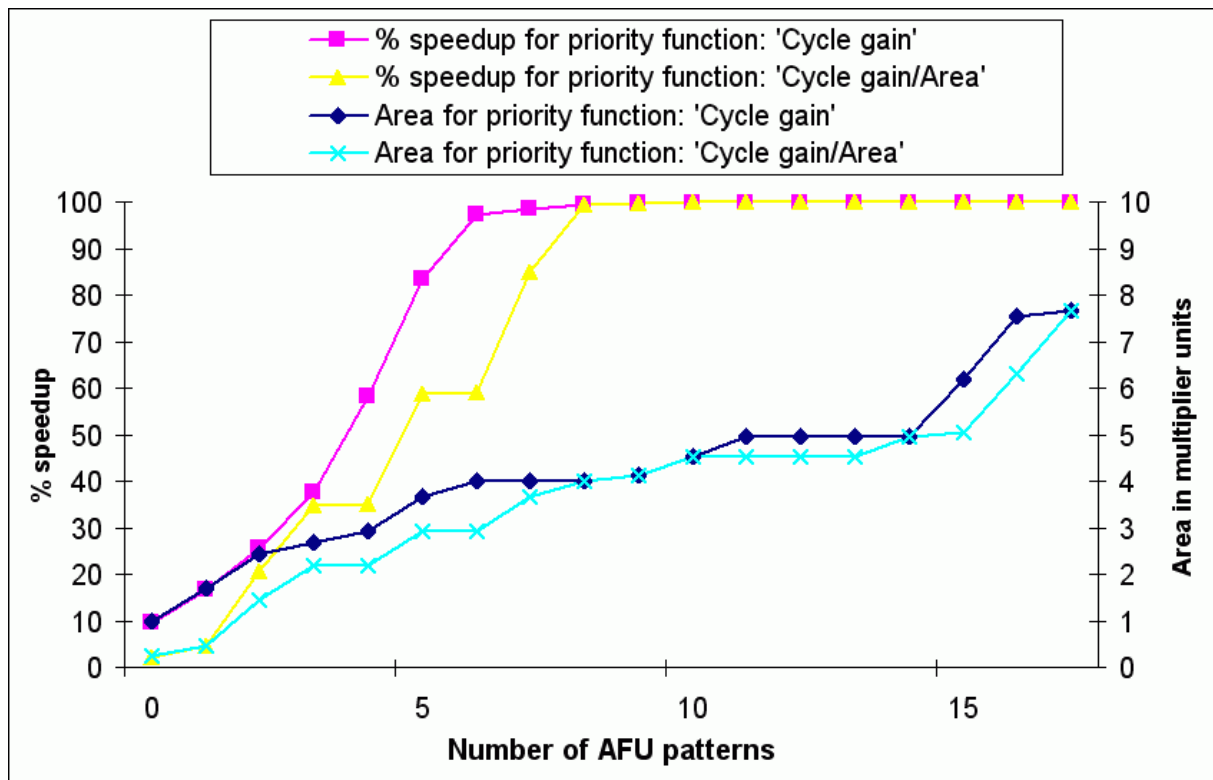


(β)

Σχήμα 3.4: Επιλογή ειδικών εντολών υπό το μετρικό προτεραιότητας 'Cycle gain' για: (α) Την εφαρμογή *fs*, (β) Την εφαρμογή *sha*.



(α)



(β)

Σχήμα 3.5: Επιλογή ειδικών εντολών υπό διαφορετικά μετρικά προτεραιότητας: 'Cycle gain' vs 'Cycle gain/Area' για: (α) fs ($n_i=\infty$), (β) sha ($n_i=\infty$).

Πίνακας 3.5: Επιτάχυνση εφαρμογής και απαιτήσεις επιφάνειας για αύξοντα αριθμό των επιλεγμένων ΕΛΜ. Οι τιμές λαμβάνονται ως μέση τιμή των ακόλουθων περιπτώσεων: (α) 'Cycle gain', (β) 'Cycle gain/Area' με $n_i=\{4,8, \infty\}$.

(α)

Εφαρμογή δοκιμής	Στο 95% της μέγιστης επιτάχυνσης	Επιφάνεια ΕΛΜ σε μονάδες πολλαπλασιαστή	Στο 100% της μέγιστης επιτάχυνσης	Επιφάνεια ΕΛΜ σε μονάδες πολλαπλασιαστή
crc	4	0.705	19	2.349
gost	8	1.827	16	2.532
rc5	8	3.087	23	7.771
sha	9	2.964	22	6.050
3ss	5	2.836	28	17.479
adpcm_dec	6	0.527	8	0.771
adpcm_enc	8	0.356	10	0.835
fs	3	0.560	27	14.091
jpeg_decode	14	5.372	35	12.084
mpeg4_senc	9	8.420	50	30.221
susan	3	2.656	16	12.102
wsq_compress	13	3.295	39	11.650
wsq_decompress	18	9.006	37	13.141
compress	17	5.769	26	7.521
dijkstra	4	0.864	10	1.573
engine	11	2.460	23	11.004
g3fax	6	0.832	16	3.061
patricia	4	0.441	4	0.441
pocsag	18	1.173	28	2.110



stringsearch	5	0.909	11	1.703
--------------	---	-------	----	-------

(β)

Benchmark	At 95% of max. speedup	Area in multiplier units	At 100%	Area in multiplier units
crc	4	0.705	19	2.349
gost	10	1.471	16	2.532
rc5	9	3.087	23	7.771
sha	12	2.976	22	6.050
3ss	6	2.892	28	17.479
adpcm_dec	6	0.527	8	0.771
adpcm_enc	8	0.356	10	0.835
fs	3	0.560	27	14.091
jpeg	16	4.529	35	12.084
mpeg4_senc	12	5.893	50	30.221
susan_smoothing	3	2.656	16	12.102
wsq_compress	15	3.000	39	11.650
wsq_decompress	20	8.182	37	13.141
compress	18	2.262	26	7.521
dijkstra	4	0.864	10	1.573
engine	12	2.578	23	11.004
g3fax	6	0.811	16	3.061
patricia	4	0.441	4	0.441
pocsag	19	1.190	28	2.110
stringsearch	7	0.851	11	1.703

4. ΓΕΝΝΗΣΗ ΚΑΙ ΕΠΙΛΟΓΗ ΕΝΤΟΛΩΝ ΜΕ ΤΟ YARDSTICK

4.1. Πρακτικά ζητήματα στη γέννηση/επιλογή ειδικών εντολών

Στις μοντέρνες ροές σχεδιασμού επεξεργαστών αλλά και των SoC που τους χρησιμοποιούν, συχνά υφίσταται η απαίτηση για την επέκταση ενσωματωμένων επεξεργαστών με χαρακτηριστικά ειδικού σκοπού που στοχεύουν ομάδα/ομάδες εφαρμογών του ενδιαφέροντος. Για την επίτευξη αυτού, είναι αναγκαία η εκμετάλλευση των δυνατοτήτων που προσφέρονται από τους σύγχρονους τροποποιήσιμους/επεκτάσιμους επεξεργαστές ώστε την εναρμόνιση του ρεπερτορίου εντολών και της μικροαρχιτεκτονικής τους με τις στοχευόμενες εφαρμογές. Ένας σημαντικός παράγοντας που συχνά καθορίζει την επιτυχία αυτής της διαδικασίας είναι το επίπεδο και η έκταση των αυτοματισμών που υφίστανται στα επιμέρους στάδια της στατικής/δυναμικής ανάλυσης των εφαρμογών και της σύνθεσης ειδικών εντολών για την επεξεργαστική πλατφόρμα που έχει αποφασιστεί.

Σε αυτό το πλαίσιο κινείται το πρωτότυπο εργαλείο YARDstick (Your ASIP is Ready to Deploy), το οποίο είναι εργαλείο αυτοματοποίησης του ερευνητή Α.Π.Θ. Νικόλαου Καββαδία για την γέννηση και αξιολόγηση επεκτάσεων υλικού ειδικού σκοπού (ASHEs), ώστε την χρήση τους σε επεξεργαστές ειδικού σκοπού που βρίσκονται στο στάδιο της ανάπτυξης. Το YARDstick μπορεί να θεωρηθεί ως ένα δομικό στοιχείο (building block) της ανάπτυξης ΕΕΣ (ASIP) το οποίο ενοποιεί τα επιμέρους βήματα της ανάλυσης των εφαρμογών και της γέννησης και επιλογής εντολών για ενδιάμεσες αναπαράστασεις μεταγλωττιστή (compiler IRs) της επιλογής του σχεδιαστή/χρήστη. Σε ένα περιβάλλον σχεδιασμού το οποίο ενσωματώνει το YARDstick, συγκεκριμένα προβλήματα που απαντώνται στον παραδοσιακό σχεδιασμό ASIP αντιμετωπίζονται με επάρκεια:

- Η υποδομή διερεύνησης πεδίου λύσεων απελευθερώνεται από ιδιοσυγκρασιακές αποφάσεις που έχουν ληφθεί στο επίπεδο του μεταγλωττιστή και του προσομοιωτή.
- Ο σχεδιαστής ASIP ενισχύεται με την ελευθερία της καταγραφής των αρχιτεκτονικών στόχων της επιλογής του και της προσθήκης νέων υλοποιήσεων τόσο αναλύσεων όσο και τεχνικών γέννησης/επιλογής ειδικών εντολών.

Προκειμένου την ανάδειξη των δυνατοτήτων της προσέγγισης YARDstick εξετάζονται ενδιαφέροντα σενάρια διερεύνησης: η ποσοτικοποίηση της επίδρασης βελτιστοποιήσεων του μεταγλωττιστή που είναι ανεξάρτητες της αρχιτεκτονικής και της επιλογής στοιχείων του αρχιτεκτονικού περιγράμματος στόχου σε όρους του συνόλου βασικών λειτουργιών και του μοντέλου μνήμης πάνω στη γέννηση/επιλογή ειδικών εντολών κάτω από διαφορετικούς περιορισμούς εισόδου/εξόδου.

Θα πρέπει να σημειωθεί εδώ ότι για τους θεωρούμενους ASIP της μεθοδολογίας εξετάζονται δύο βασικοί τρόποι αρχιτεκτονικής επέκτασης: ενσωμάτωση ειδικών λειτουργικών μονάδων και πόρων μνήμης σε στενή σύζευξη (tight coupling) [Nios-II] ή η χαλαρή σύζευξη επιταχυντών υλικού (hardware accelerators) με τον επεξεργαστή μέσω καλώς καθορισμένης διεπαφής (τοπική διασύνδεση ή δίαυλος). Πρόσφατη έρευνα [Sir07] υποστηρίζει την χρήση και των δύο τεχνικών αρχιτεκτονικής επέκτασης, αποδεικνύοντας ότι και οι δύο τρόποι μπορούν να εξετάζονται ταυτόχρονα εφόσον το πρόβλημα τυποποιείται ως κατάτμηση δύο επιπέδων (two-level partitioning).

Αποτελεί συχνά παρατηρούμενο φαινόμενο στον σχεδιασμό επεξεργαστών ειδικού σκοπού/ASIP να μην αντιμετωπίζονται πρακτικά ζητήματα που αναδεικνύονται από στοιχεία της ροής σχεδιασμού που υποτίθεται ότι είναι αμετάβλητα (invariant elements) σε όλες τις περιπτώσεις:

- a) Υποθέσεις για το IR στο οποίο χαρτογραφείται ο κώδικας της στοχευόμενης εφαρμογής (ή ομάδας εφαρμογών) επηρεάζουν άμεσα την ποιότητα της λύσης του προβλήματος σύνθεσης εντολών.
- b) Η υποδομή διερεύνησης είναι δέσμια των συμβάσεων των λογισμικών εργαλείων ανάπτυξης εφαρμογών (κυρίως του μεταγλωττιστή και των προσομοιωτών).
- c) Η ανάγκη για προσαρμοστικότητα σε διαφορετικούς μεταγλωττιστές και προσομοιωτές.
- d) Η υποστήριξη για εφαρμογές που είναι συνταχθείσες σε περιγραφή χαμηλού επιπέδου (κώδικας συμβολομεταφραστή) για σκοπούς όπως η μετανάστευση εφαρμογής (application/code migration) ανάμεσα σε επεξεργαστές-μέλη μιας οικογένειας επεξεργαστών και η αντίστροφη μηχανική (reverse engineering).

Όλα τα παραπάνω ζητήματα αντιμετωπίζονται με επιτυχία από το πρωτότυπο εργαλείο YARDstick το οποίο ενοποιεί τεχνικές γέννησης και επιλογής ειδικών εντολών με μία ευέλικτη υποδομή ενδιάμεσης αναπαράστασης που επιτρέπει τον αντικατοπτρισμό αποφάσεων του σχεδιαστή/χρήστη οι οποίες είναι δύσκολο να εφαρμοστούν με άλλο τρόπο. Για παράδειγμα η χρήση ενός IR με εγγενή υποστήριξη για λειτουργίες επιπέδου bit (bit-level operations) μπορεί να αποδώσει σημαντικά διαφοροποιημένες ειδικές εντολές (π.χ. ως επεκτάσεις συνόλου εντολών ή Instruction Set Extensions – ISEs) σε σχέση με ένα IR που δεν προσφέρει αυτή τη δυνατότητα. Επίσης, στο YARDstick είναι εφικτό να μετρηθεί άμεσα η επίδραση συγκεκριμένων μετασχηματισμών μεταγλωττιστή που είναι εξαρτώμενες από την αρχιτεκτονική, όπως είναι η κατανομή καταχωρητών στην ποιότητα και τις επιπτώσεις των παραγόμενων ειδικών εντολών (στο χρόνο εκτέλεσης ενσωματωμένης εφαρμογής που τις αξιοποιεί κατάλληλα), ένα ζήτημα που αναγνωρίζεται αλλά δεν ποσοτικοποιείται σε άλλες εργασίες [Cla03],[Cas04]. Επιπλέον, το YARDstick παρέχει ευκολίες για την εκτίμηση επιδόσεων και τη λήψη του προφίλ των στοχευόμενων εφαρμογών με τον καθορισμό στατικών και δυναμικών μετρικών των εφαρμογών όπως τύποι δεδομένων, στατιστικά για τις προσπελάσεις στα διάφορα επίπεδα ιεραρχίας μνήμης δεδομένων/εντολών καθώς και συχνότητες εκτέλεσης (εντολών, βασικών μπλοκ). Η είσοδος της εφαρμογής μπορεί να είναι από γλώσσα υψηλού (π.χ. ANSI C, C++) ή χαμηλού επιπέδου (γλώσσα συμβολομεταφραστή για μια αρχιτεκτονική στόχο ή εικονική μηχανή). Ένας αριθμός από πρόσφατες τεχνικές αναγνώρισης και επιλογής ειδικών εντολών έχουν υλοποιηθεί εντός του YARDstick ενώ παρέχονται εκτιμητές επιδόσεων υλικού (για τον υπολογισμό του χρόνου διάδοσης/εκτέλεσης και της επιφάνειας υλικού των ειδικών λειτουργικών μονάδων) και διεπαφές για τρίτα εργαλεία για το σκοπό της σύνθεσης των ειδικών μονάδων σε υλικό (σε VHDL) από CFG αναπαραστάσεις.

Είναι σημαντικό να σημειωθεί ότι η διερμηνεία των ειδικών λειτουργικοτήτων που επιτελούνται από τις ειδικές λειτουργικές μονάδες (EAM) εξαρτάται από το πλαίσιο που καθορίζεται από το αρχιτεκτονικό περιγράμμα. Έτσι μπορεί να αντιπροσωπεύουν

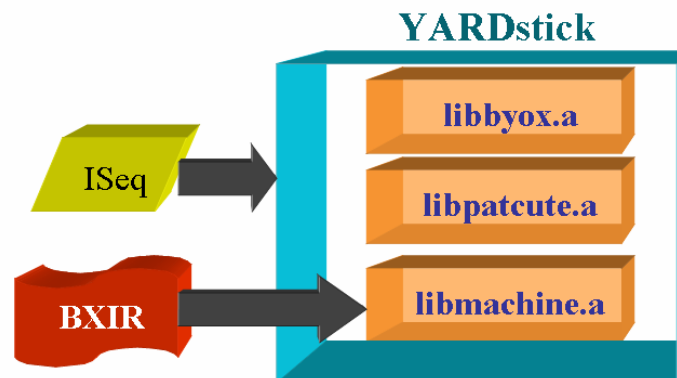
- ειδικές εντολές επέκτασης (ISEs) σε μια βασική αρχιτεκτονική συνόλου εντολών και οι οποίες πρέπει να λαμβάνονται υπ' όψη από το διάδρομο ελέγχου του βασικού επεξεργαστή (λογική αποκωδικοποίησης, επέκταση των υπηρεσιών εξυπηρέτησης διακοπών)
- ειδικές εντολές ενός ASIP ελεγχόμενες από προγραμματιζόμενο ελεγκτή ή αφοσιωμένες (dedicated) λειτουργίες που κάνουν χρήση ελεγκτή τύπου FSM και διατίθενται προς χρήση ως μη-προγραμματιζόμενοι επιταχυντές υλικού, σε χαλαρή σύζευξη με τον επεξεργαστή.

4.2. Στα εσώτερα του YARDstick

Ο κύριος ρόλος του YARDstick είναι η διευκόλυνση και επιτάχυνση της διαδικασίας διερεύνησης πεδίου λύσεων (DSE) σε ετερογενείς ροές σχεδιασμού ΕΕΣ όπου τα εργαλεία ανάπτυξης εφαρμογών (μεταγλωττιστής, συμβολομεταφραστής, συνδότης, άλλα εργαλεία χειρισμού κώδικα μηχανής, προσομοιωτής/εκσφαλματωτής) σε πολλές περιπτώσεις ελλείπονται δυνατοτήτων για DSE και/ή έχουν σχεδιαστεί με διαφορετικές ή ασταθείς (αλλάζουν συχνά, π.χ. περισσότερες των μία φορές στον κύκλο ανάπτυξης ενός ASIP) διεπαφές (επίπεδου αρχείων ή API). Έτσι είναι συχνό το φαινόμενο να απαιτούνται σημαντικοί ανθρώπινοι (και χρηματικοί) πόροι για την προσθήκη δυνατοτήτων ετεροχρονισμένα (after-thoughts) και η χρονοβόρα αντιμετώπιση προβλημάτων διεργαστικότητας (interoperability) μεταξύ των εργαλείων ανάπτυξης, ιδιαίτερα όταν αυτό συμβαίνει στις διεπαφές του μεταγλωττιστή και του προσομοιωτή.

4.2.1. Ο πυρήνας του YARDstick

Η τρέχουσα υποδομή YARDstick, όπως απεικονίζεται στο Σχ. 4.1, απαρτίζεται από τρεις διακριτές συνιστώσες στο επίπεδο του πυρήνα (**libByoX**, **libPatCUTE**, **libmachine**), και από ένα σύνολο από backends για την εξαγωγή γράφων ροής ελέγχου, βασικών μπλοκ και ειδικών εντολών προκειμένου την οπτικοποίηση, την προσομοίωση και την σύνθεση τους σε υλικό από το επίπεδο CDFG/RTL. Οι προαναφερθείσες συνιστώσες του YARDstick αποτελούν λογισμικές βιβλιοθήκες των οποίων ο πηγαίος κώδικας είναι σε C/C++ και λειτουργικές εκδόσεις τους έχουν αναπτυχθεί σε διαφορετικά περιβάλλοντα ανάπτυξης και λειτουργικά συστήματα: MS Windows, Cygwin/x86, Linux/x86. Οι βιβλιοθήκες libByoX και libPatCUTE είναι ανεξάρτητες της στοχευόμενης κάθε φορά αρχιτεκτονικής (σε επίπεδο IR) και μόνο η βιβλιοθήκη libmachine πρέπει να επαναστοχεύεται για διαφορετικές προδιαγραφές IR.



Σχήμα 4.1: Η υποδομή YARDstick.

libByoX

Η βιβλιοθήκη libByoX υλοποιεί το βασικό μέρος του YARDstick API και παρέχει κατάλληλα frontend και μεθόδους για τον χειρισμό (αρχικοποίηση, ενημέρωση, κατάργηση) εσωτερικών δομών δεδομένων (π.χ. για την διατήρηση της πληροφορίας που προκύπτει από την ανάλυση μιας εφαρμογής δοκιμών). Η βασική διαμόρφωση στην οποία καταγράφεται η πληροφορία για τις αναλυόμενες εφαρμογές είναι η ISeq (ιστορικό όμοιο για το "InstructionSequence"). Πιο λεπτομερώς, η βιβλιοθήκη ByoX (πρόκειται για αρκτικόλεξο του ευφημισμού "Bring Your Own Compiler and Simulator") παρέχει τα εξής:

- Τον συντακτικό αναλυτή (parser) ISeqinfo για τη διαμόρφωση αρχείου ISeq. Το ISeq είναι μια διαμόρφωση IR μεταγλωττιστή για την καταγραφή CDFG επίπεδης

μορφής των αναλυόμενων εφαρμογών με δυνατότητα υποστήριξης στατικών απλών αναθέσεων (flat CFG with/without SSA). Μια ιδιαιτερότητα του ISeq είναι η ρητή (explicit) καταγραφή των εξαρτήσεων δεδομένων εντός των βασικών μπλοκ των εφαρμογών με τη μορφή λίστας ακμών (edge list).

- Τον συντακτικό αναλυτή CFGinfo για τη διαμόρφωση αρχείου CFG στην οποία καταγράφονται οι εξαρτήσεις ελέγχου (και οι τύποι τους) μεταξύ των βασικών μπλοκ στα CFG των εφαρμογών.
- Απλή και εύκολα διαχειρίσιμη διεπαφή αρχείου για τις διαμορφώσεις ISeq και CFG όσο και για τα αποτελέσματα από αναλύσεις μεταγλωττιστή, όπως αναλύσεις ροής ελέγχου/δεδομένων για τον χρόνο ζωής καταχωρητών και τους φυσικούς βρόχους, τα οποία μπορούν να εισαχθούν στο YARDstick σύμφωνα με τις αντίστοιχες BNF γραμματικές τους.
- Ένα API για τον χειρισμό του IR το οποίο επιτρέπει την συγγραφή εξωτερικών περασμάτων αναλύσεων και βελτιστοποιήσεων (σε επίπεδο αντίστοιχο αυτό ενός μεταγλωττιστή).
- Παραμετροποίηση για ένα αρχιτεκτονικό μοτίβο χωρίς έμφυτους περιορισμούς για το ρεπερτόριο εντολών.

Στη διαμόρφωση ISeq καταγράφεται η ακόλουθη πληροφορία μιας εφαρμογής:

- Ο πίνακας συμβόλων καθολικής εμβέλειας (global symbol table).
- Η λίστα των συναρτήσεων (procedure list). Σε κάθε συνάρτηση αντιστοιχεί ένας γράφος ροής ελέγχου (CFG) για τον οποίο καταγράφονται οι καταχωρήσεις εξαρτήσεων δεδομένων, εξάγεται ο πίνακας συμβόλων τοπικής εμβέλειας, και η λίστα των αναθέσεων (για τις υποστηριζόμενες λειτουργίες, οι οποίες δεν είναι απαραίτητο να είναι τριών διευθύνσεων). Είναι σημαντικό ότι είναι εφικτή η αυτόματη εξαγωγή διαφορετικών όψεων (facets) του πίνακα τοπικών συμβόλων, π.χ. με απλές αναφορές ανά κατεύθυνση (είσοδος ή έξοδος) για κάθε μεταβλητή ή με επίτρεψη, σε άλλη περίπτωση, πολλαπλών σημείων ορισμού για την ίδια μεταβλητή.

Με το συνδυασμό των αναπαραστάσεων ISeq και CFG ανακτάται η πληροφορία από κάθε συνάρτηση μιας εφαρμογής στο επίπεδο CFG.

libPatCUTE

Επιπλέον, στο YARDstick έχει υλοποιηθεί ένας αριθμός από μεθόδους γέννησης και επιλογής εντολών ως τμήμα της βιβλιοθήκης PatCUTE (Pattern-based Custom UniT Exploration). Η γέννηση ειδικών εντολών περιλαμβάνει την αναγνώριση MIMO (Multiple-Input Multiple-Output) ή MISO (Multiple-Input Single-Output) μοτίβων ISeq κάτω από περιορισμούς που θέτει ο χρήστης. Οι τεχνικές γέννησης εντολών που υφίστανται στην τρέχουσα έκδοση της libPatCUTE είναι οι εξής:

- Η τεχνική MaxMISO [Poz00] για την αναγνώριση των υπογράφων μέγιστων διαστάσεων με ένα κόμβο εξόδου με τη χρήση αλγορίθμου γραμμικής πολυπλοκότητας.
- Διερεύνηση για εντολές τύπου MISO υπό περιορισμούς για τον μέγιστο αριθμό ορισμάτων εισόδου/εξόδου, και για δύο τύπους περιορισμών που σχετίζονται με την γεινίαση κόμβων [Kan05b] (Κεφ. 3).
- Γέννηση ειδικών εντολών τύπου MIMO. Στην περίπτωση του YARDstick δεν γίνεται αναζήτηση για μέγιστα μοτίβα MIMO (maximal MIMO patterns), αλλά γίνεται χρήση μιας γρήγορης ευρεστικής τεχνικής, παρόμοια με αυτή της εργασίας [Pot07]. Ο αλγόριθμος εύρεσης MIMO εντολών χρησιμοποιεί την υπόθεση ότι το κέρδος σε

χρόνο εκτέλεσης για την εφαρμογή λόγω της MIMO (ανάλογο των κύκλων εκτέλεσης εφόσον δεν μεταβάλλεται η μέγιστη συχνότητα λειτουργίας του στοχευόμενου επεξεργαστή) είναι μεγαλύτερο από αυτό που προκύπτει για κάθε άλλη MIMO που αποτελεί υπογράφο της. Στην περίπτωση που ο χρήστης του YARDstick απενεργοποιήσει την επιβολή της ευρεστικής υπόθεσης, τότε χρησιμοποιείται ο βασικός αλγόριθμος αναγνώρισης MIMO μοτίβων που έχει γενικά εκθετική πολυπλοκότητα.

Όταν ενεργοποιείται η διαδικασία γέννησης ειδικών εντολών, κατασκευάζεται μια λίστα από ειδικές εντολές (CIs) από τα εξαγόμενα ISeq μοτίβα, η οποία μπορεί να φιλτραριστεί μέσω ελέγχων ισομορφισμών γράφου και γράφου-υπογράφου [Fog01] κατά τη διαδικασία απομάκρυνσης των πλεονάζοντων μοτίβων (δηλαδή αυτών για τα οποία έχουν ήδη αναγνωριστεί ήδη ισομορφικά τους).

Για την επιλογή ειδικών εντολών υπάρχει η δυνατότητα της χρήσης ενός πολύ γρήγορου greedy (άπληστου) επιλογέα ο οποίος υποστηρίζει διαφορετικές συναρτήσεις προτεραιότητας (τα αντίστοιχα μετρικά είναι «κέρδος κύκλων» και «κέρδος κύκλων ανά μονάδα επιφάνειας»).

Ένα σημαντικό χαρακτηριστικό του YARDstick είναι ότι οι CIs μπορούν να εκφραστούν σε μορφή ISeq με τον ίδιο τρόπο που αυτό γίνεται για τμήματα των αρχικών εφαρμογών (CFG, βασικά μπλοκ) ώστε οι ίδιες δομές δεδομένων και αναλύσεις να επαναχρησιμοποιούνται για περαιτέρω χειρισμό των παραγόμενων ειδικών εντολών. Για παράδειγμα, βιβλιοθήκες (λίστες) από ειδικές εντολές μπορούν να εισαχθούν στο YARDstick (π.χ. για την οπτικοποίηση ή περαιτέρω ανάλυσή τους).

libmachine

Η βιβλιοθήκη libmachine αποτελεί τη μόνη βασική συνιστώσα του YARDstick για την οποία απαιτείται η επαναστόχευσή της για διαφορετικές αρχιτεκτονικές-στόχους που καθορίζονται από το χρήστη. Η απαιτούμενη πληροφορία για τις αρχιτεκτονικές στόχους προσδιορίζεται με τη διαμόρφωση BXIR (ByoX IR) η οποία υποστηρίζει τα απαραίτητα σημασιολογικά στοιχεία για τον καθορισμό πληροφορίας καθολικής εμβέλειας (τύποι δεδομένων, ομάδες βασικών λειτουργιών) και επιπέδου λειτουργίας (ορίσματα/έντελα, διερμηνεία της λειτουργίας του κάθε τελεστή λειτουργίας, κόστος σε επιφάνεια και χρονισμό για αντίστοιχες υλοποιήσεις σε υλικό).

Backends

Στα πλαίσια του YARDstick έχει αναπτυχθεί μεγάλος αριθμός από σύμβατα backends για την εξαγωγή και μετατροπή των αποτελεσμάτων (κύριων όσο και παράπλευρων) της διαδικασίας γέννησης και επιλογής εντολών σε χρήσιμες διαμορφώσεις. Κάτι τέτοιο είναι απαραίτητο για την αξιοποίηση των αποτελεσμάτων αυτών από τρίτα εργαλεία και θεωρείται σημαντική συνιστώσα του YARDstick σε πρακτικό επίπεδο.

Τα υποστηριζόμενα backends κατά την έκδοση του YARDstick όπως αυτό παρουσιάστηκε στο University Booth του διεθνούς έγκριτου Ευρωπαϊκού συνεδρίου DATE 2007 [Kan07] είναι τα εξής:

- Backend που υποστηρίζει την εξαγωγή των ειδικών εντολών σε υποσύνολο της ANSI C (υφίστανται και επιμέρους ρυθμίσεις προτιμήσεων). Η δυνατότητα αυτή είναι ιδιαίτερα χρήσιμη προκειμένου την ενσωμάτωση των ειδικών εντολών σε τρίτα εργαλεία ανάπτυξης εφαρμογών για τον θεωρούμενο ASIP όπως προσομοιωτές/εξομοιωτές, κ.λ.π.
- Εξαγωγή πληροφορίας σε διαμόρφωση GDL (VCG) [VCG] για την οπτικοποίηση CFG, BB και CIs.
- Εξαγωγή σε διαμόρφωση dot (Graphviz) [Graphviz] για τους προαναφερθέντες

σκοπούς οπτικοποίησης.

- Μετατροπή σε μια επέκταση της διαμόρφωσης CDFG [CDFGtool] που επιτρέπει την δρομολόγηση (scheduling), allocation (κατανομή πόρων) και binding (δέσμευση πόρων) των αντίστοιχων αναπαραστάσεων γράφου για τις CI (είναι εφαρμόσιμο όμως και για ολόκληρα βασικά μπλοκ). Στόχος αυτού είναι η εν τέλει μετατροπή σε των ειδικών εντολών σε συνθέσιμη VHDL.
- Μετατροπή σε XML αναπαράσταση τύπου GGX [AGG] προκειμένου την εφαρμογή αλγεβραϊκών μετασχηματισμών γράφου (οι κανόνες μετασχηματισμού υποτίθεται ότι συντάσσονται από τον σχεδιαστή πάλι σε διαμόρφωση GGX) σε CFGs, BBs και CIs.

Σημειωτέον ότι εξετάζεται η προσθήκη και άλλων backends προκειμένου την περαιτέρω αύξηση των δυνατοτήτων του YARDstick. Η επέκταση του YARDstick (πέρα από τις ήδη πολλές δυνατότητες που έχει) είναι ένα έργο που απαιτεί καθημερινή ενημέρωση γύρω από τα (δωρεάν) εργαλεία ανοικτού κώδικα με τα οποία θα μπορούσε να συνεργαστεί και έχει προοπτική σε βάθος χρόνου. Γενικά, πάντα εξετάζεται η προσθήκη υποστήριξης για

- Άλλους (νέους) μεταγλωττιστές με επιθυμητά χαρακτηριστικά/δυνατότητες.
- Επιπλέον διαμορφώσεις που επιτρέπουν τη σύνθεση λειτουργικών μονάδων σε VHDL από τα αντίστοιχα εργαλεία.

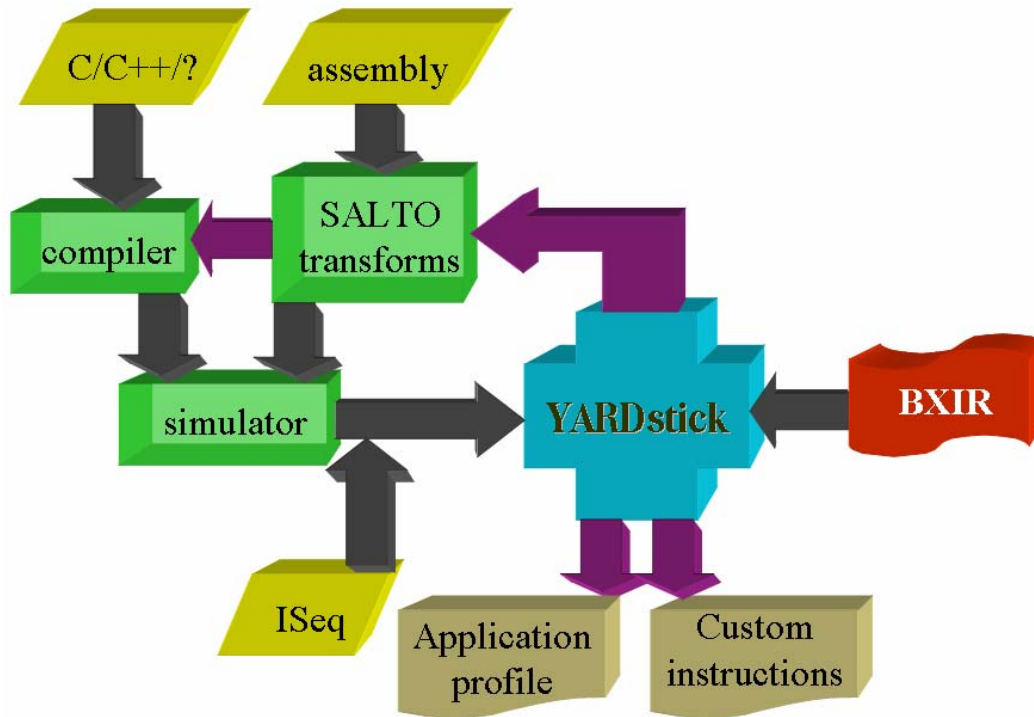
4.2.2. Δομή ενός περιβάλλοντος YARDstick

Ο πυρήνας του YARDstick μπορεί να χρησιμοποιηθεί ως υποδομή για την ανάλυση εφαρμογών και τη διερεύνηση προκειμένου την ανάδειξη ειδικών επεκτάσεων υλικού (ASHEs). Το Σχ. 4.2 δείχνει το πλαίσιο εργασίας που χρησιμοποιεί το YARDstick εντός του οποίου αξιοποιούνται και εργαλεία μεταγλώττισης και προσομοίωσης τρίτων. Το frontend στο επίπεδο μεταγλωττιστή (π.χ. GCC [GCC], COINS [COINS], LANCE [LANCE], SUIF/Machine-SUIF [SUIF],[MachSUIF]) δέχεται ως είσοδο κώδικα εφαρμογών σε γλώσσα υψηλού επιπέδου όπως C/C++ ενώ αυτό θα μπορούσε να συμβαίνει για άλλες HLL γλώσσες που ενδεχόμενα να ήταν του ενδιαφέροντος (όπως υποσύνολα της C, και άλλες γλώσσες με περιορισμένη εμβέλεια εφαρμοστικότητας). Ο κώδικας της εφαρμογής εισόδου μεταγλωττίζεται σε IR χαμηλού επιπέδου η οποία μπορεί να απεικονιστεί σε κείμενο ως μια μορφής συμβολικής assembly έπειτα από την επεξεργασία από το frontend, μετατροπή στο εσωτερικό IR, εφαρμογή βελτιστοποιήσεων που είναι ανεξάρτητες από την αρχιτεκτονική και από ένα σύνολο διεργασιών του μεταγλωττιστή στο επίπεδο του backend με μόνη την επιλογή κώδικα να πρέπει να έχει ήδη εφαρμοστεί υποχρεωτικά.

Στη συνέχεια, ο παραγώμενος κώδικας επιπέδου IR μπορεί να μακρο-επεκταθεί, να εισαχθούν υπορουτίνες εξαγωγής μετρικών για την λήψη προφίλ και να μετατραπεί σε διαμόρφωση ISeq από κατάλληλο πέρασμα που έχει συνταχθεί για την υποδομή εργαλείων μετασχηματισμού κώδικα συμβολομεταφραστή SALTO [SALTO]. Αυτή η ροή προϋποθέτει την υποστήριξη της στοχευόμενης αρχιτεκτονικής από αντίστοιχη backend βιβλιοθήκη για την υποδομή SALTO. Ο κώδικας συμβολομεταφραστή (είτε είναι έξοδος από πέρασμα του SALTO είτε όχι) είναι συμβατός με τα γνωστά εργαλεία ανάπτυξης για τη συμβολομετάφραση και σύνδεση (π.χ. binutils, παρεμφερή εργαλεία) και τα παραγώμενα εκτελέσιμα αρχεία² μπορούν να εκτελεστούν σε προσομοιωτή ακρίβειας κύκλου (π.χ. ArchC [ArchC]). Εναλλακτικά, αρχεία ISeq θα μπορούσαν να παραχθούν απευθείας από κατάλληλα τροποποιημένο μεταγλωττιστή για την αρχιτεκτονική-στόχο. Αυτή είναι και η περίπτωση που έχει βρει περισσότερο χρήση σε συνεργασία με το YARDstick. Πιο συγκεκριμένα γίνεται

² Για τις ενσωματωμένες αρχιτεκτονικές στόχους της μεθοδολογίας χρησιμοποιείται η διαμόρφωση ELF.

χρήση μιας τροποποιημένης (από τον ερευνητή Α.Π.Θ. Νικόλαο Καβαδιά) έκδοσης του Machine-SUIF 2.02.07.15 [MachSUIF] για την οποία το προφίλ εκτέλεσης βασικών μπλοκ λαμβάνεται με πλήρως αυτόματο τρόπο με μετατροπή του IR σε περιγραφή σε υποσύνολο της C και εκτέλεση του C κώδικα χαμηλού επιπέδου (δείκτες ενός επιπέδου, αναθέσεις τριών διευθύνσεων που είναι αναπαράσταση γνωστή ως three-address code ή quadruples) στο ίδιο το μηχάνημα-ξενιστή (MS Windows-Linux/x86).



Σχήμα 4.2: Αποψη υψηλού επιπέδου ενός πλαισίου εργασίας που βασίζεται στο YARDstick.

Στο σύνορο της επικοινωνίας με τον προσομοιωτή, το YARDstick αναμένει πληροφορία για το δυναμικό προφίλ της εφαρμογής (συχνότητες εκτέλεσης βασικών μπλοκ, ίχνος προγράμματος, στατιστικά προσπελάσεων κρυφών μνημών) πάνω στην στοχευόμενη αρχιτεκτονική. Στα πλαίσια του YARDstick, τα στατικά και δυναμικά μετρικά των εφαρμογών μπορούν να εκτιμηθούν, να αξιολογηθούν και να οπτικοποιηθούν. Ένας αναλυτής εφαρμογών (εργαλείο **iseqtool**) και ένας γεννήτορας/επιλογέας ειδικών εντολών (**igensel**) που συνδέονται με τις βιβλιοθήκες **libByoX** και **libPatCUTE**, χρησιμοποιούνται για την λήψη του τελικού προφίλ των στοχευόμενων εφαρμογών και των ειδικών εντολών, αντίστοιχα.

4.2.3. Χρήση του YARDstick API

Το API του YARDstick παρέχει μεθόδους για τον χειρισμό των οντοτήτων ISeq και την εξαγωγή χρήσιμης πληροφορίας για εσωτερικές δομές δεδομένων όπως λίστες τοπικών ορισμάτων, ανάλυση επιπέδου λειτουργίας (π.χ. αναζήτηση κόμβων εντολών μηδενικού-προηγμένου/διάδοχου) και την εφαρμογή επιμέρους διεργασιών ενός backend. Το Σχ. 4.3 δείχνει ένα παράδειγμα χρήσης του API για την ενημέρωση των σχετιζόμενων δομών δεδομένων προκειμένου την γέννηση ειδικών εντολών σε επίπεδο βασικού μπλοκ.

```
void evaluate_bb_ci (ISeq bb)
{
    ...
    // Setup operand list
    UIOCList Lopnd = init_opnd_library ();

    // Find unique i/o registers and constants
    find_input_opnds (bb, Lopnd, input_opnds);
    find_output_opnds (bb, Lopnd, output_opnds,
instr_has_successor);
    find_cnst_opnds (bb, Lopnd, cnst_opnds);

    if (unique i/o instances for operands/constants)
        collapse_to_unique_opnds ();

    clear_best_cut ();

    // CI generation for the BB
    if (MIMO method)
        MIMO_identification (bb);
    else if (MaxMISO or constrained MISO method)
        MAXMISO_identification (bb);
}
```

Σχήμα 4.3: Ενημέρωση των εσωτερικών δομών δεδομένων για την γέννηση ειδικών εντολών σε επίπεδο BB.

Σε μεγαλύτερη λεπτομέρεια, ένα βασικό μπλοκ στο επίπεδο ISeq συμβολίζεται με 'bb'. Πρώτα, η ρουτίνα 'init_opnd_library' αρχικοποιεί μια κενή λίστα ορισμάτων, ονομαζόμενη Lopnd η οποία ενημερώνεται με κλήσεις στις συναρτήσεις 'find_<type>_opnds', όπου με <type> συμβολίζεται ο τύπος ορίσματος που μπορεί να είναι ένας από input, output, cnst (σταθερά). Όταν είναι ενεργοποιημένη η επιλογή για μοναδική αναπαράσταση ορισμάτων καταχωρητή και σταθεράς, με εφαρμογή της ρουτίνας "collapse_to_unique_opnds" τα ορίσματα θεωρούνται ότι διαθέτουν μονή αναπαράσταση ανά υπολίστε εισόδου και εξόδου. Μετά τον καθαρισμό της ενδιάμεσης αποθήκευσης του βέλτιστου μοτίβου τομής (best cut) που πρόκειται να αναγνωριστεί κατά τη συγκεκριμένη (τρέχουσα) επανάληψη (ή αναδρομή) του αλγορίθμου γέννησης CI, μπορεί να επιλεγεί μέθοδος είτε οργανωμένη για την αναγνώριση MIMO είτε MISO για την εκτέλεση της κάθε αυτό διαδικασίας.

4.3. Περιπτώσεις διερεύνησης πεδίου λύσεων με το YARDstick

Χάρην αποδείξεως της εφαρμοσιμότητας του YARDstick, έγινε χρήση του κάτω από διάφορα σενάρια που αντικατοπτρίζουν ρεαλιστικά προβλήματα στην διερεύνηση και αποτίμηση του πεδίου σχεδιασμού όταν γίνεται ανάπτυξη νέων ASIP ή εμπλουτισμός τροποποιήσιμων αρχιτεκτονικών ειδικού σκοπού. Για τις περιπτώσεις μελέτης αυτής της ενότητας χρησιμοποιήθηκαν τρεις διαφορετικές αρχιτεκτονικές στόχοι:

- 1) Το SUIFvm IR [MachSUIF] ενισχυμένο από ένα σύνολο αυξητικών επεκτάσεων σε αυτό, με την προκύπτουσα αρχιτεκτονική να καλείται SUIFvmenh.
- 2) Η αρχιτεκτονική SUIFrmeh (SUIF "real machine" enhanced) η οποία υποστηρίζεται από backend για τον μεταγλωττιστή Machine-SUIF το οποίο συντάχθηκε από τον ερευνητή Α.Π.Θ., και η οποία εισάγει πεπερασμένο σύνολο καταχωρητών με ρυθμιζόμενο αριθμό (12, 16, 32 ή 64) στην αρχιτεκτονική SUIFvmenh. Επίσης, η SUIFrmeh επιλύει τις λειτουργίες ανάθεσης τύπου (type casting operations) χαρτογραφώντας τις από την εντολή cvt στις κατάλληλες εντολές που είναι δεκτές από το backend της SUIFrmeh: επέκταση μηδενικού (zero-extension: zxt), επέκταση προσήμου (sign-extension: sxt), αποκοπή (truncation: trunc) και μεταφορά (move:

μον) οι οποίες δηλώνουν τους τύπους δεδομένων πηγής και προορισμού.

- 3) Υποσύνολο iDLX του συνόλου εντολών αέρας αριθμητικής της αρχιτεκτονικής DLX για το οποίο ο κώδικας συμβολομεταφραστή μπορεί να θεωρηθεί ως ένα είδος IR.

Τα χαρακτηριστικά των αρχιτεκτονικών IR που επιλέχθηκαν συνοψίζονται στον Πίνακα 4.1. Στα πειράματα των ακόλουθων υποενοτήτων, όλες οι εντολές μεταφοράς ελέγχου (*beqz*, *bnez*, *j*, *jr*, *jal*, *jalr*) έχουν αποκλειστεί από την συμπερίληψή τους σε ειδικές εντολές για το iDLX IR, ενώ για τις αρχιτεκτονικές της οικογένειας SUIF, οι εντολές διακλάδωσης επιτρέπονται, αντίστοιχα. Τα δύο διαφορετικά σύνολα περιορισμών για τον αποκλεισμό λειτουργιών επιλέχθηκαν ώστε να τονιστούν διαφορετικές ενδεχόμενες απαιτήσεις στη διερεύνηση πεδίου λύσεων και δεν θεωρούνται άμεσα συγκρίσιμα. Το IR που βασίζεται στην αρχιτεκτονική DLX θα ήταν πιθανή επιλογή όταν το αντικείμενο είναι η βελτιστοποίηση προϋπάρχοντος κώδικα συμβολομεταφραστή (ή μηχανής) για έναν επεξεργαστή DLX. Αντίθετα, η SUIFvmenh υλοποιεί ένα αντιπροσωπευτικό IR για RISC επεξεργαστές χωρίς να περιορίζεται περαιτέρω το αρχιτεκτονικό περίγραμμα αυτών των RISC, το οποίο είναι καταλληλότερο για την ανάπτυξη ASIP από μηδενικής βάσης.

Πίνακας 4.1: Ρυθμίσεις για τα διαφορετικά IR που χρησιμοποιήθηκαν στη διαδικασία γέννησης ειδικών εντολών.

IR	Operations
SUIFvmenh	SUIFvm plus: type conversion (<i>sxt</i> , <i>zxt</i>), partial predication (<i>select</i>), bit manipulation (<i>bitinsert</i> , <i>bitextract</i> , <i>concat</i>)
SUIFrmeh	SUIFvmenh with finite register set (12, 16, 32 or 64 registers); here 32 is used
iDLX	The DLX integer instruction set

Για τα πειράματα χρησιμοποιήθηκαν εφαρμογές δοκιμής από ένα σύνολο ενσωματωμένων εφαρμογών δοκιμής επιδόσεων που αποτελείται από 5 κρυπτογραφικές (*crc32*, *deraiden*, *enraiden*, *idea*, *sha*) και 5 εφαρμογές που βρίσκουν χρήση στα πολυμέσα (*adpcm_dec*, *adpcm_enc*, *fir*, *fsme*, *mc*) οι οποίες παρουσιάζονται στον Πίνακα 4.2.

Πίνακας 4.2: Σύνοψη των εφαρμογών δοκιμής που εξετάστηκαν.

Benchmark	Description
<i>crc32</i>	Cyclic redundancy check
<i>deraiden</i>	Decoding raiden cipher
<i>enraiden</i>	Encoding raiden cipher
<i>idea</i>	IDEA cryptographic kernel
<i>sha</i>	Secure Hash Algorithm producing an 160-bit message digest for a given input
<i>adpcmdec</i>	Adaptive Differential Pulse Code Modulation (ADPCM) decoder
<i>adpcmenc</i>	Adaptive Differential Pulse Code Modulation (ADPCM) encoder
<i>fir</i>	FIR filter
<i>fsme</i>	Full-search motion estimation
<i>mc</i>	Motion compensation

4.3.1. Εισαγωγικό παράδειγμα εξαγωγής ειδικής εντολής: Ανίχνευση ακμής με απλή παράγωγο

Το Σχήμα 4.4 παρουσιάζει το κύριο μέρος απλού αλγορίθμου ανίχνευσης ακμών με χρήση παραγώγου (gradient-based edge detector). Πρώτα, οι διπλά φωλιασμένοι βρόχοι (που δεικτοδοτούνται από τις μεταβλητές x , y) σκανάρουν τα εικονοστοιχεία (pixel) της εικόνας εισόδου "source" (διαστάσεων 16x16) από αριστερά προς τα δεξιά και από πάνω προς τα κάτω. Από την source επιλέγεται το εικονοστοιχείο $p1$ με διεύθυνση $p1_addr$. Το $p1$ στη συνέχεια συγκρίνεται με δύο γειτονικά του εικονοστοιχεία: το $p2a$ (βρίσκεται στην ίδια στήλη, αλλά στην αμέσως προηγούμενη σειρά) και το $p2b$ (αμέσως προηγούμενο στην ίδια σειρά). Στη συνέχεια συγκρίνεται το $p1$ με τα $p2a$ και $p2b$ για την ανίχνευση οριζόντιων και κάθετων ακμών, αντίστοιχα. Εφόσον, υπολογίζεται απόλυτη τιμή διαφοράς μεγαλύτερη από μία τιμή κατωφλίου (threshold), είτε στην περίπτωση κάθετης είτε οριζόντιας ακμής, η διεύθυνση του εικονοστοιχείου $p1$ επισημαίνεται ως διεύθυνση που βρίσκεται πάνω σε ακμή.

```
for (x=0; x<NSIZE; x++) /* Scan the Image Pixel by pixel */
{
  for (y=0; y<MSIZE; y++)
  {
    /* An edge is found if the difference between the value of the
    * pixel and the pixel to its left or the pixel on top of it is
    * greater than a given threshold value
    */
    p1_addr = x*MSIZE+y;
    p2a_addr = p1_addr - MSIZE;
    p2b_addr = p1_addr - 1;

    p1 = source[p1_addr];
    if (p2a_addr < 0)
      p2a = 0;
    else
      p2a = source[p2a_addr];
    if (p2b_addr < 0)
      p2b = 0;
    else
      p2b = source[p2b_addr];

    if (iabs(p1 - p2a)>threshold || /* Detect a horizontal edge */
        iabs(p1 - p2b)>threshold) /* Detect a vertical edge */
    {
      /* If an edge is found mark it as a black pixel in the output */
      dest[p1_addr] = N;
    }
    else
    {
      /* If no edge is found mark it as a white pixel in the ouput */
      dest[p1_addr] = W;
    }
  }
}
```

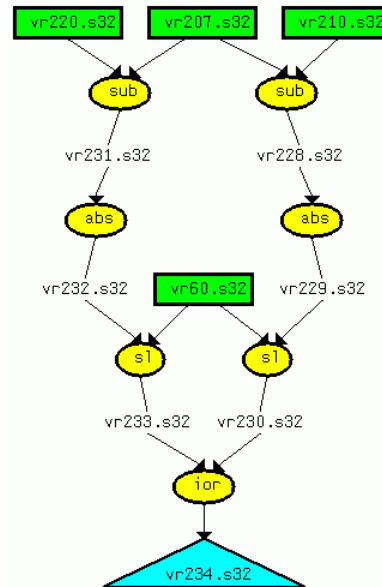
Σχήμα 4.4: Τμήμα του κώδικα της εφαρμογής ανιχνευτή ακμών.

Για περιορισμό ορισμάτων εισόδου/εξόδου (Ni/No) 4/1 και επίτρεψη συμπερίληψης σε CIs όλων των εντολών εκτός των: {bfalse, btrue, beq, bne, ble, blt, bge, bgt, jmp, jmpri, cal, ret} μία από τις 4 αναδεικνυόμενες ειδικές εντολές χρησιμοποιώντας τον αλγόριθμο εύρεσης MIMO είναι και η σύνθετη λειτουργία της ανίχνευσης κάθετης και οριζόντιας ακμής. Στο επίπεδο του πηγαίου κώδικα, το αντίστοιχο τμήμα δίνεται από την έκφραση

```
(iabs(p1 - p2a)>threshold || iabs(p1 - p2b)>threshold)
```

όπου iabs η λειτουργία υπολογισμού απόλυτης τιμής ακεραίων ενώ η απεικόνιση VCG όπως

παράγεται από το YARDstick δίνεται στο Σχ. 4.5. Η ειδική εντολή αυτή μπορεί να ονομαστεί DETEDGS (detect edges) και να υλοποιηθεί ως ΕΛΜ πέντε ορισμάτων. Η εντολή θα πρέπει να δέχεται ως ορίσματα ανάγνωσης τους καταχωρητές που αντιστοιχούν στα r1, r2a, r2b, threshold και ως όρισμα εγγραφής, τέταρτο καταχωρητή για την αποθήκευση του αποτελέσματος της έκφρασης.



Σχήμα 4.5: Διάταξη γράφου από αρχείο VCG για την ειδική εντολή DETEDGS.

Στην προκειμένη περίπτωση, η εξαγωγή του αρχείου ISeq έγινε με το πέρασμα bbrpart (Π2.1) για την αρχιτεκτονική SUIFvm. Η ειδική εντολή DETEDGS είναι μονού κύκλου, μειώνει τον αριθμό κύκλων εκτέλεσης κατά 35% και επιφάνεια υλικού ίση με 1.23 ενός 32x32-bit πολλαπλασιαστή.

4.3.2. Αποτίμηση των επιδράσεων του μεταγλωττιστή: Περίπτωση μελέτης για πυρήνες εφαρμογών πολυμέσων

Έχει πρόσφατα καταγραφεί [Bon06] ότι οι παραδοσιακοί μετασχηματισμοί βελτιστοποίησης των μεταγλωττιστών σε συνδυασμό με την τετριμμένη επιλογή του να εφαρμόζεται η αναγνώριση ειδικών εντολών μετά το πέρας του συνόλου των περασμάτων βελτιστοποίησης, δεν αποδίδουν απαραίτητα τις βέλτιστες επιδόσεις όταν στοχεύουν επεξεργαστές ειδικού σκοπού. Αντιθέτως, στην περίπτωση των ΕΕΣ χρειάζονται εναρμονισμένοι μετασχηματισμοί στο επίπεδο του IR αλλά και στο επίπεδο πηγαίου κώδικα (source-level) για την ανάδειξη των ωφέλιμων ASHs. Σε αυτή την υποενότητα, τονίζεται η επίδραση των επιλογών όσον αφορά τις επιμέρους διαδικασίες στις φάσεις μεταγλώττισης με αφορμή δημοφιλείς περιπτώσεις εφαρμογών μελέτης: ένας κωδικοποιητής/αποκωδικοποιητής ADPCM, ένα FIR φίλτρο, και τυπικές υλοποιήσεις σε ενσωματωμένο λογισμικό, βασικών αλγορίθμων εκτίμησης και αντιστάθμισης κίνησης. Ιδιαίτερα, γίνεται διερεύνηση για συγκεκριμένες επιδράσεις που οφείλονται στον μεταγλωττιστή όταν αυτός χρησιμοποιείται για την ανεύρεση ειδικών εντολών:

- Η επίδραση της κατανομής καταχωρητών στην ποιότητα των παραγόμενων CIs. Για τον σκοπό αυτό, στοχεύεται η αρχιτεκτονική SUIFrmeh. Υποθέτουμε ένα αρχείο καταχωρητών 32 θέσεων, η σύμβαση κλήσης ρουτίνων (procedure calling convention) είναι η ίδια με αυτή που χρησιμοποιήθηκε για τον μεταγλωττιστή GCC

(για τον επεξεργαστή DLX). Το backend αυτό σχεδιάστηκε από τον ερευνητή Α.Π.Θ..

- b) Την καταλληλότητα χρήσης ενός μεταγλωττιστή υψηλής βελτιστοποίησης όπως είναι ο GCC, ο οποίος έχει οργανωθεί για την υποστήριξη RISC επεξεργαστών γενικού σκοπού, για τον DLX που είναι η περίπτωση εδώ, σε σχέση με ένα γνωστό ερευνητικό μεταγλωττιστή (MachSUIF που στοχεύει SUIFvm). Ο τελευταίος έχει χρησιμοποιηθεί εκτενώς για την διερεύνηση του πεδίου μετασχηματισμών ώστε την ανάδειξη νέων CIs.

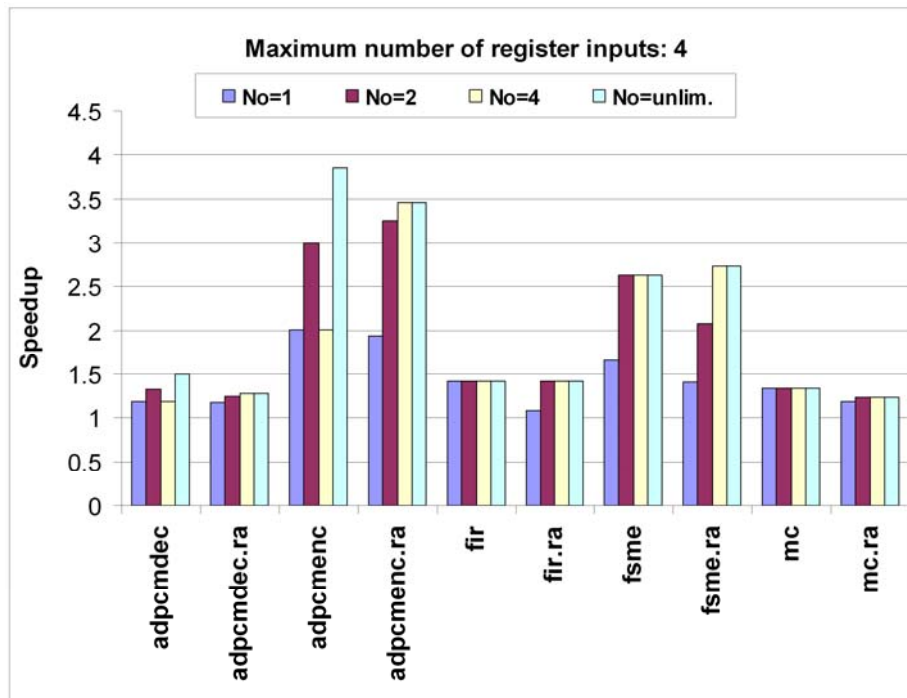
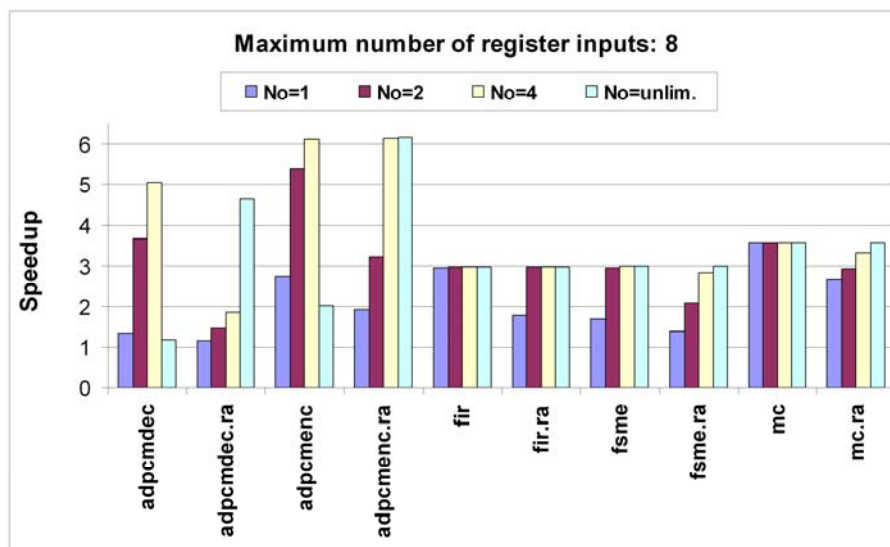
Προκειμένου να λαμβάνονται υπ' όψη μόνο οι πραγματικές εξαρτήσεις δεδομένων ανάμεσα στις στοιχειώδεις λειτουργίες, είναι απαραίτητο να απομακρυνθούν όλες οι λανθάνουσες εξαρτήσεις που προκύπτουν όταν σκανάρεται η λίστα λειτουργιών σε γραμμικό χρόνο. Αυτό επιτυγχάνεται με ένα απλό πέρασμα επιπέδου IR (για παράδειγμα, τέτοιο πέρασμα έχει υλοποιηθεί για τον μεταγλωττιστή Machine-SUIF και την αρχιτεκτονική SUIFvmenh) το οποίο συμπεριλαμβάνει την χρήση του ψευδοκώδικα του Σχ. 4.6. Ο αλγόριθμος του Σχ. 4.6 μπορεί να χρησιμοποιηθεί για δρομολόγηση εντολών σε σειρά (in-order), ήτοι δεν επιτρέπονται οπισθόδρομες εξαρτήσεις δεδομένων εντός των βασικών μπλοκ. Για δοθέν σύνολο ακμών εξαρτήσεων $\bigcup \{(i \rightarrow j)_k\}$ ανάμεσα στις εντολές $mi(i)$, $mi(j)$ με αύξοντες αριθμούς i , j αντίστοιχα, θεωρείται το διάστημα $[i, j]$. Τα ορίσματα προορισμού των εντολών που είναι εντός του διαστήματος συγκρίνονται διαδοχικά με το όρισμα (opnd) για το οποίο θέλουμε να απομακρύνουμε όλες τις λανθάνουσες εξαρτήσεις, έχοντας το ίδιο το opnd ως το έντελο εξάρτησης. Εάν το opnd γράφεται (ορίζεται) τουλάχιστον μία φορά, τότε αναγνωρίζεται μία λανθάνουσα εξάρτηση (και σημειώνεται ως TRUE) και αναιρείται η αντίστοιχη ακμή εξάρτησης δεδομένων.

```
boolean is_false_dependency(BB* bb, InstrID mi_lpos,
                           mi_hpos, LOpnd opnd)
{
    boolean false_dependency_f = FALSE;
    ...
    // Iterate through the [mi_lpos..mi_hpos] range
    foreach machine instruction (mi) in range do
        if the current mi is within the specified range
            get destination operand dstop of mi
            if dstop is ((a base register or address symbol)
                and writes memory)
                if dstop is equal to opnd
                    // a false dependency has been found
                    false_dependency_f |= TRUE;
            fi
        fi
    od
    return false_dependency_f;
}
```

Σχήμα 4.6: Διαδικασία απομάκρυνσης των λανθανουσών ακμών εξάρτησης δεδομένων από βασικά μπλοκ.

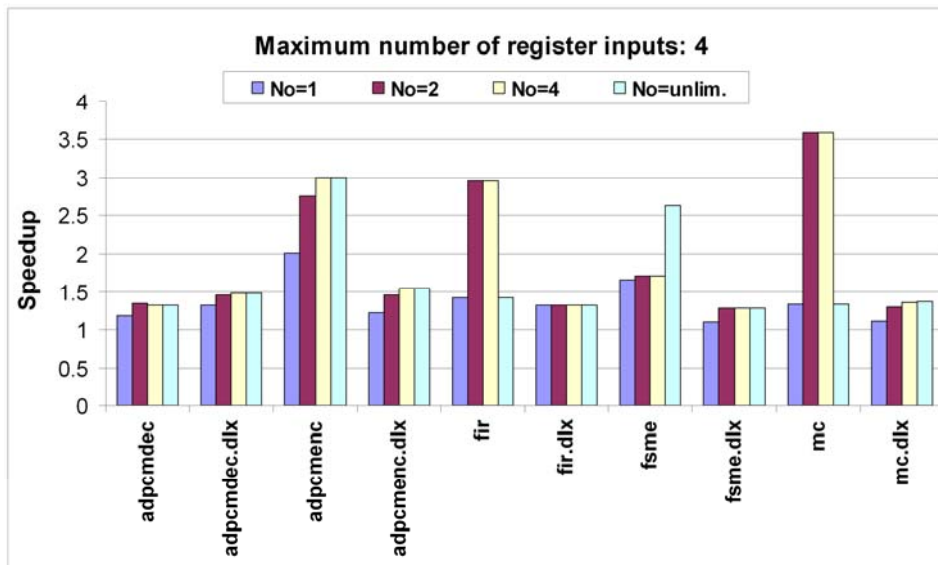
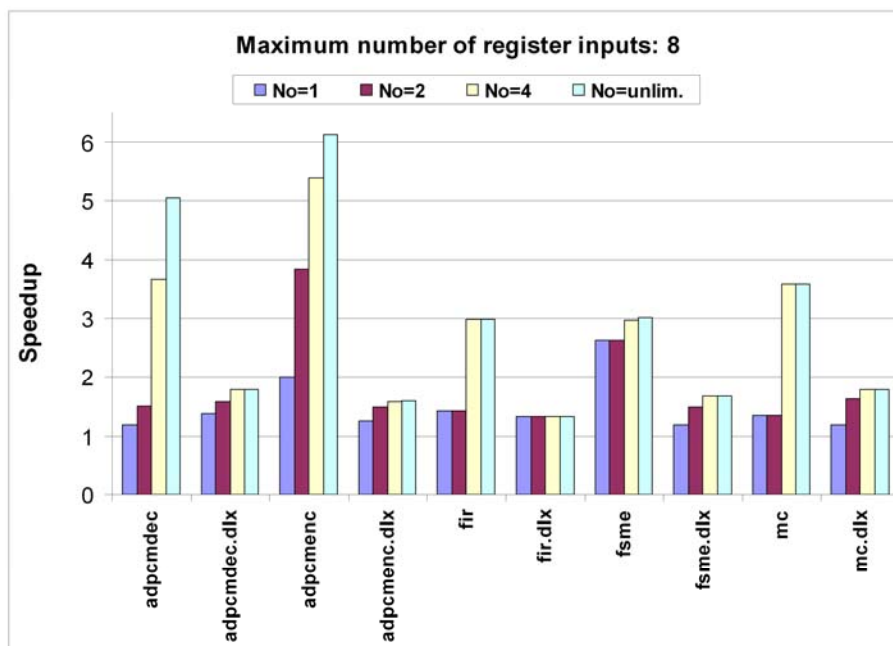
Η επιτάχυνση των εφαρμογών που επιτυγχάνεται προ και μετά της κατανομής καταχωρητών (το τελευταίο σημειώνεται με το επίθεμα 'ra' στο όνομα της εφαρμογής δοκιμής) δίνεται στο Σχ. 4.7. Σε αντίθεση με ότι πιστεύεται κοινώς [Cla03], [Cas04], η εισαγωγή ενός πεπερασμένου συνόλου καταχωρητών και η χαρτογράφηση των προσωρινών μεταβλητών της επιλογής κώδικα (code/instruction selection) στο σύνολο αυτό, δεν έχει πάντοτε αρνητική επίπτωση στις επιτυγχάνόμενες επιταχύνσεις για τις εφαρμογές υπό μελέτη. Ενώ είναι ευκρινές ότι υφίσταται μετρήσιμη επίδραση (επιβάρυνση του 22.15%) λόγω της κατανομής καταχωρητών για εντολές μονής εξόδου (No = 1), η έκταση αυτής της επιβάρυνσης μειώνεται για μεγαλύτερο αριθμό ορισμάτων εξόδου (για επιλογή κώδικα με ειδικές εντολές που έχουν

αυτή την ιδιότητα). Έτσι, για $No = \{2, 4, \infty\}$, οι αντίστοιχες επιβαρύνσεις υπολογίζονται με τη βοήθεια του YARDstick σε 17%, 2.5% και -21.3%, με το τελευταίο να σημαίνει ότι IR στο οποίο έχει επιτελεστεί κατανομή καταχωρητών επιτρέπει μεγαλύτερες επιταχύνσεις εφαρμογής σε σχέση με IR που λαμβάνεται πριν την κατανομή καταχωρητών στην περίπτωση απεριόριστου αριθμού ορισμάτων εξόδου. Το σημαντικό αυτό συμπέρασμα υπονοεί ότι η επιβάρυνση λόγω εγχύσεων (spills) και γεμισμάτων (fills) που προκύπτουν λόγω της πίεσης καταχωρητών μπορεί να αντιμετωπιστεί επαρκώς όταν χρησιμοποιούνται εντολές MIMO στις εκτιμήσεις. Επιπρόσθετα, οι ειδικές εντολές έχουν την παράπλευρη επίδραση της εξουδετέρωσης της ανάγκης για ορισμένες προσωρινές μεταβλητές εντός ενός CI υπογράφου, δοθέντος ότι αυτές δεν χρειάζεται να επιβιώσουν εκτός του υπογράφου.

(α) $N_i = 4$ (β) $N_i = 8$

Σχήμα 4.7: Επίδραση της κατανομής καταχωρητών στην επιτάχυνση εφαρμογής για διαφορετικούς αριθμούς ορισμάτων καταχωρητή εισόδου/εξόδου.

Το Σχ. 4.8 απεικονίζει τα αποτελέσματα για τις σχετικές επιταχύνσεις εφαρμογής όσον αφορά διαφορετικούς αριθμούς ορισμάτων καταχωρητή εισόδου/εξόδου για τις δύο επιλεχθείσες αρχιτεκτονικές. Επίσης, τέθηκε και η περίπτωση του απεριόριστου αριθμού εισόδων και τα αντίστοιχα αποτελέσματα ήταν εντός του 0.4% των μετρήσεων για την περίπτωση $N_i = 8$. Η διαφοροποίηση στη μέση επιτάχυνση για τους δοθέντες αριθμούς εισόδων για την ίδια εφαρμογή είναι περίπου 44% (κυμαινόμενη από 20% ως 61%). Αυτό οφείλεται μερικώς στο γεγονός ότι κατανομή των ορισμάτων στοίβας που εφαρμόζεται μόνο για τον iDLX προσθέτει εντολές προσπέλασης μνήμης για την αποθήκευση και επαναφορά ορισμάτων συναρτήσεων τα οποία συνήθως δεν συμπεριλαμβάνονται σε νέες CIs. Ακόμη και όταν αναγνωρίζονται MIMO εντολές που συμπεριλαμβάνουν την ακολουθία για αποθήκευση μεταβλητών από καλούμενο (callee-saved sequence) η οποία αποτελεί μια σειρά από εντολές sw, οι λαμβανόμενες επιταχύνσεις περιορίζονται σημαντικά από το εύρος δεδομένων μνήμης που υποτίθεται για τις εκτιμήσεις το οποίο είναι μία θύρα ανάγνωσης/ μία θύρα εγγραφής (1R/1W) για όλες τις αρχιτεκτονικές-στόχους.

(α) $N_i = 4$ (β) $N_i = 8$

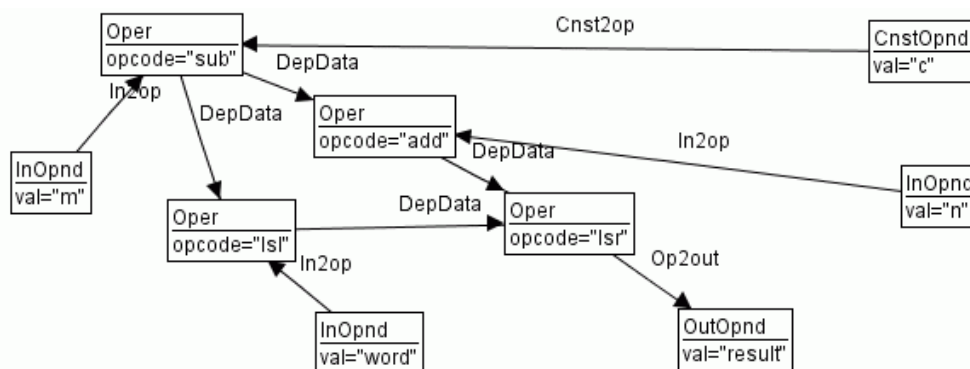
Σχήμα 4.8: Επιτάχυνση εφαρμογών για διαφορετικούς αριθμούς ορισμάτων καταχωρητή εισόδου/εξόδου στις αρχιτεκτονικές SUIFvmenh και IDLX.

4.3.3. Περίπτωση μεταχηματισμού σε καταλληλότερο IR

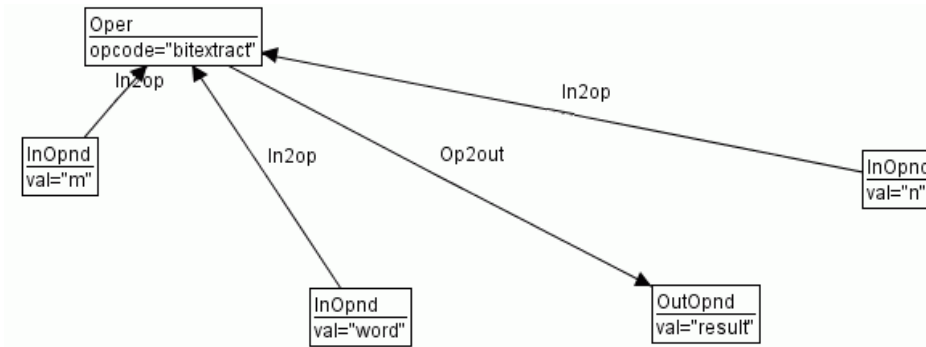
Παρ' όλο που δεν έχει γίνει ρητή αναφορά στη σχετική βιβλιογραφία (πλην της πρόσφατης εξαίρεσης [Scho07] όπου το πρόβλημα εντοπίζεται στο ίδιο πλαίσιο που αναγνώρισε ο ερευνητής Α.Π.Θ.), η επιλογή του IR επηρεάζει σημαντικά την ποιότητα των παραγόμενων ειδικών εντολών. Στο YARDstick, από ISeq μοτίβα μπορούν να παραχθούν αυτόματα οι αναπαράστασεις γράφου για αυτά, συμβατές με το πρότυπο GGX XML [AGG] και να μετασχηματιστούν από AGG κανόνες επανεγγραφής γράφων προκειμένου τη χρήση διαφορετικών IR τελεστών επιτυγχάνοντας την ισοδύναμη λειτουργική συμπεριφορά. Οι περισσότεροι μεταγλωττιστές (με την εξαίρεση του εμπορικού CoSy [ACE]) δεν λαμβάνουν υπ' όψη χειρισμούς επιπέδου bit οι οποίοι είναι επιθυμητοί σε πεδία εφαρμογών όπως είναι οι εφαρμογές δικτύων και οι γενετικοί αλγόριθμοι (GAs). Για να αντιμετωπιστεί η ανεπάρκεια αυτή, ο ερευνητής Α.Π.Θ. όρισε τρεις εξειδικευμένους τελεστές IR ονόματι *bitinsert*, *bitextract*, και *concat* κατά την σημασιολογία του Πίνακα 4.3. Ως κινητήρια παραδείγματα, χρησιμοποιήθηκαν οι γνωστοί τελεστές διασταύρωσης *single-point* (*crdsp*) και *double-point* (*crcdr*), οι οποίοι απαντώνται σε τυπικούς GA όπως είναι ο Απλός Γενετικός Αλγόριθμος (Simple GA: SGA) [Gol89]. Πρέπει να σημειωθεί ότι οι ANSI C υλοποιήσεις των τελεστών διασταύρωσης βελτιστοποιήθηκαν χειρωνακτικά με τροποποιήσεις όπως η μετατροπή όλων των κλήσεων σε συναρτήσεις εντός των *crdsp* και *crcdr* σε μακρο-εισαγωγές. Το Σχ. 4.9 δείχνει το αποτέλεσμα της εφαρμογής ενός κανόνα μετασχηματισμού στο AGG [AGG] για την αντικατάσταση ενός τμήματος του SUIFvmenh IR (Σχ. 4.9.α) από τη χρήση ενός τελεστή *bitextract* όπως φαίνεται από τον λαμβανόμενο γράφο μετά το μετασχηματισμό (Σχ. 4.9.β). Προκειμένου την ανάδειξη της σημασίας της κατάλληλης επιλογής για το IR μεταγλωττιστή, το Σχ. 4.10 σπικιοποιεί τις VCG αναπαραστάσεις ειδικής εντολής που εξάγεται από τον *crdsp*, χωρίς (Σχ. 4.10.α) αλλά και με τη χρήση των τελεστών IR επιπέδου bit (Σχ. 4.10.β).

Πίνακας 4.3: Εξειδικευμένοι IR τελεστές για την βελτίωση της υποστήριξη λειτουργιών επιπέδου bit σε μεταγλωττιστές. r_d , r_s είναι ορίσματα καταχωρητή, $hpos$, $lpos$ δηλώνουν ένα διάστημα bit, και n είναι ο αριθμός ορισμάτων ενός βριαδικού τελεστή.

Operator	Semantics
<i>bitinsert</i>	$r_d[lpos..hpos] \Leftarrow r_s$
<i>bitextract</i>	$r_d \Leftarrow r_s[lpos..hpos]$
<i>concat</i>	$r_d \Leftarrow r_{s(0)} \& r_{s(1)} \& \dots \& r_{s(n-1)}$

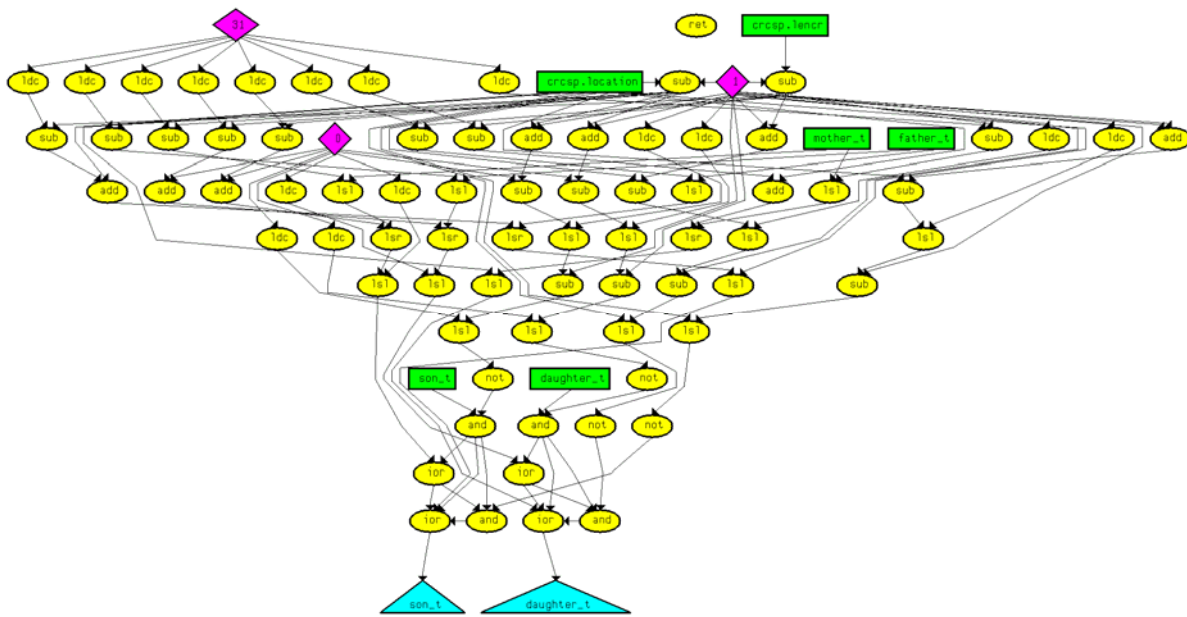


(α)

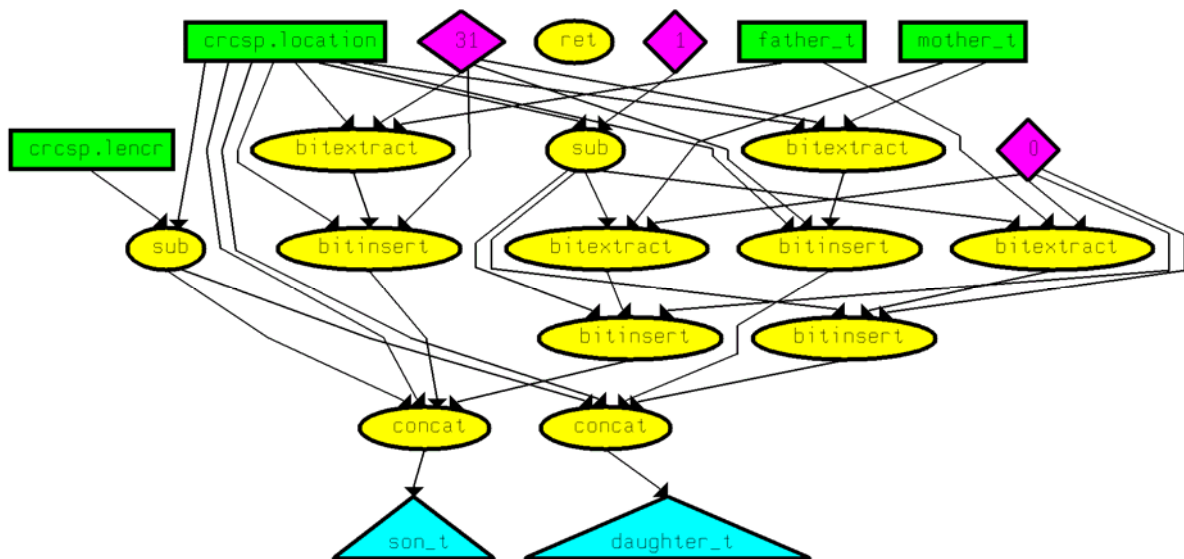


(β)

Σχήμα 4.9: Ένα παράδειγμα επανεγγραφής IR γράφου με κανόνες μετασχηματισμών AGG. (α) Οπτικοποίηση ενός SUIFvmeτη IR γράφου ξενιστή. (β) Ο γράφος που προκύπτει μετά την εφαρμογή κανόνα μετασχηματισμού για τον τελεστή bitextract.



(α)



(β)

Σχήμα 4.10: Οπτικοποίηση του γενετικού τελεστή *crcsp* ως ειδική εντολή για διαφορετικά IR μεταγλωττιστή. (α) Για IR που δεν χρησιμοποιεί λειτουργίες επιπέδου bit. (β) Για IR που χρησιμοποιεί λειτουργίες επιπέδου bit.

Τα κέρδη σε επιδόσεις όταν γίνεται χρήση του υλικού που υλοποιεί τις αυτόματα παραγόμενες CI εξαρτώνται σε μεγάλο βαθμό από το IR στόχο που χρησιμοποιείται για τη αναπαράσταση του κώδικα της εφαρμογής όπως γίνεται εμφανές από τα αποτελέσματα του Πίνακα 4.4. Στον Πίνακα 4.4, οι τρεις πρώτες στήλες είναι αυτο-εξηγούμενες. Η στήλη 'Cycles...' σημειώνει τους κύκλους που απαιτούνται για ακολουθιακή δρομολόγηση του αντίστοιχου GA τελεστή, θεωρώντας ότι γίνεται κατάλληλη χρήση των ειδικών εντολών που παρήχθησαν. Οι δύο τελευταίες στήλες δείχνουν τον αριθμό κύκλων και την επιφάνεια υλικού για την CI. Η απαίτηση επιφάνειας υπολογίζεται σχετικώς με την επιφάνεια (μονάδα επιφάνειας πολλαπλασιαστή ή MAU) ενός 32-bit πολλαπλασιαστή μονού κύκλου που επιστρέφει αποτέλεσμα των 64-bit.

Πίνακας 4.4: Χαρακτηριστικά των CI για χειρωνακτικά βελτιστοποιημένες ANSI C υλοποιήσεις των *crcsp* και *crcdp*.

GA operator	Bit-level operations	CI gen. constraints N_i/N_o	Cycles (seq. schedule)	CI cycles	CI area (MAU)
<i>crcsp</i>	No	4/1	76	-	-
<i>crcsp</i>	No	8/1	41	3	0.977
<i>crcsp</i>	No	8/2	5	3	1.867
<i>crcsp</i>	Yes	4/1	13	-	-
<i>crcsp</i>	Yes	8/1	6	1	0.142
<i>crcsp</i>	Yes	8/2	1	1	0.153
<i>crcdp</i>	No	4/1	111	-	-
<i>crcdp</i>	No	8/1	58	3	1.466
<i>crcdp</i>	No	8/2	5	3	2.800
<i>crcdp</i>	Yes	4/1	18	-	-
<i>crcdp</i>	Yes	8/1	8	1	0.147
<i>crcdp</i>	Yes	8/2	1	1	0.164

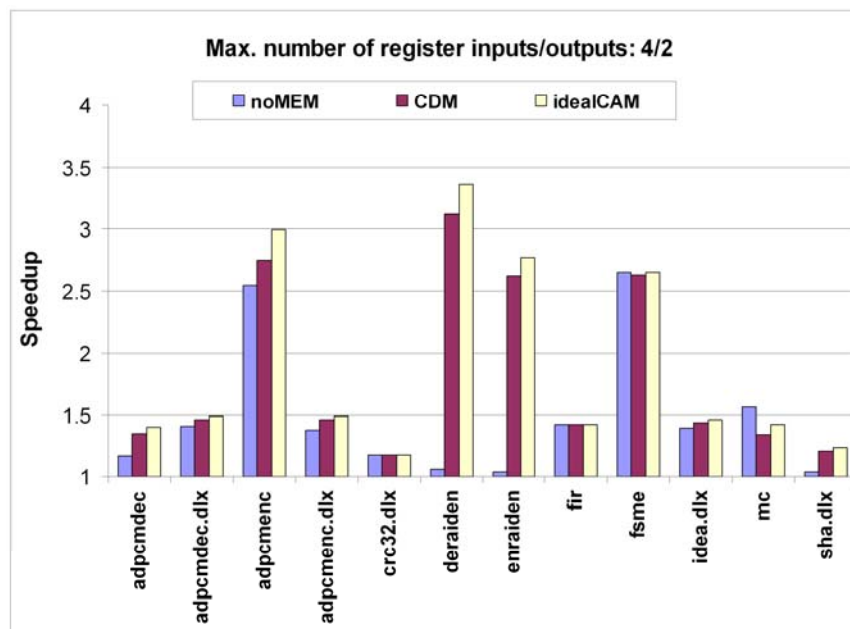
Για τον υπολογισμό δρομολογήσεων με απεριόριστους πόρους, τα παραχθέντα ISeq αρχεία για τις ειδικές εντολές μετατρέπονται αυτόματα μέσω του YARDstick στο φoρμά CDFG [CDFGtool] και εφαρμόζονται σε έναν δρομολογητή ASAP. Εάν δεν χρησιμοποιούνται οι λειτουργίες επιπέδου bit, ο ελάχιστος αριθμός κύκλων για τον τελεστή *crcsp* είναι 76 για ακολουθιακή δρομολόγηση και 12 για απεριόριστους πόρους, ενώ για τον *crcdp* είναι 111 και 14, αντίστοιχα. Με χρήση των λειτουργιών επιπέδου bit, οι ακολουθιακές χρονοδρομολογήσεις πριν την εισαγωγή των ειδικών εντολών απαιτούν 13 και 18 κύκλους για *crcsp* και *crcdp* αντίστοιχα, με ASAP δρομολόγηση των 5 κύκλων και για τα δύο. Στην τελευταία περίπτωση, αναγνωρίζεται μια ειδική εντολή MIMO μονού κύκλου για κάθε γενετικό τελεστή για περιορισμό ορισμάτων $N_i/N_o=8/2$, επίσης με εντυπωσιακά θετικά αποτελέσματα επιφάνειας υλικού.

4.3.4. Επίδραση του μοντέλου προσπελάσεων μνήμης δεδομένων

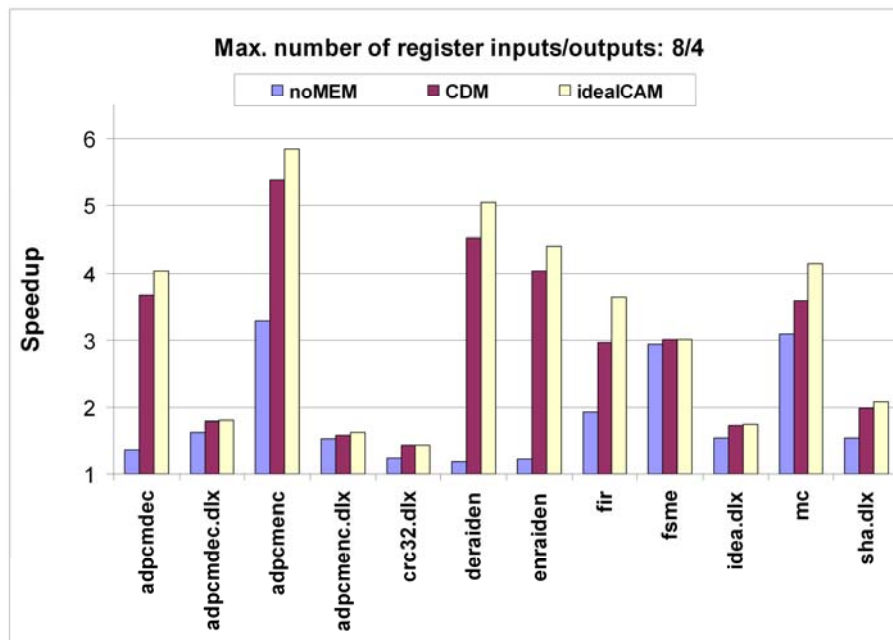
Με χρήση του εργαλείου YARDstick είναι εφικτή η διερεύνηση για το μοντέλο πόρων τοπικής αποθήκευσης για την εξυπηρέτηση των ειδικών λειτουργικών μονάδων (EAM). Η εμβέλεια της χρήσης ειδικών εντολών περιορίζεται από το εύρος bit για την προσπέλαση δεδομένων από την μνήμη δεδομένων και από τα στοιχεία τοπικής αποθήκευσης (αρχείο καταχωρητών) που καθορίζεται από τον αριθμό θυρών εισόδου/εξόδου και την ανάλυση των εξαρτήσεων ανάμεσα σε εντολές φόρτωσης και αποθήκευσης. Σε συγκεκριμένες προσεγγίσεις [Leu06],[Cla05], που ασχολούνται με προκαθορισμένες αρχιτεκτονικές όπως οι MIPS

CorExtend [MIPS] και ARM OptimoDE [ARM], ο περιορισμός αυτός θέτει έναν δεσμευτικό παράγοντα. Παρ' όλα αυτά, για τον σκοπό της διερεύνησης, κατά τη διάρκεια ανάπτυξης ενός ASIP (π.χ. μιας νέας αρχιτεκτονικής χωρίς δεσμεύσεις συμβατότητας) είναι χρήσιμο να εξετάζονται διαφορετικά μοντέλα συνάφειας μνήμης (memory consistency) όπως αυτή ορίζεται στο [Bis07]. Ακολουθώντας τη σημειολογία που εισήχθη στο [Bis07], για τη συνάφεια κατάστασης (state consistency) ανάμεσα στις ΕΛΜ με τοπική αποθήκευση και την μνήμη δεδομένων (π.χ. μια «πρόχειρη» μνήμη δεδομένων στο ολοκληρωμένο), θεωρούμε δύο διακριτά μοντέλα:

- Τη συναφή μνήμη δεδομένων (consistent data memory), όπου η AFU προσπελαύνει άμεσα την μνήμη δεδομένων και δεν υπάρχει ανάγκη για τοπική αποθήκευση στις AFU. Επίσης κάνουμε τη συντηρητική υπόθεση ότι οι εντολές φόρτωσης και αποθήκευσης πρέπει πάντοτε να είναι σε σειρά (serialized).
- Την ιδανική συναφή μνήμη ΕΛΜ (ideal consistent AFU memory), όπου κάθε φόρτωση/αποθήκευση προς/από την μνήμη δεδομένων μετασχηματίζεται σε μια προσπέλαση στην τοπική μνήμη ΕΛΜ. Θεωρούμε ότι η κατάσταση της μνήμης δεδομένων ανανεώνεται με προσπελάσεις DMA που συμβαίνουν παράλληλα ως προς τις εντολές του ΕΕΣ. Για την διερεύνηση της επίδρασης του μοντέλου μνήμης στην επιτάχυνση των εφαρμογών (παρουσία των ειδικών εντολών), πρώτα παρήχθησαν ειδικές εντολές χωρίς επίτρεψη για την συμπερίληψη τοπικής μνήμης ("noMEM"), έπειτα επιτράπηκε η προσθήκη τοπικής μνήμης και πραγματοποιήθηκαν εκτιμήσεις για το μοντέλο συναφούς μνήμης δεδομένων ("CDM") και εν συνεχεία υποτέθηκε το μοντέλο ιδανικής συναφούς μνήμης AFU ("idealCAM"). Τα αντίστοιχα αποτελέσματα παρουσιάζονται στο Σχ. 4.11 για ενδεικτικούς συνδυασμούς (Ni/No) για επεξεργαστή ΕΕΣ (με βάση τις αρχιτεκτονικές SUIFvm, DLX) με μονή έκδοση εντολής (single-issue).



(α) Ni/No = 4/2



(β) Ni/No = 8/4

Σχήμα 4.11: Επίδραση των προσπελάσεων στη μνήμη δεδομένων πάνω στην επιτάχυνση εφαρμογής που οφείλεται στις ειδικές εντολές. Οι προσπελάσεις στη μνήμη δεδομένων απαιτούν επιβάρυνση ενός κύκλου μηχανής.

Όπως προκύπτει από τα αποτελέσματα, η συμπερίληψη λειτουργιών φόρτωσης και αποθήκευσης στις παραγόμενες ειδικές εντολές έχει σημαντικά θετική επίδραση: η επιτάχυνση εφαρμογών αυξάνεται κατά 15.5 ως 33.3% για τους δοσμένους περιορισμούς αριθμού ορισμάτων εισόδου/εξόδου. Ιδιαίτερα για την αρχιτεκτονική SUIF_{vm}, οι βελτιώσεις στην επιτάχυνση εφαρμογών είναι γύρω στο 43.4%. Άλλη σημαντική παρατήρηση είναι ότι το μοντέλο συναφούς μνήμης AFU έχει περιορισμένη επίδραση (βελτιώσεις μέχρι 6.3% κατά μέσο όρο και 8.9% για τις εφαρμογές SUIF_{vm} μόνο). Όμως, για μεγαλύτερη επιβάρυνση (σε αριθμό κύκλων) για την προσπέλαση δεδομένων, οι επιταχύνσεις είναι περισσότερο αξιοσημείωτες. Σε δύο άλλα παραδείγματα διερεύνησης, τέθηκε καθυστέρηση ίση με 2 και 5 κύκλους, αντίστοιχα. Εδώ υποτέθηκε ότι η προσπέλαση γίνεται μέσω τοπικού διαύλου (ένας κύκλος διευθυνσιοδότησης ακολουθούμενος είτε από ένα κύκλο δεδομένων μήκους πλήρους λέξης είτε από διαδοχικούς κύκλους προσπέλασης εύρους ενός byte κάθε φορά). Στις περιπτώσεις αυτές υπολογίζονται σημαντικά υψηλότερες επιταχύνσεις. Συγκεκριμένα, η περίπτωση “CDM” αποδίδει καλύτερα από την “noMEM” κατά 34.7% και 49.9% αντίστοιχα, για τις περιπτώσεις των επιβαρύνσεων κατά 2 και 5 κύκλους. Όταν συγκρίνονται μεταξύ τους τα μοντέλα που επιτρέπουν στις λειτουργίες φόρτωσης/αποθήκευσης να είναι μέρος των ειδικών εντολών, οι αντίστοιχες τιμές είναι 7% και 20.9% υπέρ του μοντέλου “idealCAM” για τις δοθείσες επιβαρύνσεις προσπέλασης.

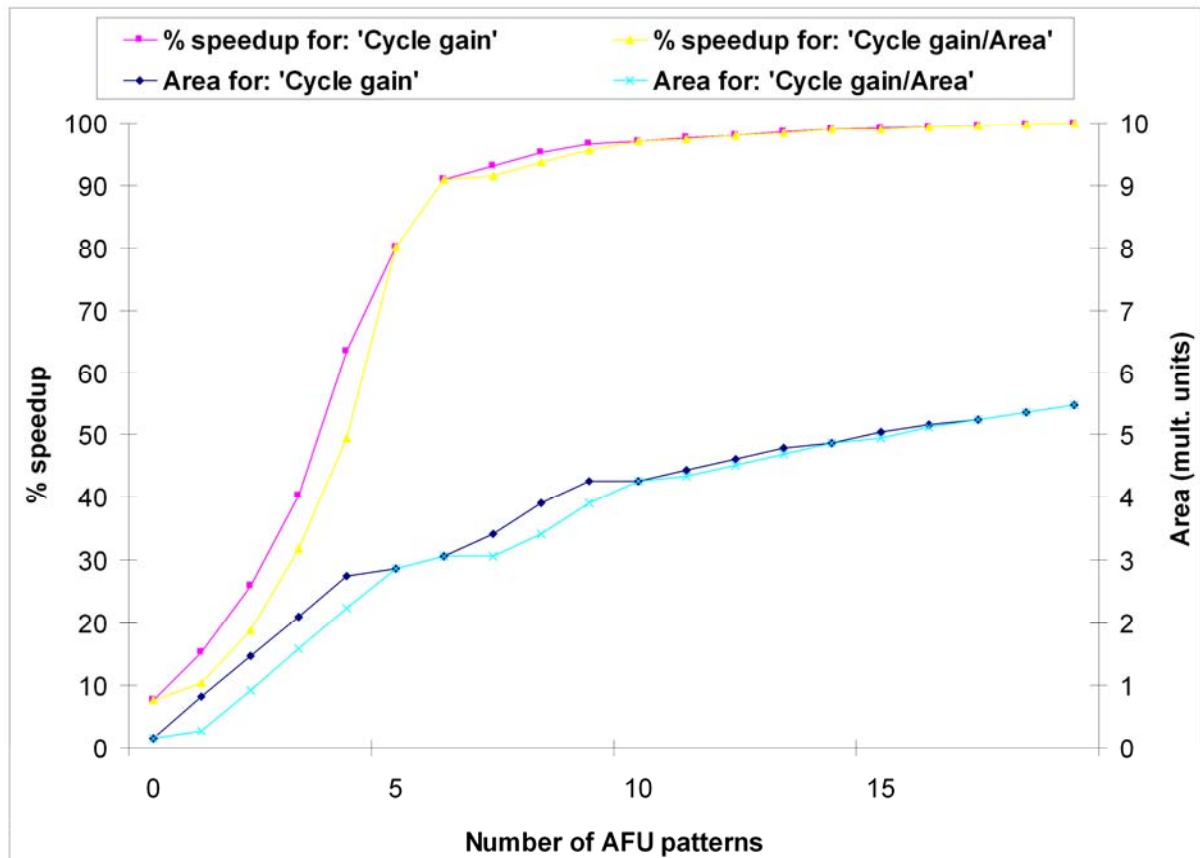
4.3.5. Άπληστη επιλογή ειδικών εντολών υπό μετρικών προτεραιότητας

Πίνακας 4.5: Επιτάχυνση εφαρμογής/κατάληψη επιφάνειας υλικού για ‘Cycle Gain και Cycle Gain/Area’ για περιορισμό ορισμάτων εισόδου/εξόδου Ni/No={8/4}.

Benchmark	0.95 × max. speedup	Area (MAU)	At max. speedup	Area (MAU)
<i>adpcmdec</i>	4/4	0.895/0.895	6	0.983
<i>adpcmdec.dlx</i>	11/11	1.123/1.123	17	1.721
<i>adpcmenc</i>	4/4	0.998/0.998	6	1.086
<i>adpcmenc.dlx</i>	16/16	1.475/1.375	22	2.074
<i>crc32.dlx</i>	3/3	0.12/0.12	3	0.12
<i>deraiden</i>	4/4	2.657/2.657	4	2.657
<i>enraiden</i>	3/3	1.949/1.949	3	1.949
<i>fir</i>	4/4	1.398/1.398	5	1.398
<i>fir.dlx</i>	2/2	0.155/0.155	2	0.155
<i>fsme.dlx</i>	9/9	1.143/1.143	11	1.546
<i>fsme.dlx</i>	6/6	1.03/1.03	10	1.65
<i>idea.dlx</i>	40/50	10.23/9.325	69	13.002
<i>mc</i>	5/5	1.824/1.824	7	2.53
<i>mc.dlx</i>	7/7	1.489/1.489	12	2.516
<i>sha.dlx</i>	7/7	1.671/1.671	20	3.378

Μια σύνοψη των μετρήσεων για το σύνολο των θεωρούμενων εφαρμογών δοκιμής δίνεται στον Πίνακα 4.5. Παίρνοντας την *sha.dlx* για παράδειγμα, παρ' όλο που αναγνωρίζονται δεκάδες υποψηφίων ειδικών εντολών, λίγες από αυτές (7 για την επίτευξη του 95% της μέγιστης επιτάχυνσης σε σύγκριση με τον αριθμό των 20 που χρειάζεται για την κάλυψη του 100%) συνεισφέρουν σημαντικά στον χρόνο εκτέλεσης της εφαρμογής για οποιοδήποτε μετρικό προτεραιότητας. Ο αριθμός των απαιτούμενων επεκτάσεων για την κάλυψη της επιτάχυνσης εφαρμογής στο επίπεδο του 95% κυμαίνεται από 2 (*fir.dlx*) ως 40 (*idea.dlx*), ενώ η απαίτηση σε επιφάνεια υλικού είναι μικρότερη από 3.4 μονάδες MAU. Αυτό ισχύει για όλες τις εφαρμογές με εξαίρεση την *idea.dlx* για την οποία απαίτηση αυτή φτάνει τις 10.23 MAU.

Τελικά, στο Σχ. 4.12 συγκρίνονται τα πλεονεκτήματα και μειονεκτήματα των συναρτήσεων προτεραιότητας που χρησιμοποιούνται για την διαδικασία της επιλογής ειδικών εντολών για το παράδειγμα της εφαρμογής *sha.dlx*. Έτσι, για την εφαρμογή sha στην αρχιτεκτονική iDLX η επιλογή ειδικών εντολών κάτω από το μετρικό 'Cycle gain' επιτυγχάνει το 95% της μέγιστης επιτάχυνσης για μία ειδική εντολή λιγότερη και μικρή αύξηση στην επιφάνεια (0.04 MAU) σε σχέση με το μετρικό 'Cycle gain/Area'.



Σχήμα 4.12: Επιλογή ειδικών εντολών κάτω από μετρικά προτεραιότητας για την εφαρμογή *sha* ($N_i/N_o=\{\infty/\infty\}$).

4.4. Περιβάλλον χρήσης του YARDstick

Το YARDstick έχει χρησιμοποιηθεί μαζί με τους μεταγλωττιστές SUIF/Machine-SUIF [SUIF],[MachSUIF], GCC [GCC], COINS [COINS] και την υποδομή προσομοίωσης ArchC [ArchC]. Προσομοιωτές ακρίβειας εντολής και ακρίβειας κύκλου που έχουν παραχθεί με την έκδοση 1.5.1 της ArchC μπορούν να χρησιμοποιηθούν με το YARDstick χωρίς καμία περαιτέρω τροποποίηση. Το μεγαλύτερο μέρος της λειτουργικότητας του YARDstick είναι προσβάσιμο από ένα γραφικό περιβάλλον χρήστη (GUI) συμβατό με πρόσφατες εκδόσεις της Tcl/Tk (8.5.a5 και νεώτερες εκδόσεις).

Οι υποστηρίζομενες πλατφόρμες ανάπτυξης συμπεριλαμβάνουν το GNU/Linux (RedHat 9.0 και πιθανόν και άλλες διανομές/εκδόσεις), Cygwin/x86 και Win32 (Windows/XP SP2) σε x86-συμβατούς επεξεργαστές.

5. ΓΛΩΣΣΑΡΙ ΟΡΩΝ

Addressing mode	τρόπος διευθυνσιοδότησης
Adjacency list	λίστα γειτνίασης
Application analysis	ανάλυση εφαρμογών
Argument	όρισμα (κατηγορημα) συνάρτησης
Argument stack	στοίβα ορισμάτων
Assembler	συμβολομεταφραστής
Assembly	γλώσσα συμβολομεταφραστή
Backtracking	οπισθοδρόμηση (κατά τα ακριβώς αντίρροπα βήματα)
Benchmarking	δοκιμασία επιδόσεων
Compiler	μεταγλωττιστής
Compiler infrastructure	υποδομή μεταγλωττιστή
Configuration	ρύθμιση, προσδιορισμός
Consistent data memory	συναφής μνήμη δεδομένων
Control step	βήμα (λογικής) ελέγχου
Convexity	κυρτότητα
Critical path	κρίσιμο μονοπάτι
Cut	τομή (γράφου)
Cycle accurate	ακρίβειας κύκλου
Cycle estimation	εκτίμηση κύκλου
Data hazard	κίνδυνος δεδομένων
Debugging	εκσφαλμάτωση
Delay slot	θυρίδα καθυστέρησης
Disjoint	ασύνδετος (γράφος, τομή)
Edge	ακμή (γράφου)
Embedded processor	ενσωματωμένος επεξεργαστής
Embedded software	ενσωματωμένο λογισμικό
Graph	γράφος
Graph isomorphism	ισομορφισμός γράφων
Graph matching	ταίριασμα γράφων
Graph rewriting	επανεγγραφή γράφου
Greedy selection	«άπληστη» επιλογή
Hardware	υλικό
Heuristic	ευρεστικός
Ideal consistent AFU memory	ιδανική συναφής μνήμη ΕΑΜ
Induced (sub)graph	παρακινούμενος (υπο)γράφος
Instruction accurate	με ακρίβεια εντολής
Instruction fetch	ανάκληση εντολής
Instruction scheduler	(χρονο)δρομολογητής εντολών
Interoperability	διεργαστικότητα
Intrinsic ILP	εγγενές ILP
Iterated register coalescing	επαναληπτική συνάσπιση καταχωρητών
Linker	συνδέτης
Memory consistency	συνάφεια μνήμης
Merit function	συνάρτηση αξίας
Motion compensation	αντιστάθμιση κίνησης
Motion estimation	εκτίμηση κίνησης
Multimedia	πολυμέσα
Node	κόμβος (γράφου)
Operand	έντελο/όρισμα (λειτουργίας)
Pass	εδώ: πέραςμα μεταγλωττιστή
Peephole optimization	βελτιστοποίηση κλειδαρότρυπας
Pipeline stage	βαθμίδα διοχέτευσης
Pixel	εικονοστοιχείο
Predecessor node	προηγούμενος (κόμβος)
Preorder traversal	διάβαση προκατατάξης
Priority function	συνάρτηση προτεραιότητας (βάσει κόστους)
Procedure calling convention	σύμβαση κλήσης ρουτίνων



Public domain	δημόσιο πεδίο
Quadruple (quad)	τετράδα
Reconvergent fan-out	επανασυγκλίνον φορτίο εξόδου
Register allocator	κατανομής καταχωρητών
Retargetable compiler	επαναστοχευόμενος μεταγλωττιστής
Reverse engineering	αντίστροφη μηχανική
Scheduling	(χρονο-) δρομολόγηση
Simulator	προσομοιωτής
Software	λογισμικό
Software development toolchain	αλυσίδα εργαλείων ανάπτυξης λογισμικού
Source code	πηγαίος κώδικας
State register	καταχωρητής κατάστασης
Successor (node)	διάδοχος (κόμβος)
Topological sort	τοπολογική ταξινόμηση
Type casting	ανάθεση τύπου
User-defined constraint	περιορισμός (καθορισμένος από τον) χρήστη
Virtual machine	εικονική μηχανή

API	Application Programming Interface	Διεπαφή Προγραμματισμού Εφαρμογής
ASAP	As Soon As Possible	
ASHE	Application-Specific Hardware Extension	Επέκταση Υλικού Ειδικού Σκοπού
ASIP	Application-Specific Instruction-set Processor	Επεξεργαστής Ειδικού Σκοπού Συνόλου Εντολών
AST	Abstract Syntax Tree	Αφηρημένο Συντακτικό Δένδρο
AFU	Application-specific Functional Unit	Ειδική Λειτουργική Μονάδα
BB	Basic block	Βασικό μπλοκ
CDFG	Control-Data Flow Graph	Γράφος Ροής Ελέγχου-Δεδομένων
CFG	Control Flow Graph	Γράφος Ροής Ελέγχου
CI	Custom Instruction	Ειδική Εντολή
CTI	Control Transfer Instruction	Εντολή Μεταφοράς Ελέγχου
DAG	Directed Acyclic Graph	Κατευθυνόμενος Άκυκλος Γράφος
DDG	Data Dependence Graph	Γράφος Εξάρτησης Δεδομένων
DFG	Data Flow Graph	Γράφος Ροής Δεδομένων
DMA	Direct Memory Access	Άμεση Προσπέλαση Μνήμης
DSE	Design Space Exploration	Διερεύνηση Πεδίου Λύσεων
DSP	Digital Signal Processor/Processing	Επεξεργαστής/Επεξεργασία Ψηφιακού Σήματος
GA	Genetic Algorithm	Γενετικός Αλγόριθμος
GCC	GNU Compiler-Compiler	
HDL	Hardware Description Language	Γλώσσα Περιγραφής Υλικού
HLL	High Level Language	Γλώσσα Υψηλού Επιπέδου
HLS	High-Level Synthesis	Σύνθεση Υψηλού Επιπέδου
ILP	Instruction-Level Parallelism	Παραλληλία Επιπέδου Εντολής
IR	Intermediate Representation	Ενδιάμεση Αναπαράσταση (μεταγλωττιστή)
ISE	Instruction-Set Extension	Επέκταση Συνόλου Εντολών
MAU	Multiplier Area Unit	Μονάδα Επιφάνειας Πολλαπλασιαστή
MaxMISO	Maximal MISO	Μέγιστη MISO (βλ. παρακάτω)
MISO	Multiple-Input/Single-Output	Πολλαπλών Εισόδων/Μονής Εξόδου
MIMO	Multiple-Input/Multiple-Output	Πολλαπλών Εισόδων/Πολλαπλών Εξόδων
PC	Program Counter	Απαριθμητής Προγράμματος
RISC	Reduced Instruction Set Computer	Υπολογιστής Μειωμένου Συνόλου Εντολών
RTOS	Real-Time Operating System	Λειτουργικό Σύστημα Πραγματικού Χρόνου
SCI	Single-Cut Identification	Αναγνώριση Μονής Τομής
SoC	System-on-Chip	Σύστημα-σε-ολοκληρωμένο
SSA	Static Single Assignment	Στατική Απλή Ανάθεση
SUIFrm	SUIF real machine	Πραγματική Μηχανή SUIF
SUIFvm	SUIF virtual machine	Εικονική Μηχανή SUIF
TAC	Three Address Code	Κώδικας Τριών Διευθύνσεων
VLIW	Very Long Instruction Word	Πολύ Μεγάλο Μήκος Εντολής



ZOLC	Zero-Overhead Loop Controller	Ελεγκτής Βρόχου Μηδενικής Επιβάρυνσης
BAM	Basic functional unit	Βασική Λειτουργική Μονάδα
ΕΕΣ	Application-specific Processor	Επεξεργαστής Ειδικού Σκοπού
ΕΑΜ	Application-specific functional unit	Ειδική Λειτουργική Μονάδα

6. ΑΝΑΦΟΡΕΣ

- [ACE] CoSy Compiler Development System, Associated Compiler Experts. <http://www.ace.nl>
- [AGG] AGG homepage. <http://tfs.cs.tu-berlin.de/agg/>
- [ARC] ARC homepage. <http://www.arc.com>
- [ArchC] The ArchC resource center. <http://www.archc.org>
- [ARM] ARM7TDMI Data Sheet, Advanced RISC Machines Ltd., Cambridge, UK, Dec. 1994. <http://www.arm.com>
- [Ber86] R. Bernstein, "Multiplication by integer constants," *Software: Practice and Experience*, Vol. 16, No. 7, pp. 641-652, July 1986.
- [Beu04] J.-L. Beuchat. A VHDL Library for Integer and Modular Arithmetic - User Manual, 0.1 edition, Sept. 2004. <http://perso.ens-lyon.fr/jean-luc.beuchat/ArithLib>
- [Binutils] GNU Binutils. <http://sources.redhat.com/binutils/>
- [Bis04] P. Biswas, V. Choudhary, K. Atasu, L. Pozzi, P. lenne, and N. Dutt, "Introduction of local memory elements in instruction set extensions," *Proceedings of the 41st ACM/IEEE Design Automation Conference*, pp. 729-734, Jun. 2004, San Diego, California, USA.
- [Bis07] P. Biswas, N. L. Dutt, L. Pozzi, and P. lenne, "Introduction of architecturally visible storage in instruction set extensions," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 26, No. 3, pp. 435-446, March 2007.
- [Bon06] P. Bonzini and L. Pozzi, "Code transformations strategies for extensible embedded processors," in *Proceedings of the 2006 International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, pp. 242-252, Seoul, Korea, Oct. 2006.
- [Bra04] A. Bracy, P. Prahlaad, and A. Roth, "Dataflow mini-graphs: Amplifying superscalar capacity and bandwidth," In *Proceedings of the 37th International Symposium on Microarchitecture*, pp. 18-29, Dec. 2004.
- [Bri94] P. Briggs, and T. Harvey, "Multiplication by integer constants," Technical report, Rice University, July 1994.
- [Cas04] P. Castro, E. Borin, R. Azevedo, and G. Araujo, "Looking for Instruction Patterns in the Design of Extensible Processors," *Proceedings of the 3rd Workshop on Application Specific Processors (WASP-3)*, Sep. 2004, Stockholm, Sweden.
- [CDFGtool] CDFG toolset. <http://poppy.snu.ac.kr/CDFG/cdfg.html>
- [Cha94] C. Charnes, L. O'Connor, J. Pieprzyk, R. Saifavi-Naini, and Y. Zheng, Further comments on the Soviet encryption algorithm, Technical report TR-9-94, University of Wollongong, Australia, June 1994.
- [Cla03] N. Clark, H. Zhong, W. Tang, and S. Mahlke, "Automatic design of application specific instruction set extensions through dataflow graph exploration," *International Journal of Parallel Programming*, Vol. 31, No. 6, pp. 429-449, Dec. 2003.
- [Cla05] N. Clark, J. A. Blome, M. L. Chu, S. A. Mahlke, S. Biles, and K. Flautner, "An architecture framework for transparent instruction set customization in embedded processors," in *Proc. 32nd Int. Symp. on Computer Architecture*, Madison, Wisconsin, USA, June 2005, pp. 272-283.
- [COINS] COINS: A COmpiler Infra-Structure home page. <http://www.coins-project.org>
- [Con04] J. Cong, Y. Fan, G. Han, and Z. Zhang, "Application-specific instruction generation for configurable processor architectures," In *Proceedings of the 2004 ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays*, pp. 183-189, Feb. 2004.
- [Cyt91] R. Cytron, J. Ferrante, B. Rosen, M. Wegman, and K. Zadeck, "Efficiently computing static single assignment form and the control dependence graph," *ACM Transactions on Programming Languages and Systems*, Vol. 13, No. 4, pp. 451-490, October 1991.
- [EPFL-passes] Optimization passes for Machine-SUIF. http://lap.epfl.ch/dev/machsuif/opt_passes/
- [Fog01] P. Foggia, The VFLib Graph Matching Library, 2.0 edition, Mar. 2001. <http://amalfi.dis.unina.it/graph/db/vflib-2.0>
- [GCC] The GNU Compiler Collection homepage. <http://gcc.gnu.org>
- [Geo96] L. George and A. W. Appel, "Iterated register coalescing," *ACM Transactions on Programming Languages and Systems*, vol. 18, no. 3, pp. 300-324, May 1996.
- [Gol89] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, USA, 1989.

- [Goo03] D. Goodwin and D. Petkov, "Automatic generation of application specific processors," Proceedings of the International Conference on Compilers, Architectures and Synthesis for Embedded Systems, pp. 137-147, Oct. 2003, San Jose, California, USA.
- [Graphviz] Graphviz homepage. <http://www.graphviz.org>
- [Kav04] N. Kavvadias and S. Nikolaidis, "Application analysis with integrated identification of complex instructions for configurable processors," In Proceedings of the 14th International Workshop on Power and Timing Modeling, Optimization and Simulation, pp. 633-642, Sept. 15-17 2004.
- [Kav05a] N. Kavvadias and S. Nikolaidis, "Zero-overhead loop controller that implements multimedia algorithms," IEE Proceedings - Computers and Digital Techniques, vol. 152, no. 4, pp. 517-526, July 2005.
- [Kav05b] N. Kavvadias and S. Nikolaidis, "Automated Instruction-Set Extension of Embedded Processors with Application to MPEG-4 Video Encoding," in Proceedings of the IEEE 16th International Conference on Application-specific Systems, Architectures and Processors, pp. 140-145, July 2005, Samos, Greece.
- [Kav07] N. Kavvadias and S. Nikolaidis, "YARDstick: Automation for custom processor development," in presented at the University Booth of the Design, Automation and Test in Europe Conference (DATE'07), Apr. 2007.
- [Kog81] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," In Proceedings of the National Telecommunications Conference, pp. G5.3.1-G5.3.5, Nov. 1981.
- [Kum00] S. Kumar, L. Pires, S. Ponnuswamy, C. Nanavati, J. Golusky, M. Vojta, S. Wadi, D. Pandalai, and H. Spaanenburg, "A benchmark suite for evaluating configurable computing systems - status, re_ections, and future directions," in Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays, Monterey, CA, USA, February 2000.
- [LANCE] LANCE C compiler. <http://www.lancecompiler.com>
- [LCC] LCC retargetable C compiler. <http://www.cs.princeton.edu/software/lcc/>
- [Lee99] L. H. Lee, W. Moyer, and J. Arends, "Instruction fetch energy reduction using loop caches for embedded applications with small tight loops," In Proceedings of the International Symposium on Low Power Electronics and Design, San Diego, CA, Aug. 1999.
- [Leu06] R. Leupers, K. Karuri, S. Kraemer, and M. Pandey, "A design flow for configurable embedded processors based on optimized instruction set extension synthesis," in *Proc. of the Design, Automation and Test in Europe Conf.*, March 6-10 2006.
- [LLVM] The LLVM compiler infrastructure. <http://www.llvm.org>
- [MachSUIF] Machine-SUIF research compiler. <http://www.eecs.harvard.edu/hube/research/machsuiif.html>
- [MiBench] MiBench benchmark suite. <http://www.eecs.umich.edu/mibench/>
- [MIPS] MIPS Technologies Inc.. <http://www.mips.com>
- [MPEG4senc] MPEG-4 shape encoder project. <http://www.ece.cmu.edu/~ece796/project99/12/final/>
- [Nios-II] Altera Nios-II home page. <http://www.altera.com/products/ip/processors/nios2/>
- [Nios-II-CI] Altera Corporation, "Nios II Custom Instruction: User Guide," December 2004.
- [Pot07] N. Pothineni, A. Kumar, and K. Paul, "Application specific datapath extension with distributed I/O functional units," in Proceedings of the 20th International Conference on VLSI Design (VLSI Design 2007), 6th International Conference on Embedded Systems (ICES 2007), pp. 551-558, Bangalore, India, Jan. 2007.
- [Poz00] L. Pozzi, "Methodologies for the Design of Application-Specific Reconfigurable VLIW Processors," Ph.D. Thesis, Politecnico di Milaon, Dipartimento di Elettronica e Informazione, Milan, Italy, Jan. 2000.
- [Poz01] L. Pozzi, M. Vuletic, and P. lenne, "Automatic topology-based identification of instruction-set extensions for embedded processors," Technical report CS 01/377, Swiss Federal Institute of Technology Lausanne, Processor Architecture Laboratory, Dec. 2001.
- [Poz06] L. Pozzi, K. Atasu, and P. lenne, "Exact and Approximate Algorithms for the Extension of Embedded Processor Instruction Sets," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 25, No. 7, pp. 1209-1229, July 2006.
- [Riv94] R. L. Rivest, "The RC5 encryption algorithm," In Proceedings of Fast Software Encryption, pp. 86-96, 1994.
- [SALTO] SALTO. <http://www.irisa.fr/caps/projects/Salto/>
- [Sas04] P. G. Sassone, and D. S. Wills, "On the extraction and analysis of prevalent dataflow patterns," In Proceedings of the 7th Annual IEEE International Workshop on Workload Characterization, Oct. 2004.



- [Scho07] Andreas Schoesser and Rubino Geiss, “Graph rewriting for hardware dependent program optimizations”, In Proceedings of AGTIVE 2007.
- [Sin04] J. Singer, Static Single Information form in Machine SUIF, machsuifssi user manual, version 0.1, Mar. 2004.
- [Sir07] S. Sirowy, Y. Wu, S. Lonardi, and F. Vahid, “Two-level microprocessor accelerator partitioning,” in Proc. of the Design, Automation and Test in Europe Conf., Nice, France, April 16-20, 2007.
- [SPARK] Parallelizing High-Level Synthesis Framework. <http://mesl.ucsd.edu/spark/>
- [SUIF] SUIF compiler infrastructure. <http://suif.stanford.edu/suif/suif2/>
- [Tensilica] Xtensa configurable processors. <http://www.tensilica.com>
- [Vas05] N. D. Vassiliadis, N. Kavvadias, G. Theodoridis, and S. Nikolaidis, “A RISC architecture extended by an efficient tightly coupled reconfigurable unit,” In Proceedings of the 1st International Workshop on Applied Reconfigurable Computing 2005 (ARC 2005), pp. 41-49, Feb. 2005.
- [VCG] VCG website. <http://rw4.cs.ui-sb.de/~sander/html/gsvcg1.html>
- [Xilinx] Xilinx homepage. <http://www.xilinx.com>
- [Yu04] P. Yu, and T. Mitra, “Characterizing Embedded Applications for Instruction-Set Extensible Processors,” Proceedings of the 41st ACM/IEEE Design Automation Conference, pp. 723-728, Jun. 2004, San Diego, CA, USA.