ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

Τμήμα Ηλεκτρονικών Μηχανικών και Μηχανικών Υπολογιστών

Διπλωματική Εργασία

Υλοποίηση Αλγορίθμων Επίλυσης Προβλημάτων Ικανοποιησιμότητας (SAT) με χρήση Αναδιατασσόμενης Λογικής

Μιχαήλ Σκίτσας

Επιτροπή
Απόστολος Δόλλας (Επιβλέπων)
Ιωάννης Παπαευσταθίου
Μιχαήλ Λαγουδάκης

Χανιά, 17 Ιουλίου 2009

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον Καθηγητή κ. Απόστολο Δόλλα για την επίβλεψη, την καθοδήγηση και τις χρήσιμες υποδείξεις του για την ολοκλήρωση αυτής της εργασίας. Επίσης θα ήθελα να ευχαριστήσω και τα υπόλοιπα μέλη της επιτροπής, τους Καθηγητές κ. Ιωάννη Παπαευσταθίου και κ. Μιχάηλ Λαγουδάκη για το ενδιαφέρον που έδειξαν για αυτήν την εργασία και για τις χρήσιμες παρατηρήσεις τους.

Επίσης θα ήθελα να ευχαριστήσω τον κ. Κυπριανό Παπαδημητρίου για τις πολύτιμες συμβουλές του, τη συνεχή καθοδήγησή του και τη βοήθειά του στη λύση διαφόρων προβλημάτων μέχρι την ολοκλήρωση της εργασίας αυτής.

Επίσης ευχαριστώ τον κ. Μάρκο Κιμιωνή και όλους τους προπτυχιακούς και μεταπτυχιακούς φοιτητές του Εργαστηρίου Μικροεπεξεργαστών και Υλικού για την βοήθεια τους όποτε αυτή χρειάστηκε.

Επίσης ευχαριστώ όλους τους φίλους και συμφοιτητές μου για το συνεχές ενδιαφέρον τους και την συμπαράσταση τους σε όλη τη διάρκεια των σπουδών μου.

Τέλος ένα μεγάλο ευχαριστώ στους χρηματοδότες κατά την διάρκεια των σπουδών μου που δεν είναι άλλοι από τους γόνεις μου, Ανδρέα και Αργυρούλα οι οποίοι με στήριξαν όλα αυτά τα χρόνια.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

Τα προβλήματα ικανοποιησιμότητας (**SAT**isfiability) είναι μια μεγάλη κατηγορία προβλημάτων τα οποία ανοίκουν στην κλάση NP-Complete. Τα προβλήματα αυτά παρουσιάζονται κυρίως ώς Boolean εκφράσεις μέσα από προτάσεις **CNF** (Conjuctive Normal Form). Το ζητούμενο στα προβλήματα SAT είναι να δούμε κατα πόσο υπάρχει ένα μοντέλο μεταβλητών το οποίο να ικανοποιεί την συγκεκριμένη πρόταση. Σε αυτή την διπλωματική εργασία θα γίνει μελέτη αλγορίθμων οι οποίοι λύνουν αυτά τα προβλήματα και πως μπορούν να αποτυπωθούν σε αναδιατασσόμενη λογική.

Στο δεύτερο κεφάλαιο θα δούμε αναλυτικότερα τα προβλήματα SAT, την δυσκολία των προβλημάτων καθώς και διάφορους αλγόριθμους επίλυσης τους με κυριότερη έμφαση στους αλγορίθμους τοπικής αναζήτησης στους οποίου ανήκει και ο WalkSAT που υλοποιούμε και σε αναδιατασσόμενη λογική. Επίσης στο κεφάλαιο αυτό θα γίνει αναφορά στις εφαρμογές των προβλημάτων SAT και στο SAT Competition.

Στο τρίτο κεφάλαιο παρουσιάζουμε τον αλγόριθμο επίλυσης SAT, WalkSAT. Γίνεται η περιγραφή του αλγορίθμου όσο αφορά την λειτουργία του και τις ευριστικές που χρησιμοποιεί. Γίνεται μοντελοποίηση του αλγορίθμου σε γλώσσα προγραμματισμού C όπου θα γίνει πειραματική μελέτη των ευριστικών με τις οποίες θα ασχοληθούμε. Τέλος παρουσιάζουμε τα πειραματικά αποτελέσματα και γίνεται σύγκριση των αποδόσεων των ευριστικών συναρτήσεων.

Στο τέταρτο κεφάλαιο γίνεται μια πρώτη εισαγωγή σε αναδιατασσόμενη λογική και προβλήματα SAT. Παρουσιάζεται η αρχική προσέγγιση της διπλωματικής η οποία είχε σκοπό να εφαρμοστεί η δυναμική αναδιάταξη στον αλγόριθμο WalkSAT καθώς και άλλες δουλειές γύρω από την δυναμική αναδιάταξη. Στη συνέχεια παρουσιάζουμε τις κυριότερες δουλειές που έγιναν με την χρήση αναδιατασσόμενης λογικής και γιατί επιλέξαμε να υλοποιήσουμε τον αλγόριθμο WalkSAT με την ευριστική Novelty σε αναδιατασσόμενη λογική.

Στο πέμπτο κεφάλαιο παρουσιάζεται και αναλύεται η αρχιτεκτονική που υλοποιεί τον αλγόριθμο WalkSAT σε αναδιατασσόμενη λογική με την χρήση της ευριστικής Novelty. Παρουσιάζονται αναλυτικά τα τμήματα της αρχιτεκτονικής που αποτελούν το Datapath μέσα από διαγράμματα και περιγραφή καθώς επίσης και η μονάδα ελέγχου (Control path) του συστήματος η οποία αποτελείται από μια FSM. Γίνεται περιγραφή της διαδικασίας αρχικοποίησης του συστήματος και της ροής του αλγορίθμου.

Στο έκτο κεφάλαιο παρουσιάζονται τα αποτελέσματα της αρχιτεκτονικής όσο αφορά την αποτύπωση σε αναδιατασσόμενη λογική και τους πόρους που χρειάζονται για κάθε μέγεθος δεδομένων καθώς και την συχνότητα λειτουργίας. Στη συνέχεια γίνεται σύγκριση με αντίστοιχες δουλείες τόσο σε επίπεδο υλικού όσο και επίπεδο λογισμικού.

Στο τελευταίο, έκτο, κεφάλαιο παρατίθενται τα συμπεράσματα που βγαίνουν από τα αποτελέσματα αλλά και από την σχεδίαση. Δηλαδή, το πόσο βελτιώνεται η απόδοση με την απεικόνιση ενός απαιτητικού υπολογιστικά αλγόριθμου σε τεχνολογία ΦΠΓΑ. Επίσης παρατίθενται μελλοντικές επεκτάσεις για την αρχιτεκτονική και γενικά για την εργασία αυτή.

Κεφάλαιο 2

Προβλήματα Ικανοποιησιμότητας SAT

Τα προβλήματα ικανοποιησιμότητας γνωστά και ως Boolean Satisfiability Problems (SAT) ανήκουν στην κλάση NP-complete προβλημάτων. Τα προβλήματα SAT όπως θα δούμε και στην συνέχεια παρουσιάζονται σε Conjuctive Normal Form (CNF). Σκοπός είναι να βρεθεί μια ανάθεση τιμών, των μεταβλητών που το αποτελούν ώστε η πρόταση CNF να ικανοποιείται.

2.1 Μορφή Προβλημάτων SAT

2.1.1 Conjunctive Normal Form

Στη συνέχεια παρουσιάζεται μια γενική μορφή της CNF όπου φαίνονται τα στοιχεία που την αποτελούν.

$$\underbrace{(v_{11} \vee v_{12} \vee ... \vee v_{1k})}_{Clause} \wedge (\overbrace{v_{21}}^{Literal} \vee v_{22} \vee ... \vee v_{2l}) \wedge ... \wedge (v_{n1} \vee v_{n2} \vee ... \vee v_{nm}) \quad (2.1)$$

Όπως φαίνεται και από την έκφραση 2.1 η CNF αποτελείται από clauses και literals τα οποία είναι συνδεδεμένα μεταξύ τους με τους τελεστές And($\wedge$) και Or($\vee$). Πιο αναλυτικά για την έκφραση έχουμε:

Variable: Οι μεταβλητές που αποτελούν το πρόβλημα. Εμφανίζονται μέσα στα clauses ως literals και έχουν πεδίο τιμών true/false. Κάθε μεταβλητή συνήθως παρουσιάζετε περισσότερο από μία φορές σε ένα πρόβλημα.

Literal: Με τον όρο αυτό εννοούμε την απεικόνιση της μεταβλητής μέσα στο clause. Κάθε ένα literal αναπαριστά μια μεταβλητή και μπορεί να είναι στην θετική (′1′) ή αρνητική (′0′) μορφή του.

Clause: Κάθε ένα clause αποτελείται από literals τα οποία συνδέονται μεταξύ τους με τελεστές OR. Πολλά clauses συνδεδεμένα με τελεστές AND αποτελούν την CNF. Μέσα σε μια CNF τα clauses μπορεί να έχουν διαφορετικό αριθμό από literals.

2.1.2 Προβλήματα 3-SAT

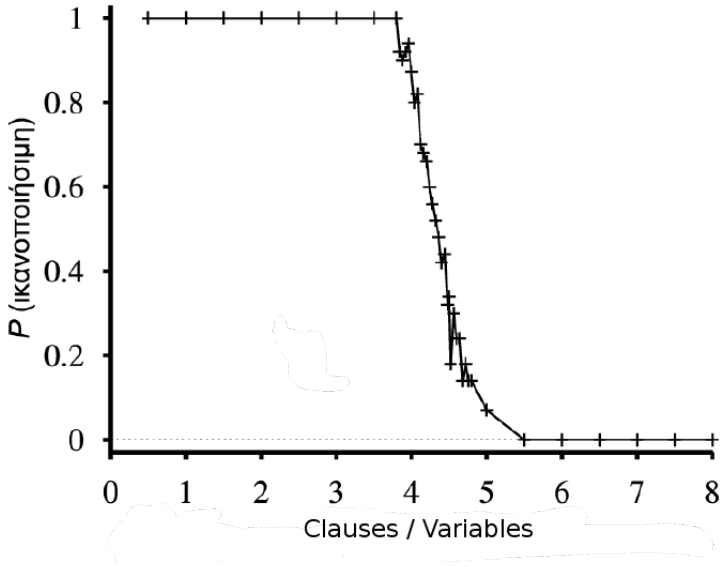
Είναι μια κατηγορία προβλημάτων SAT στην οποία κάθε clause περιέχει ακριβώς τρία literals. Η ιδιαίτεροτητα αυτών των προβλημάτων είναι ότι όλα τα SAT μπορούν σε πολυονυμικό χρόνο να αναχθούν σε 3-SAT όπως αναφαίρεται και στο [ΛΠ97].

2.2 Δύσκολα Προβλήματα SAT

Η δυσκολία των προβλημάτων καθορίζεται από τον λόγο των clauses/variables που περιέχει το κάθε πρόβλημα. Στο γράφημα 2.1 από το [PN02] παρουσιάζεται η πιθανότητα εύρεσης λύσης συναρτήσει του λόγου $Clauses/Variables$ Για το γράφημα αυτό χρησιμοποιήθηκαν τυχαία προβλήματα με 50 μεταβλητές και μεταβαλλόμενο τον αριθμό clauses. Στο σχήμα 2.2 παρουσιάζονται τα γραφήματα τα οποία προέκυψαν μέσα από μια σειρά πειράματατων που έγιναν χρησιμοποιώντας το λογισμικό που περιγράφεται στην ενότητα 3.3 και προβλήματα τα οποία δημιουργήθηκαν τυχαία με την χρήση της γλώσσας προγραμματισμού C με σκοπό να ελεγχθεί η πιθανότητα εύρεσης λύσης ανάλογα με τις διάφορες τιμές του λόγου $Clauses/Variables$.

Στο 2.2(α′) έχουμε την πιθανότητα εύρεσης λύσης έχοντας σταθερές τις μεταβλητές στα προβλήματα και μεταβάλλοντας των αριθμών των clauses. Ο αριθμός των μεταβλητών τέθηκε ίσος με 50 και τα clauses κυμάνθηκαν από 50 μέχρι 1000. Στο 2.2(β′) έχουμε παρόμοιο πείραμα αλλά αυτή την φορά κρατήσαμε σταθερό των αριθμό των clauses ίσο με 1000 και μεταβάλουμε τον αριθμό των μεταβλητών από 50 μέχρι 500.

Όπως φαίνεται και πιο πάνω η συσχέτιση clauses και μεταβλητών καθορίζει την δυσκολία των προβλημάτων SAT. Στην βιβλιογραφία ο λόγος $\frac{Clauses}{Variables} = 4.26$ θεωρείται το κρίσιμο σημείο κατά το οποίο η πιθανότητα εύρεσης λύσης σε ένα πρόβλημα αρχίζει να μειώνεται. Για λόγους μέχρι το κρίσιμο σημείο παρατηρούμε πιθανότητα λύσης 1 και όσο αυξάνεται ο λόγος η πιθανότητα τίνει προς το 0 όπου και γίνεται.



Σχήμα 2.1: Σχέση Μεταβλητών - Clauses

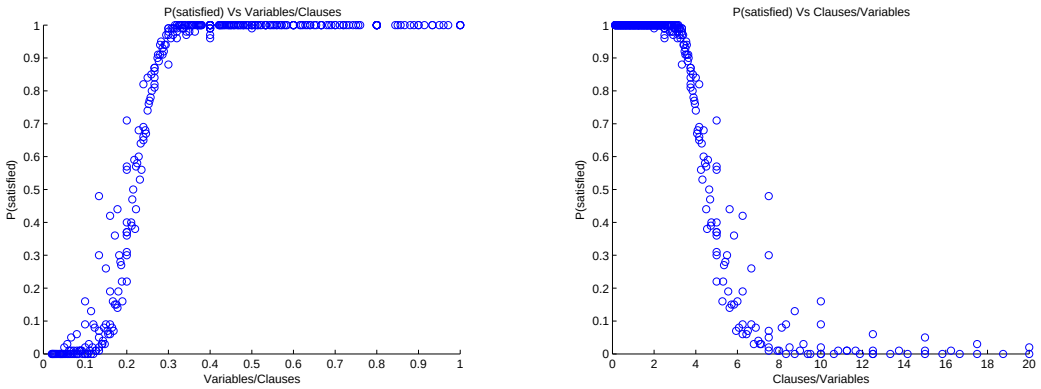
2.3 Αλγόριθμοι Επίλυσης SAT

Υπάρχουν δύο κατηγορίες αλγορίθμων για την επίλυση προβλημάτων SAT. Οι αλγόριθμοι συστηματικής αναζήτησης και οι στοχαστικοί αλγόριθμοι τοπικής αναζήτησης.

2.3.1 Αλγόριθμοι Συστηματικής Αναζήτησης

Οι αλγόριθμοι συστηματικής αναζήτησης ερευνούν όλα τα πιθανά μοντέλα μεταβλητών μέχρι να βρούνε το μοντέλο που ικανοποιεί το πρόβλημα. Τα χαρακτηριστικά αυτού του αλγορίθμου είναι ότι είναι πλήρης και κάνει αναζήτηση με υπαναχώρηση.

Ο πιο διαδεδομένος αλγόριθμος συστηματικής αναζήτησης είναι ο DPLL (Davis, Putman, Logemann, Loveland) ο οποίος προτάθηκε το 1962. Εκτός από τον DPLL στην συστηματική αναζήτηση έχουν αναπτυχθεί και άλλοι αλγόριθμοι όπως Chaff, GRASP και march οι οποίοι βασίζονται στον DPLL και εφαρμόζουν τεχνικές βελτίωσης της απόδοσης του όπως μάθηση βοηθητικών clauses ή ευρετική επιλογή συμβόλου διακλάδωσης.

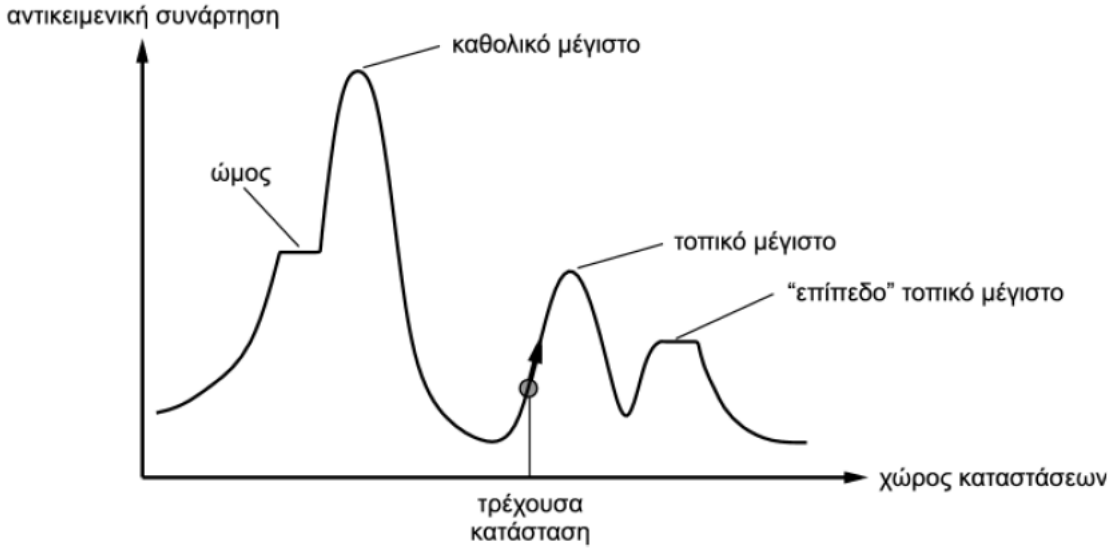


(α′) Μεταβλητές (β′) Προτάσεις

Σχήμα 2.2: Σχέση Μεταβλητών - Clauses

2.3.2 Αλγόριθμοι Τοπικής Αναζήτησης

Σε αυτή την ενότητα θα αναφέρουμε δύο αλγορίθμους τοπικής αναζήτησης και πως αυτοί μπορούν να εφαρμοστούν σε προβλήματα SAT απευθείας εφόσον επιλεγεί η κατάλληλη ευριστική συνάρτηση ή συνάρτηση αξιολόγησης. Πριν αναφερθούμε στους αλγόριθμους αυτούς θα ορίσουμε τον χώρο καταστάσεων ο οποίος είναι πολύ σημαντικός στην τοπική αναζήτηση.



Σχήμα 2.3: Τοπίο Χώρου Καταστάσεων

Στο σχήμα 2.3 μέσα από το [PN02] παρατηρούμε το γράφημα που παρουσιάζει κάποιο χώρο καταστάσεων. Ένα τοπίο έχει τοποθεσία που ορίζεται από την κατάσταση και υψόμετρο που ορίζεται από την τιμή της ευριστικής συνάρτησης ή συνάρτησης αξιολόγησης. Ζητούμενο είναι να βρεθεί η χαμηλότερη κοιλάδα, καθολικό ελάχιστο, σε περίπτωση που το υψόμετρο αντιστοιχεί σε κόστος ή σε αντίθετη περίπτωση να βρεθεί η ψηλότερη κορυφή, καθολικό μέγιστο. Ανάλογα με το πρόβλημα που μελετά ο αλγόριθμος τοπικής αναζήτησης καταστάσεις στόχου μπορεί να είναι οποιαδήποτε μέγιστα ή ελάχιστα. Στην περίπτωση που ο αλγόριθμος είναι πλήρης τότε θα μπορεί να βρεί μια κατάσταση στόχου εφόσον υπάρχει και σε περίπτωση που είναι βέλτιστος τότε θα είναι σε θέση να βρει το καθολικό μέγιστο ή ελάχιστο.

Αναρρήχιση Λόφων

Η αναζήτηση με αναρρίχηση λόφων (Hill Climbing) λέγεται και μερικές φορές άπληστη τοπική αναζήτηση (Greedy Local Search) και ο λόγος είναι ότι πάει στην καλύτερη γειτονική κατάσταση χώρις να ακολουθεί την ευριστική συνάρτηση ή συνάρτηση αξιολόγησης. Σε γενικές γραμμές οι άπληστοι αλγόριθμοι αποδίδουν καλά γιατί μπορούν έυκολα να βελτιώσουν μια κακή κατάσταση και να κάνουν πολύ γρήγορη πρόοδο προς μια λύση.

Στο χώρο καταστάσεων υπάρχουν κάποια σημεία τα οποία μπορεί να δημιουργήσουν πρόβλημα στην αναζήτηση με αναρρίχηση λόφων. Πιο κάτω περιγράφουμε τα σημεία αυτά που φαίνονται και στο σχήμα 2.3 και πως επηρεάζουν την αναρρίχηση λόφων.

Τοπικά μέγιστα Τοπικό μέγιστο είναι μια κορυφή υψηλότερη από κάθε γειτονική της κατάσταση αλλά χαμηλότερη από το καθολικό μέγιστο. Οι αλγόριθμοι αναρρίχησης λόφων που φτάνουν κοντά σε ένα τοπικό μέγιστο ανεβαίνουν προς αυτό και στην συνέχεια παγιδεύονται μη έχοντας που αλλού να πάνε. Σε αυτοί την περίπτωση η συνάρτηση αξιολόγησης θα είναι χειρότερη σε κάθε γειτονική κατάσταση από την τρέχουσα κατάσταση, έτσι ο αλγόριθμος θα τερματίσει και θα επιστρέψει την τρέχουσα κατάσταση η οποία δεν θα είναι και η βέλτιστη εφόσον είναι τοπικό ελάχιστο ή μέγιστο και όχι καθολικό.

Κορυφογραμμές Οι κορυφογραμμές έχουν ώς αποτέλεσμα μια ακολουθία τοπικών μεγίστων που κάνουν πολύ δύσκολη την πλοήγηση για τους άπληστους αλγόριθμους.

Οροπέδια Οροπέδιο είναι μια περιοχή του τοπίου του χώρου κατάστασης όπου η συνάρτηση αξιολόγισης είναι επίπεδη. Το οροπέδιο παρουσιάζεται σαν ένα επίπεδο τοπικό μέγιστο ή σαν όμως όπως φαίνεται στο σχήμα 2.3. Το πρόβλημα της αναζήτηση με αναρρίχηση λόφων είναι ότι μπορεί να μην βρει διέξοδο από το οροπέδιο και ο αλγόριθμος να μην προχωρά προς μέγιστο ή ελάχιστο.

Στην προσπάθεια βελτίωσης του αγλορίθμου αναζήτησης με αναρρίχηση λόφων έχουν προταθεί κάποιες παραλλαγές όπως η στοχαστική αναρρίχηση λόφων, η αναρρίχηση λόφων με πρώτη επιλογή και η αναρρίχηση λόφων με τυχαίες επανεκκινήσεις.

Προσομοιωμένη Ανόπτηση

Ο αλγόριθμος της προσομοιωμένης ανόπτησης (Simulated Annealing) είναι ένας αλγόριθμος ο οποίος συνδυάζει την αναρρίχηση λόφων που προαναφέρεται με έναν τυχαίο περίπατο. Ένας τυχαίος περίπατος κάνει μετακίνηση σε διαδοχική κατάσταση επιλέγοντας τυχαία μέσα από το σύνολο διαδοχικών καταστάσεων και μπορεί να θεωρηθεί πλήρης με εξαιρετικά χαμηλή απόδοση σε σχέση με την αναρρίχηση λόφων.

Ο βασικός βρόγχος του αλγορίθμου είναι παρόμοιος με αυτόν της αναρρίχησης λόφων αλλά επιλέγει τυχαία μια κίνηση μέσα από το σύνολο διαδοχικών καταστάσεων. Στη συνέχεια ελέγχεται αν η κίνηση παράγει καλύτερη λύση. Σε περίπτωση η λύση είναι καλύτερη αυτή γίνεται αποδεκτή και εφαρμόζεται ενώ σε αντίθετη περίπτωση επιλέγεται βάση μιας πιθανότητας p.

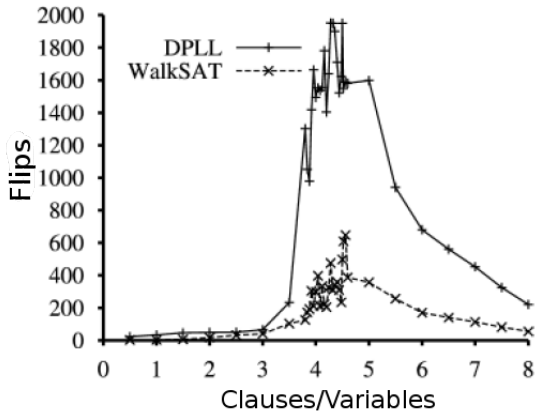
Εφόσον στα προβλήματα SAT ο στόχος είναι να βρεθεί μια ανάθεση τιμών που να ικανοποιείται η πρόταση, μπορούν να εφαρμοστούν οι αλγόριθμοι τοπικής αναζήτησης όπως προαναφέρονται. Ο αριθμώς των clauses που δεν ικανοποιούντε μπορεί να καθορίσει την ευριστική συνάρτηση όπως θα δούμε και στην ενότητα 3.2. Όλοι αυτοί οι αλγόριθμοι πραγματοποιούν βήματα στον χώρο των πλήρων αναθέσεων τιμών, αλλάζοντας την τιμή αλήθειας μιας μεταβλητής τη φορά. Ο χώρος αυτός περιέχει συνήθως πολλά τοπικά ελάχιστα όπου χρησιμοποιούντε διάφορες μορφές τυχαιότητας για αποφυγή τους.

2.3.3 Σύγκριση συστηματικής και τοπικής αναζήτησης

Στο σχήμα 2.4 από το [PN02] παρουσιάζεται ο μέσος χρόνος εκτέλεσης σε 100 επιλύσιμα προβλήματα στο οποίο φαίνεται η καλή απόδοση του αλγορίθμου WalkSAT και η δυνατότητα που έχει για εύρεση λύσης σε σχετικά γρήγορο χρόνο.

2.4 Εφαρμογές των προβλημάτων SAT

Στη συνέχεια παρουσιάζονται ενδεικτικές εφαρμογές που χρησιμοποιούν μεθόδους επίλυσης προβλημάτων SAT.



Σχήμα 2.4: Σύγκριση DPLL και Walksatώς προς το χρόνο επίλυσης

Combinational Equivalence Checking

Είναι μια διαδικασία η οποία χρησιμοποιείτε στα εργαλία EDA (Electronic Design Automation) κατά την ανάπτυξη ψηφιακών ολοκληρωμένων κυκλωμάτων. Σκοπό έχει την απόδειξη ότι δύο αναπαραστάσεις κυκλωμάτων έχουν την ίδια συμπεριφορά.

Automatic Test-Pattern Generation

Μια μέθοδος η οποία πάλι χρησιμοποιείτε από τα εργαλία EDA και έχει σκοπό να βρεί μια σειρά εισόδων οι οποίες όταν εφαρμοστούν σε ένα ψηφιακό κύκλωμα να μπορούν να διαπιστώσουν αν το κύκλωμα συμπεριφέρεται σωστά. Τα πρότυπα που δημιουργούντε χρησιμοποιούντε για έλεγχο σε ψηφιακά κυκλώματα τα οποία έχουν ήδη παραχθεί.

Planning in Artificial Intelligence

Μέσα από την λύση του προβλήματος SAT το οποίο αποτελείται από τις καταστάσεις μετάβασης, αξιώματα κίνησης, θέσης κλπ μπορούμε να διαπιστόσουμε αν ο στόχος μπορεί να γίνει εφικτός.

2.5 SAT Competition

Το SAT Competition είναι ένας διαγωνισμός που διεξάγεται σχεδόν κάθε χρόνο και αφορά αλγορίθμους για την επίλυση SAT προβημάτων. Σκοπός του διαγωνισμού είναι να οριστούν νέα benchmarks από τα οποία κάποια προέρχονται από πραγματικά προβλήματα μέσα από το χώρο του planning, software και harware verification και νέοι sat solvers όπου θα συγκριθούν ως προς την απόδοση τους.

Κεφάλαιο 3

Αλγόριθμος Walksat

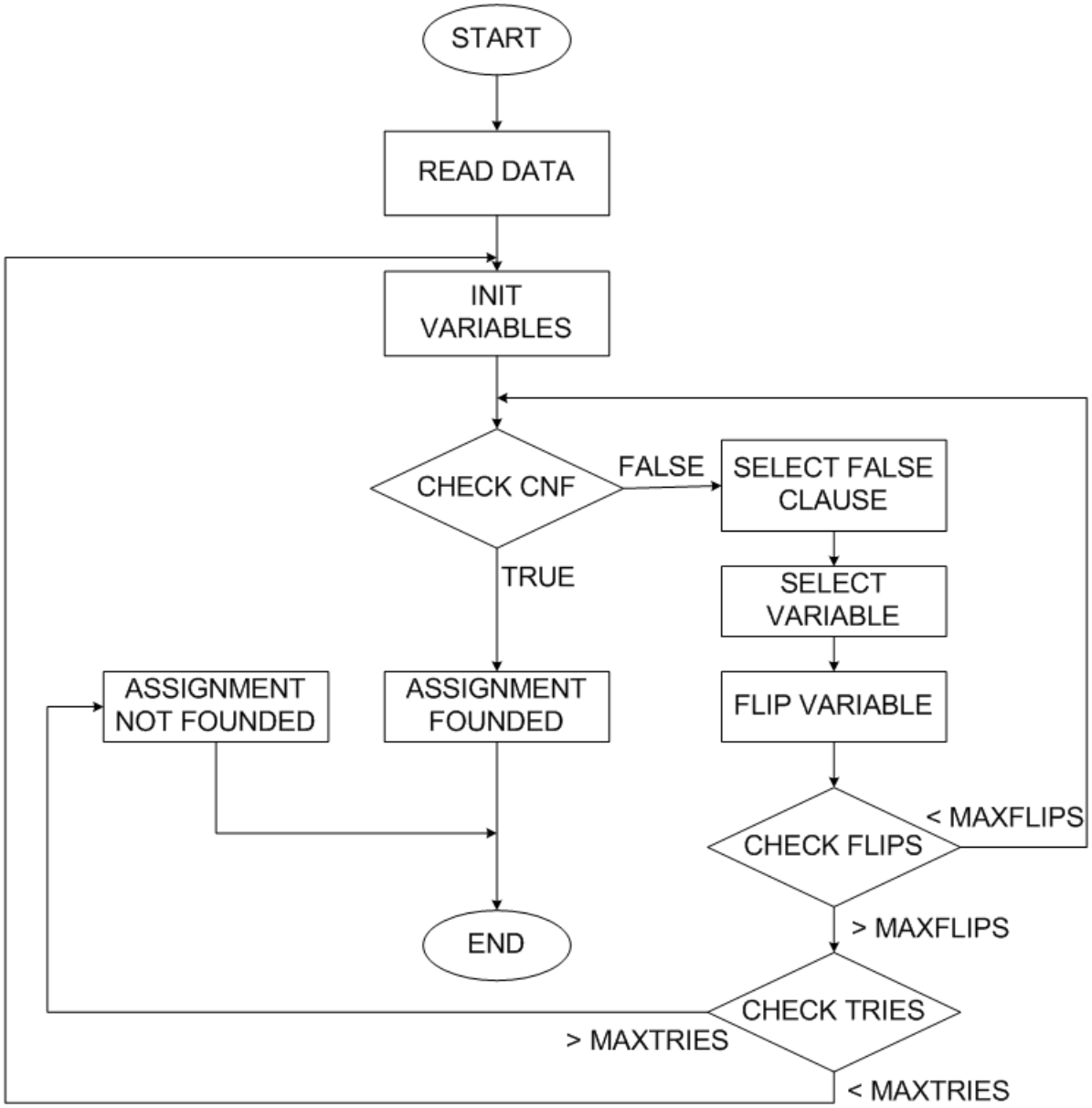
Ο αλγόριθμος Walksat είναι αλγόριθμος τοπικής αναζήτησης ο οποίος χαρακτηρίζεται από την απλότητα του και την δυνατότητα εύρεσης λύσης σε δύσκολα προβλήματα ικανοποιησιμότητας μέσα σε σύντομο χρονικό διάστημα. Υπάρχουν και άλλοι αλγόριθμοι όπως Minisat, Picosat οι οποίοι είναι πέρα από τους σκοπούς της παρούσας διπλωματικής εργασίας.

3.1 Περιγραφή Αλγορίθμου

Βασική ιδέα του αλγορίθμου είναι, σε κάθε επανάληψη να παίρνει μια μη ικανοποιούμενη διαζευκτική πρόταση (clause) και να επιλέγει μια μεταβλητή της οποίας θα αλλάξει την τιμή. Πιο κάτω φαίνεται ο ψευδοκώδικας του αλγορίθμου του οποίου παρουσιάζεται και το διάγραμμα ροής στο σχήμα 3.1.

function WalkSAT(*clauses*, *p*, *max-flips*) **returns** a satisfying model or *failure*
  inputs: *clauses*, a set of clauses in propositional logic
    p, the probability of choosing to do a "random walk" move, typically around 0.5
    max-flips, number of flips allowed before giving up
  model ← a random assignment of *true*/*false* to the symbols in *clauses*

  for $i = 1$ **to** *max-flips* **do**
    if *model* satisfies *clauses* **then return** *model*
    clause ← a randomly selected clause from *clauses* that is false in *model*
    With **select()** choose a variable to flip
  return *failure*



Σχήμα 3.1: Διάγραμμα Ροής WalkSAT

Ανάλυση Ψευδοκώδικα

Ο αλγόριθμος δέχεται για είσοδο το πρόβλημα σε μορφή CNF , τον μέγιστο αριθμό flips , τον μέγιστο αριθμό προσπαθιών (Max-Tries) και τις παραμέτρους οι οποίες καθορίζουν τις πιθανότητες που χρειάζονται για να λειτουργήσουν οι ευριστικές (ενότητα 3.2) που θα χρησιμοποιηθούν στον αλγόριθμο WalkSAT.

Στο πρώτο στάδιο του ο αλγόριθμος κάνει τυχαία ανάθεση τιμών (true/false) στις μεταβλητές του προβλήματος. Ο λόγος που γίνεται αυτό είναι για να έχουμε ένα μοντέλο στον χώρο καταστάσεων όπου θα εφαρμοστεί ο αλγόριθμος τοπικής αναζήτησης δηλαδή να γίνουν flip οι μεταβλητές και να έχουμε μεταβάσεις σε γειτονικές καταστάσεις μέχρι να βρεθεί λύση ή να ξεπεραστούν τα όρια βημάτων και επαναλήψεων.

Στη συνέχεια έχουμε την εκτέλεση του βασικού κώδικα, ο οποίος αποτελείται από ένα βρόγχο με αριθμό επαναλήψεων ίσο με το μέγιστο αριθμό των flips που δίνουμε στις εισόδους του αλγορίθμου. Όπως προαναφέρθεικε σε κάθε επανάληψη ο αλγόριθμος επιλέγει τυχαία ένα false clause και σύμφωνα με την ευριστική select() που χρησιμοποιείται προχωρά στην επιλογή της μεταβλητής που θα αλλάξει τιμή.

Στην περίπτωση εύρεσης λύσης ο αλγόριθμος τερματίζει και επιστρέφει το μοντέλο μεταβλητών το οποίο ικανοποίησε την πρόταση. Αν οι επαναλήψεις του βρόγχου που προαναφέρθηκε ολοκληρωθούν και δεν έχει βρεθεί το μοντέλο που ικανοποιεί την πρόταση τότε γίνεται επανεκκίνηση του αλγορίθμου με καινούρια αρχικοποίηση μεταβλητών. Ο αριθμός των επανεκκινήσεων καθορίζεται από το όρισμα των Max-Tries.

Ο αλγόριθμος τερματίζει οριστικά σε περίπτωση που δεν βρεί λύση όταν ξεπεραστούν όλες οι δυνατές επανεκκινήσεις.

3.2 Ευριστικές

Όπως είδαμε και στο κεφάλαιο με τους αλγορίθμους τοπικής αναζήτησης η απόδοση ενός τέτοιου αλγορίθμου εξαρτάται από την ευριστική που θα χρησιμοποιήσει ο αλγόριθμος και μια παράμετρο θορύβου η οποία καθορίζει αν θα γίνει καποια βέλτιστη κίνηση ή θα γίνει μια κίνηση η οποία από την μία δεν είναι βέλτιστη (άπληστη) αλλά μπορεί να αποφύγει τοπικά ελάχιστα ή μέγιστα, αποφεύγοντας έτσι το μεγάλο μειονέκτημα του αλγορίθμου.

Σύμφωνα με την δημοσίευση [ΜΣΚ97] μελετούνται έξι ευριστικές για τον αλγόριθμο WalkSAT τις οποίες βλέπουμε αναλυτικά στη συνέχεια.

G: Με πιθανότητα p παίρνουμε μια τυχαία μεταβλητή από το επιλεγμένο clause ,

αλλιώς επιλέγουμε την μεταβλητή η οποια ελαχιστοποιεί τον αριθμό των false clauses , δηλαδή πια τη μεταβλητή του clause που μετά την αντιστροφή της έχει τα περισσότερα σωστά clauses. [1]

B: Με πιθανότητα p επιλέγουμε μια τυχαία μεταβλητή από το επιλεγμένο clause , αλλιώς θα επιλέξουμε την μεταβλητή η οποια ελαχιστοποιεί των αριθμό των clauses τα οποία ήταν αληθή και μετά την αντιστροφή της μεταβλητής έγιναν ψευδή.

SKC: Όπως και το προηγούμενο (B) αλλά δεν θα χρησιμοποιείται το στοχαστικό κομμάτι (δηλαδή με πιθανότητα p παίρνουμε μια τυχαία μεταβλητή) εφόσον υπάρχουν ακόμα μεταβλητές οι οποίες με την αντιστροφή τους κάνουν κάποια clauses από αληθές σε ψευδές. Η στοχαστική φύση του αλγορίθμου χρησιμοποιείται μόνο όταν υπάρχει τοπικό μέγιστο.

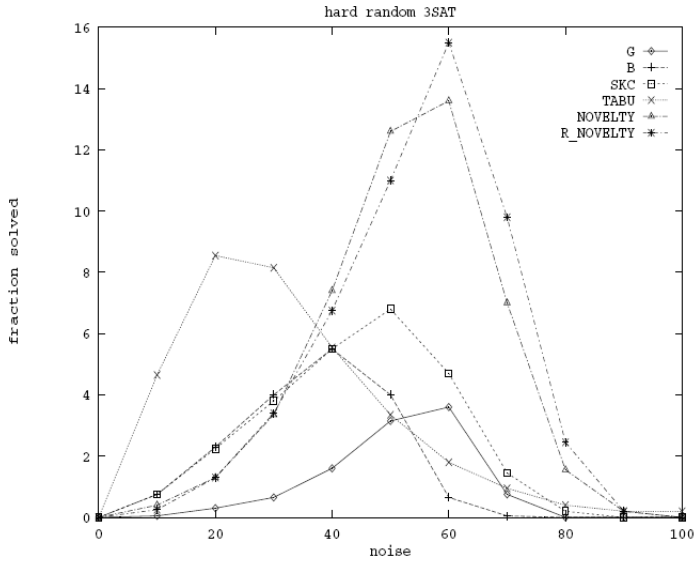
TABU: Στη στρατηγική αυτή επιλέγουμε μια μεταβλητή η οποία ελαχιστοποιεί τον αριθμό των false clauses . Σε κάθε βήμα απορρίπτουμε να αντιστρέψουμε μια μεταβλητή η οποία εχει αντιστραφεί σε συγκεκριμένο αριθμό βήματων. Σε περίπτωση που και οι τρεις μεταβλητές είναι στη λίστα του TABU τότε επιλέγουμε ένα άλλο clause . Στην περίπτωση που όλες οι μεταβλητές σε όλα τα μη ικανοποιήσιμα clauses είναι στη λίστα τότε αγνοούμε την λίστα και αλλάζουμε μια μεταβλητή η οποία ελαχιστοποιεί τον αριθμό false clauses.

Novelty: Η στρατηγική εδώ λειτουργεί λαμβάντας υπόψιν δύο παραμέτρους στον υπολογισμό του score. Η πρώτη παράμετρος είναι όπως την SKC δηλαδή τον αριθμό των clauses που ήταν true και έγιναν false και η δεύτερη παράμετρος το αντίθετο, τα clauses που ήταν false και έγιναν true. Η πρώτη παράμετρος ορίζεται και ως breakvalue και η δεύτερη ως makevalue. Η κατάταξη των μεταβλητών γίνεται με το score το οποίο είναι η τιμή makevalue - breakvalue. Στη συνέχεια γίνεται έλεγχος της καλύτερης και της δεύτερης καλύτερης μεταβλητής και αν η καλύτερη μεταβλητή δεν είναι η πιο πρόσφατη αντιστραμμένη μεταβλητή τότε την επιλέγουμε αλλιώς με πιθανότητα p επιλέγουμε μία από τις δύο, καλύτερη και δεύτερη καλύτερη.

R-Novelty: Είναι παρόμοια με την προηγούμενη στρατιγική αλλά διαφέρει στην περίπτωση που η μεταβλητή είναι η πιο πρόσφατη αντιστραμμένη. Σε αυτή την περίπτωση διαφέρει ο τρόπος που επιλέγουμε την μεταβλητή προς αντιστροφή. Έστω n η διαφορά από την καλύτερη και την δεύτερη καλύτερη, τότε αν $n \geq 1$ επιλέγουμε την μεταβλητή ανάλογα σε πια από τις τέσσερις περιπτώσεις ανοίκει σε σχέση με μια πιθανότητα p και το n.

[1] Σε αυτό το σημείο αξίζει να αναφέρουμε τον αλγόριθμο $GreedySAT(GSAT)$ ο οποίος χρησιμοποιεί την ευριστική μόνο χωρίς τη στοχαστική φύση,δηλαδή $p = 0$.

1. Όταν $p < 0.5$ και $n > 1$ επιλέγουμε την καλύτερη μεταβλητή.
2. Όταν $p < 0.5$ και $n = 1$ επιλέγουμε την δεύτερη καλύτερη με πιθανότητα $2p$ αλλιώς την καλύτερη.
3. Όταν $p \geq 0.5$ και $n = 1$ επιλέγουμε τη δεύτερη καλύτερη.
4. Όταν $p \geq 0.5$ και $n > 1$ επιλέγουμε τη δεύτερη καλύτερη με πιθανότητα $2(p - 0.5)$ αλλιώς την καλύτερη.



Σχήμα 3.2: Απόδοση Ευριστικών

Στο σχήμα 3.2 παρουσιάζεται ένα γράφημα το οποίο συγκρίνει τις ευριστικές ως προς τα προβλήματα που μπορεί κάθε μια να λύσει και στην τιμή της πιθανότητας κατά την οποία κάθε ευριστική έχει το καλύτερο αποτέλεσμα. Από τις έξι ευριστικές που αναφέρονται πιο πάνω θα ασχοληθούμε και θα μελετήσουμε τις τρεις, G, SKC και Novelty. Όπως φαίνεται και στο σχήμα 3.2 καλύτερη από τις τρεις παρουσιάζεται η Novelty και ακολουθούν οι SKC και G. Στην επόμενη ενότητα παρουσιάζεται η μοντελοποίηση του αλγορίθμου στη γλώσσα προγραματισμού C. Συγκεκριμένα υλοποιήθηκαν οι τρείς ευριστικές G, SKC και Novelty , μελετήθηκαν και συγκριθήκαν ως προς την απόδοση τους. Τα αποτελέσματα παρουσιάζονται στην ενότητα 3.4.

Στην επόμενη ενότητα αναφερόμαστε στην μοντελοποίηση του αλγορίθμου σε γλώσσα προγραμματισμού C κατά την οποία έγινε και η υλοποίηση των τριών ευριστικών για τις οποίες έγινε μελέτη ως προς την απόδοση τους. Τα αποτελέσματα παρουσιάζονται στην ενότητα 3.4.

Στη συνέχεια θα παρουσιάσουμε τις βασικές διαφορές των τριών ευριστικών που μελετάμε σε θέμα υλοποίησης. Στον πίνακα 3.1 παρουσιάζουμε τα διάφορα χαρακτηριστικά που μπορεί κάποια ευριστική να λάβει υπόψην της και ποιες από τις τρεις ευριστικές τα εκμεταλλεύονται.

	WalkSAT/G	WalkSAT/SKC	WalkSAT/Novelty
Αριθμός false clauses	√	√	√
Αριθμός breakvalues		√	√
Αριθμός makevalues			√
Ιστορικό των μεταβλητών ως προς τα flips			√

Πίνακας 3.1: Χαρακτηριστικά ευριστικών για τον αλγόριθμο WalkSAT

Στον πίνακα αυτό παρατηρούμε ότι η Novelty λαμβάνει πολυ περισσότερα κριτήρια για την επιλογή μεταβλητής και είναι και ο κύριος λόγος για την καλύτερη απόδοση που έχει σε σχέση με τις υπόλοιπες.

3.3 Μοντελοποίηση Αλγορίθμου σε C

Έγινε η βασική υλοποίηση του αλγορίθμου σε γλώσσα προγραμματισμού C. Στο λογισμικό υλοποιήθηκε ο βασικός πυρήνας του WalkSAT μαζί με τις τρεις ευριστικές που θέλουμε να μελετήσουμε και να συγκρίνουμε. Το πρόγραμμα δέχεται ορίσματα την πρόταση CNF, τον μέγιστο αριθμό Flips που θέτουμε και τον αριθμό των Max-Tries όπως περιγράφεται στην παράγραφο 3.1. Η πιθανότητα (p) για τις ευριστικές δεν δίνεται σαν όρισμα αλλά χρησιμοποιείται η βέλτιστη τιμή όπως φαίνεται και στο σχήμα 3.2.

Η CNF βρίσκεται σε αρχείο κειμένου και είναι σε μορφή Dimacs [διμ] όπως φαίνεται και στο σχήμα 3.2.

Οι πρώτες γραμμές του αρχείου περιέχουν χρήσιμη πληροφορία για την πρόταση.Η πληροφορία αυτή αποτελείται από τον αριθμό των variables και clauses που έχει το συγκεκριμένο πρόβλημα. Στη συνέχεια έχουμε την πληροφορία για κάθε clause η οποία αποτελείται από τρεις προσημασμένους αριθμούς [2] που αφορούν την τιμή κάθε μεταβλητής. Το αρνητικό πρόσημο καθορίζει αν η μεταβλητή βρίσκεται στην αρνητική μορφή της.

Το δεύτερο όρισμα του προγράμματος, μέγιστο αριθμό Flips, χρησιμοποιείται στον βασικό πυρήνα και έχει την ίδια χρησιμότητα για κάθε μία ευριστική, οπως και το τρίτο

[2] Το προβλήματα που μελετάμε είναι $3 - SAT$

```
c some comments
c like clause length = 3
p cnf 225 960
-133 217 -81 0
-202 159 -129 0
-185 -52 -21 0
...
158 -65 -56 0
```

Πίνακας 3.2: Παράδειγμα αρχείου με μορφή Dimacs

όρισμα, Max-Tries.

Για την ανάγνωση του αρχείου χρησιμοποιείται μια συνάρτηση η οποία παίρνει ως όρισμα το όνομα του αρχείου. Αρχικά βρίσκει τον αριθμό των Variables και Clauses και στη συνέχεια διαβάζει την πληροφορία για κάθε Clause και την αποθηκεύει στο αντίστοιχο διάνυσμα που έχει δεσμευτεί στην μνήμη. Έπειτα δημιουργείται ένα διάνυσμα το οποίο περιέχει τιμές 0 και 1 και έχει μέγεθος όσα τα Variables του προβλήματος. Το διάνυσμα αυτό αρχικοποιείται τυχαία σε κάθε επανεκκίνηση και σε κάθε flip μεταβλητής ενημερώνεται και αλλάζει τιμή στην επιλεχθείσα μεταβλητή σύμφωνα με την ευριστική που χρησιμοποιούμε.

Συμφωνα με τον αλγόριθμο το πρόγραμμα τερματίζει όταν βρει λύση ή όταν ξεπεραστεί το όριο flips σε συνδιασμό με τον αριθμό Max-Tries.Στην συγκεκριμένη μοντελοποίηση όταν βρεθεί λύση και δεν έχουν ξεπεραστεί τα Max-Tries το πρόγραμμα δεν τερματίζει οριστικά αλλά συνεχίζει με νέα αρχικοποίηση και προσπάθια εύρεσης λύσης. Ο λόγος που γίνεται αυτό είναι για να μπορέσουμε να πάρουμε στατιστικά ως προς τον αριθμό flips και να εκτιμήσουμε τον μέσο αριθμό max-tries που χρειάζεται κάθε ευριστική για να βρει λύση όπως θα δούμε και στην ενότητα 3.4.

3.4 Πειραματικά Αποτελέσματα

Το λογισμικό που περιγράφεται στην παράγραφο 3.3 χρησιμοποιείται για τα πειράματα μέσα από τα οποία θα γίνει σύγκριση των τριών ευριστικών.

Τα πειράματα έγιναν με πέντε Benchmarks από το SATlib του Πανεπιστημίου British Columbia (UBC) [σατ]. Τα πέντε αυτά Benchmarks έχουν λόγο $Clauses/Variables = 4.26$ το οποίο είναι και το κρίσιμο σημείο δύσκολων SAT προβλημάτων όπως είδαμε και στην παράγραφο 2.2. Συγκεκριμένα κάθε benchmark έχει 225 μεταβλητές και 990

clauses. Για την εκτέλεση των πειραμάτων και την λήψη των μετρήσεων κάθε πείραμα εκτελέστηκε 10 φορές με 10 επαναλήψεις κάθε φορά. Οι 10 επαναλήψεις καθορίζονται από το όρισμα των Max-Tries και ο μέγιστος αριθμός των flips είναι διαφορετικός για κάθε πρόβλημα ανάλογα με την δυσκολία του και προέκυψε μέσα από δοκιμές. Για κάθε ευριστική κρατάμε το μέγιστο αριθμό flips ίδιο και έτσι μπορούμε να συγκρίνουμε τις ευριστικές μεταξύ τους. Στα πρώτα δύο προβλήματα έχουμε 10000 flips, στα επόμενα δύο 100000 και στο τελευταίο 5000000. Η διαφορετική δυσκολία στα Benchmarks αν και έχουν ίδιο λόγο clauses/variables προκύπτει από την κατανομή των μεταβλητών μέσα στα clauses.

Η σύγκριση των ευριστικών έγινε σε δύο επίπεδα. Το πρώτο επίπεδο σύγκρισης αφορά των αριθμό των flips που χρειάζεται κάθε ευριστική για να βρει λύση και το δεύτερο επίπεδο είναι το ποσοστό επιτυχίας [3] της κάθε ευριστικής. Στον πίνακα 3.3 φαίνονται συνοπτικά τα αποτελέσματα για τα πειράματα.

	WalkSAT/G		WalkSAT/SKC		WalkSAT/Novelty	
Benchmarks	Solved %	Flips	Solved %	Flips	Solved %	Flips
uf225-087	100	2763.03	100	2321.79	100	1163.41
uf225-026	100	3330.34	100	2684.20	100	1387.09
uf225-028	94	29088.89	96	23354.12	100	10156.05
uf225-091	93	27771.9	96	25048.22	100	10887.16
uf225-039	0	-	23	2610977.01	93	1278917.95

Πίνακας 3.3: Σύγκριση ευριστικών για τον αλγόριθμο WalkSAT

Ανάλυση αποτελεσμάτων πίνακα 3.3

Παρατηρούμε ότι η ευριστική SKC χρειάζεται μεγαλύτερο από διπλάσιο αριθμό flips σε σχέση με την Novelty και το ποσοστό επιτυχίας είναι μικρότερο στα προβλήματα μέτριας δυσκολίας και αρκετά μικρότερο σε προβλήματα μεγαλύτερης δυσκολίας. Η ευριστική G φαίνεται να έχει τα χειρότερα αποτελέσματα σε σχέση με τις άλλες δύο. Παρατηρείται ακόμα ότι στα δύσκολα προβλήματα δεν βρίσκει λύση και στα υπόλοιπα βρίσκει λύση σε μεγαλύτερο αριθμό βημάτων σε σχέση με τη Novelty και την SKC.

Αναλυτικότερα, στα πρώτα δύο προβλήματα (uf225-087,uf225-026) και οι τρεις ευριστικές έχουν 100% επιτυχία αλλά διαφέρουν στον αριθμό των flips που κάνουν για

[3] Στην περίπτωση αυτή ορίζουμε επιτυχημένη προσπάθεια όταν ο αλγόριθμος καταφέρει να βρεί λύση μέσα στον μέγιστο αριθμό από flips σε ένα try

εύρεση λύσης. Η Novelty να είναι πολύ καλύτερη από τις άλλες δύο και απαιτεί μικρότερο αριθμό βημάτων. Στα δύο αυτά προβλήματα παρατηρείται η SKC και η G να έχουν πολύ μικρή διαφορά με την SKC λίγο καλύτερη.

Στη συνέχεια μελετάμε δύο προβλήματα μεγαλύτερης δυσκολίας, τα (uf225-028,uf225-091), στα οποία βλέπουμε και πάλι την Novelty να έχει ποσοστό επιτυχίας 100% και τις άλλες δύο να έχουν μεν καλό ποσοστό, αλλά μικρότερο. Όσο αφορά τα flips η Novelty έχει λιγότερα από τα μισά που χρειάστηκε η SKC και ακόμα λιγότερα από την G.

Τέλος έχουμε το (uf225-039) το δυσκολότερο από τα πέντε προβλήματα που μελετήσαμε. Παρατηρούμε ότι καμια από τις τρεις ευριστικές δεν έχει 100% επιτυχία στην άμεση έυρεση λύσης. Η Novelty έχει ένα ποσοστό επιτυχίας 93% σε σχέση με την SKC της οποίας το ποσοστό κειμένεται γύρω στο 23%. Η ευριστική G δεν μπορεί να βρει λύση σε αυτά τα προβλήματα. Όπως και στα άλλα προβλήματα μικρότερης δυσκολίας από το τελευταίο παρατηρούμε ότι η Novelty είναι πάλι πολύ καλύτερη από την SKC όσο αφορά τον αριθμό των flips που χρειάζονται για την εύρεση λύσης.

Το ποσοστό επιτυχίας στα προβλήματα που μελετήσαμε καθορίζει των αριθμό των Max-Tries του αλγορίθμου. Στο δυσκολότερο πρόβλημα uf225-039 βλέπουμε την Novelty να μπορει να βρίσκει λύση σε 1.08 όπως φάινεται και στον πίνακα 3.4 προσπάθειες σε σχέση με την SKC η οποία βρίσκει λύση σε 4.76 προσπάθειες. Συνδυάζοντας τον

	Mean Tries	Total Flips
Novelty	1.08	1375180.6
SKC	4.76	12433223.85

Πίνακας 3.4: Novelty Vs. SKC για το πρόβλημα uf225-039

uf225-039	Flips
Software [KM05]	2726196
Hardware [KM05]	2690011
Our SKC (Software)	2610977
Our Novelty (Software)	1278917

Πίνακας 3.5: Σύγκριση flips για το πρόβλημα uf225-039

μέσο αριθμό flips για Novelty και SKC με τον αριθμό των Max-Tries που αναφέραμε πιο πάνω έχουμε την Novelty να έχει μέσο αριθμό flips 1375180.6 και την SKC

12433223.85. Συνοψίζοντας τα πιο πάνω παρατηρούμε ότι χρησιμοποιώντας την Novelty έχουμε $SpeedUp = 9.04$ ως προς την χρήση της SKC.

Στον πίνακα 3.5 γίνεται σύγκριση των flips μεταξύ του λογισμικού που υλοποιήσαμε και των αποτελεσμάτων σύμφωνα με την δημοσίευση [KM07] για το δυσκολότερο πρόβλημα, uf225-039.

Κεφάλαιο 4

Αναδιατασσόμενη λογική και SAT

4.1 Μερική Αναδιάταξη

Η αρχική προσέγγιση της εργασίας αυτής η οποία ξεκίνησε από το μάθημα των Ενσωματομένων Συστημάτων είναι γύρω στην μερική αναδιάταξη και αν μπορεί να εφαρμοστεί για τον αλγόριθμο WalkSAT. Στο κεφάλαιο αυτό θα κάνουμε αναφορά σε σχετικές εργασίες γύρω από την δυναμική αναδιάταξη και στη συνέχεια θα παρουσιάσουμε την δική μας προσέγγιση και τα προβλήματα που συναντήσαμε.

4.1.1 Σχετική βιβλιογραφία

Στην βιβλιογραφία υπάρχουν δύο δημοσιεύσεις οι οποίες αναφέρουν την υλοποίηση αλγορίθμων WalkSAT και GreedySAT σε αναδιατασσόμενη λογική σε συνδυασμό με δυναμική αναδιάταξη.

Real-time reconfigurable WSAT

Η πιο πρόσφατη δημοσίευση που έγινε το 2003 [ΨΩΗ03], η οποία προσπαθεί δυναμικά να αλλάζει διάφορα προβλήματα σε πραγματικό χρόνο. Συγκεκριμένα αυτό που κάνει είναι να αλλάζει τις τιμές στους registers οι οποίες περιγράφουν τις προτάσεις CNF ώστε κάθε νέο πρόβλημα που πρόκειτε να λυθεί να μην χρειάζετε επανασύνθεση της σχεδίασης.

Runtime reconfiguration for GSAT

Μια παλιότερη δημοσίευση του τόσο [ΨΣΛΛ99], σκοπός της όπως και στην [ΨΩΗ03] είναι να αλλάζει τα διάφορα προβλήματα χωρίς να χρειάζεται επανασύνθεση όλης της

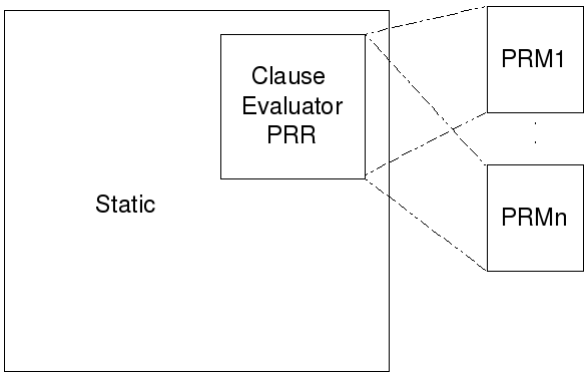
σχεδίασης. Αυτή η δουλειά είναι πιο κοντά στην δυναμική αναδιάταξη σε σχέση με την [ΨΩΗ03].

4.1.2 Η δική μας προσέγγιση

Στο σημείο αυτό θα παρουσιάσουμε τρεις προσεγγίσεις που έγιναν για την χρήση της δυναμικής αναδιάταξης στην αρχιτεκτονική μας. Οι πρώτες δύο αφορούν την πρόταση CNF και η άλλη τις ευριστικές συναρτήσεις.

Διάσπαση της CNF σε μικρότερες

Σε πρώτη φάση έγινε η υλοποίηση του WalkSAT χρησιμοποιώντας δυναμική αναδιάταξη στον Clause Evaluator . Το block diagram της σχεδίασης φαίνεται στο σχήμα 4.1.

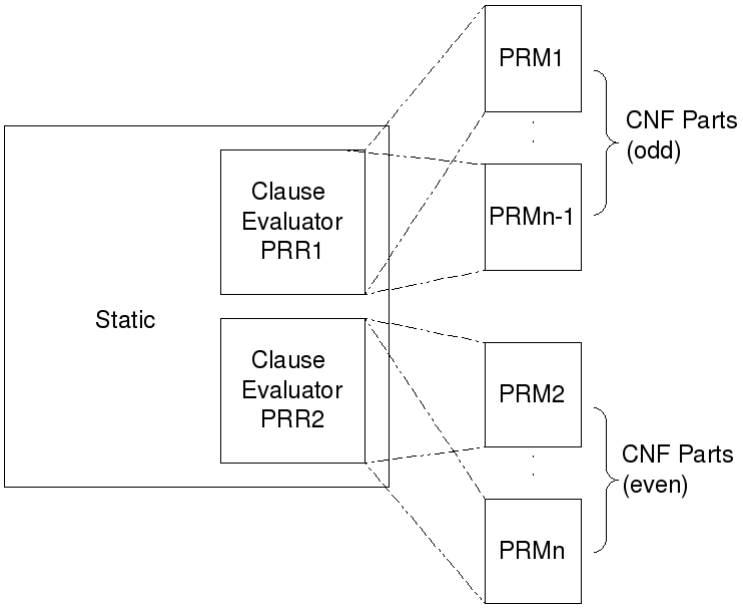


Σχήμα 4.1: Block Diagram with one PRR

Το PRR της υλοποίησης είναι ο Clause Evaluator στο οποίο θα αντιστοιχούν N PRMs τα οποία θα αποτελούν ολόκληρη τη CNF. Τα modules αυτά θα φορτώνονται διαδοχικά και θα γίνεται η επεξεργασία σε κάθε ένα, και τα αντίστοιχα (μερικά) αποτελέσματα θα αποθηκέυονται σε καταχωρητές. Η πληροφορία που αποθηκεύετε για κάθε PRM είναι ο συνολικός αριθμώς των false clauses ο οποίος θα αυξάνεται κάθε φορά ώστε στο τέλος να έχουμε τον τελικό αριθμό από τα false clauses. Επίσης για κάθε PRM κρατάμε αποθηκευμένο ένα τυχαίο false clause ώστε στο τέλος να επιλέξουμε και πάλι τυχαία ένα. Μετά από την επεξεργασία όλων των modules θα παίρνεται η απόφαση για το επόμενο βήμα του αλγορίθμου.
Σε αυτή τη έκδοση η οποία υποστιρίζει μικρό αριθμό clauses και variables και συγκεκριμένα 32 clauses και 8 variables η δυναμική αναδιάταξη αρχίζει όταν έχουμε τα αποτελέσματα από τα πρώτα 8 clauses και δεν χρειαζόμαστε άλλο το συγκεκριμένο τμήμα της CNF . Υπάρχει ένας μικρός χρόνος επικάλυψης μεταξύ επεξεργασίας και φόρτωσης του επόμενου PRM .

Προβλήματα:
Ο χρόνος επικάλυψης που προαναφέρω είναι πολύ μικρότερος από το χρόνο δυναμικής αναδιάταξης. Γενικά το μεγαλύτερο πρόβλημα αυτής της προσέγγισης θεωρητικά είναι ο μεγάλος αριθμός εναλλαγών (δυναμική αναδιάταξη) των PRMs. Ο αλγόριθμος αυτός είναι ευριστικός, αυτό σημαίνει ότι για να συγκλίνει χρειάζονται πολλές επαναλήψεις της τάξης του ενός εκατομυρρίου σε μεγάλα προβλήματα. Αυτό σημαίνει ότι θα έχουμε N εκατομμύρια δυναμικές αναδιατάξεις, πράγμα το οποίο είναι σημαντικά χρονοβόρο.

Μια άλλη σχεδίαση είναι να χρησιμοποιείσουμε δύο PRRs , δύο Clause Evaluators με στόχο να κάνουμε δυναμική αναδιάταξη του ενός κατά την διάρκεια επεξεργασίας του άλλου. Το σχήμα 4.2 δείχνει τα δύο PRRs και πια PRMs αντιστοιχούν σε κάθε ένα.



Σχήμα 4.2: Block Diagram with two PRRs

Προβλήματα:
Όπως και στην περίπτωση με ένα PRR έχουμε και εδώ το πρόβλημα με τις επαναλήψης και τις πολλές δυναμικές αναδιατάξης της σχεδίασης μας. Αν και έχουμε μεγαλύτερη επικάλυψη μεταξύ χρόνου επεξεργασίας και χρόνου δυναμικής αναδιάταξης το πρόβλημα παραμένει.

Εφαρμογή στις ευριστικές

Μια άλλη προσέγγιση σε θεωρητικό επίπεδο είναι να εφαρμοστή η δυναμική αναδιάταξη στις ευριστικές συναρτήσεις. Πιο συγκεκριμένα είναι να μπρούμε να αλλάζουμε σε πραγματικό χρόνο λειτουργίας την ευριστική συνάρτηση που χρησιμοποιείται. Όπως διαπιστώσαμε από την μελέτη του αλγορίθμου και την απόδοση των ευριστικών συναρτήσεων από τη στιγμή που θα υλοποιηθεί και θα χρησιμοποιειθεί μια ευριστική δεν υπάρχει λόγος να γίνει αλλαγή ειδικά με μια χειρότερης απόδοσης.

4.2 Στατικές Υλοποιήσεις

Στο σημείο αυτό θα παρουσιάσουμε σχετικές δουλειές γύρω από τα προβλήματα SAT και στατικές υλοποιήσεις σε FPGA. Το 2004, στο [SdBF04] γίνεται αξιολόγιση των υπάρχοντων συστημάτων με ιδιαίτερη έμφαση στις [SYSN], [Zho], [PM], [DP], [LSW$^+$] και [SF].

Το 2005, 2006 και 2007 υπαρχουν τρεις δημοσιέυσεις οι οποίες σκοπό έχουν να λύνουν μεγάλα προβλήματα SAT. Στον πίνακα 4.1 φαίνονται κάποια βασικά χαρακτηριστικά των δημοσιέυσεων οι οποίες παρουσιάζονται αναλυτικότερα στη συνέχεια.

Ref	Device	Slices	BRAMs	SAT	Clauses/Variables
[KM05]	XC2V6000	45%	5	WSAT/[G,SKC]	1024/256
[KM06]	XC2V6000	51%	95%	WSAT/SKC	8500/2000
[KM07]	XC2V6000	88%	97%	WSAT/SKC	128K/32K

Πίνακας 4.1: Χαρακτηριστικά δημοσιεύσεων [KM05],[KM06],[KM07]

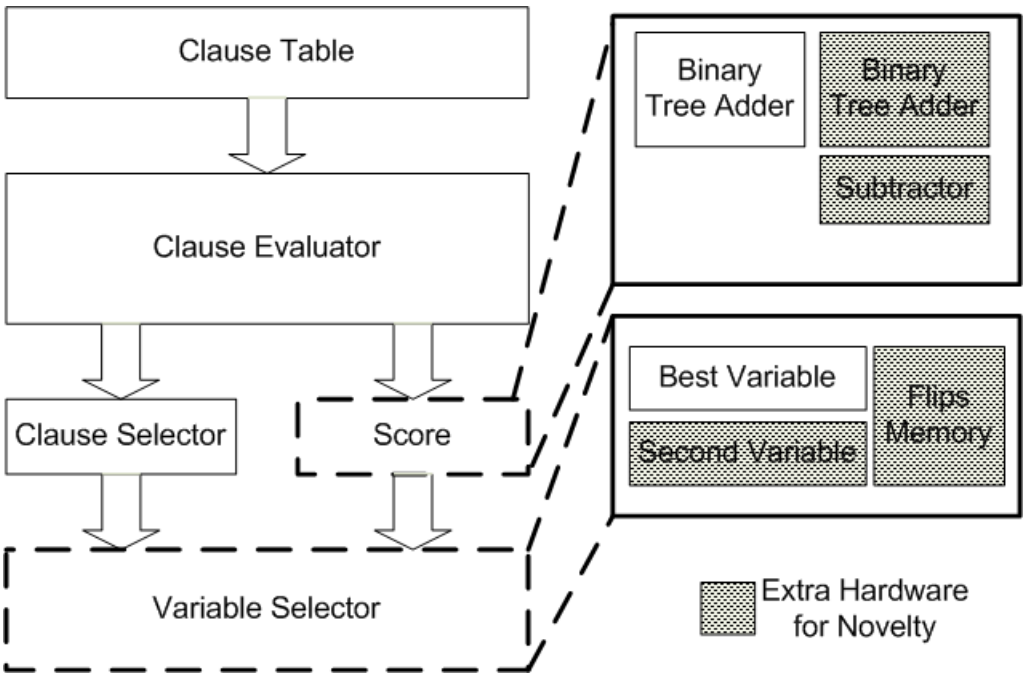
[KM05] Γίνεται υλοποίηση του WalkSAT με χρήση των ευριστικών G και SKC. Ο αριθμός των clauses και variables που μπορεί να υποστηρίξει είναι 1024 και 256. Η υλοποίηση έγινε σε Virtex II, XC2V6000 και καταναλώνει 45% των slices και 5% των BRAMs. Αυτή η δημοσίευση έχει άμεση σχέση με την παρούσα διπλωματική όπου γίνεται υλοποίηση της Novelty και σύγκριση των υλοποιήσεων ως προς την απόδοση.

[KM06] Γίνεται βελτίωση της [KM05] στον τρόπο που ελέγχονται τα clauses ώστε να χρειάζονται λιγότεροι πόροι αλλά να μπορεί να υποστηρίξει περισσότερα clauses και variables. Χρησιμοποιείτε η ίδια Virtex στην οποία καταναλώνονται το 51% των slices και 95% των BRAMs και μπορεί να υποστηρίξει μέχρι 8500 clauses και 2000 variables.

[KM07] Γίνεται ένας συνδιασμός εξωτερικών μνημών με της προηγούμενες σχεδιάσεις ώστε να μπορεί να λύνει πολύ μεγάλα προβλήματα της τάξεως των $128K$ clauses και $32K$ variables. Η σχεδίαση είναι και πάλι σε Virtex II, XC2V6000 και καταναλώνει 88% των slices και 97% των BRAMs.

Μια άλλη σχετική έρευνα σε αλγορίθμους επίλυσης προβλημάτων είναι η [ΔΤΨΖ08].

4.3 Υλοποίηση Novelty σε υλικό



Σχήμα 4.3: Διάγραμμα μονάδων υλικού για τις ευριστικές

Μελετόντας τις διάφορες εργασίες και λαμβάνοντας υπόψιν την σύγκριση στις τρις ευριστικές αποφασίσαμε να υλοποιήσουμε την ευριστική novelty σε υλικό η οποία παρουσιάζεται καλύτερη από τις άλλες και δεν έχει γίνει υλοποίηση της στο παρελθόν. Η αρχιτεκτονική που ακολουθήσαμε στην υλοποίηση είναι παρόμοια με την [KM05] και αυτό συμβαίνει γιατί υλοποιούν τον αλγόριθμο WalkSAT/SKC. Στην υλοποίηση που έγινε με την χρήση δυναμικής αναδιάταξης υλοποιήθηκε η ευριστική G. Στο σχήμα 4.3 παρουσιάζουμε τα κυριότερα σημεία της αρχιτεκτονικής στα οποία φαίνονται οι βασικές διαφορές των τριών ευριστικών όσο αφορά το υλικό.

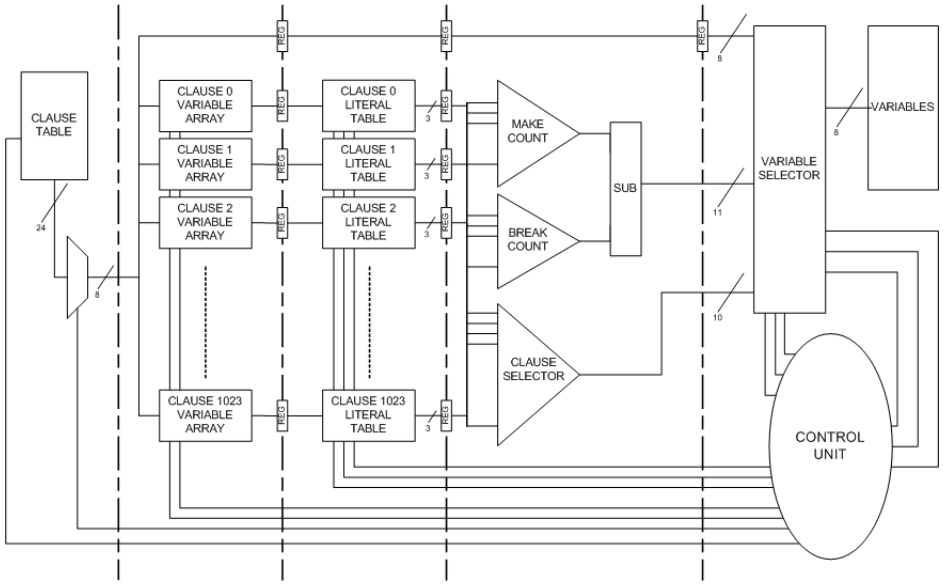
Οι κυριότερες διαφορές παρατηρούντε στην μονάδα του score και στην μονάδα επιλογής μεταβλητής. Στο κεφάλαιο (5 γίνεται αναλυτική παρουσίαση της αρχιτεκτονικής για την υλοποίηση του Walksat/Novelty.

Κεφάλαιο 5

Υλοποίηση σε Υλικό

5.1 Αρχιτεκτονική **WalkSAT**

Στο σχήμα 5.1 παρουσιάζεται το διάγραμμα του αλγορίθμου. Όπως μπορούμε να διακρίνουμε από το σχήμα η αρχιτεκτονική μας αποτελείται από 5 βασικά στάδια ομοχειρίας τα οποία θα αναλυθούν στην ενότητα 5.2.



Σχήμα 5.1: Αρχιτεκτονική WalkSAT/Novelty

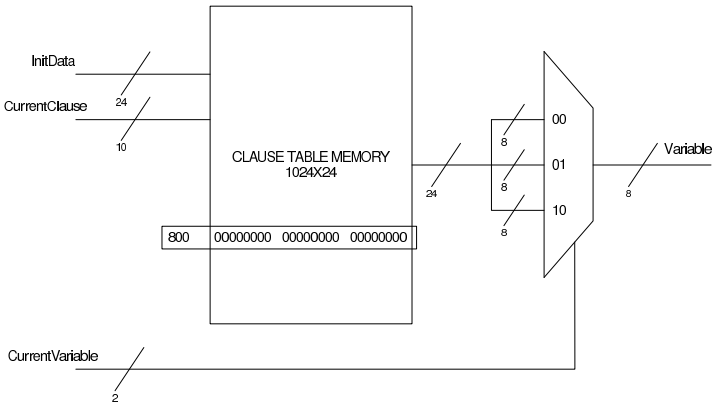
Τα δεδομένα του προβλήματος μετά την αρχικοποίηση βρίσκονται μέσα στο Clause Table. Στη συνέχεια έχουμε τις βασικές μονάδες του Clause Evaluator ο οποίος αποτελείται από clauses ανάλογα με το μέγεθος της σχεδίασης. Όπως θα δούμε τα clauses

κρατούν τις τιμές των μεταβλητών που τα αποτελούν στα Variable Arrays καθώς και τις τιμές των literals σε κάθε προσορινό flip μεταβλητών στο Literal Value Table.Στη συνέχεια θα αναλυθούν οι μονάδες του Score και της επιλογής μη ικανοποιήσιμου clause καθώς και η μονάδα επιλογής μεταβλητής προς αντιστροφή.

5.2 Ανάλυση Σταδίων Ομοχειρίας

5.2.1 1ο Στάδιο - Ανάκτηση μεταβλητής

Σε αυτό το στάδιο γίνεται ανάγνωση της μνήμης που περιέχει την πληροφορία των μεταβλητών που αποτελούν το κάθε Clause. Τα δύο βασικά στοιχεία που το αποτελούν είναι το Clause Table και ένας πολυπλέκτης στην διάταξη που φαίνεται στο σχήμα 5.2. Τα δεδομένα που εισέρχονται στο στάδιο αυτό είναι η πληροφορία της CNF κατά την διάρκεια αρχικοποίησης του συστήματος. Η μοναδική έξοδος του σταδίου είναι η μεταβλητή η οποία προωθείται στο επόμενο στάδιο και έχει πλάτος σε bits ανάλογο με τον αριθμό των μεταβλητών της σχεδίασης, συγκεκριμένα $\log_2(\#Variables)$.



Σχήμα 5.2: Clause Table

Clause Table

Το Clause Table είναι μια μνήμη η οποία κρατά την πληροφορία της CNF όπως προαναφέρεται. Είναι μια μνήμη με πλάτος $3 \times \log_2(\#Variables)$ bit και βάθος όσος ο αριθμός των Clauses. Κάθε ποσότητα των $3 \times \log_2(\#Variables)$ bit περιέχει την πληροφορία ενός clause , δηλαδή τις τρεις μεταβλητές που το αποτελούν. Θεωρούμε ότι το κάθε clause αποτελείται από τις μεταβλητές V_0, V_1 και V_2. Τα $\log_2(\#Variables)$ λιγότερα σημαντικά bit αντιστοιχούν στην V_0, τα επόμενα οκτώ στην V_1 και τα περισσότερο σημαντικά στην V_2.

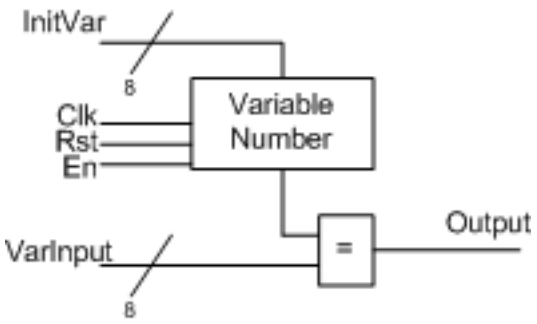
Η μνήμη έχει είσοδο τα δεδομένα, $3 \times \log_2(\#Variables)$ bit , που χρησιμοποιούνται στην αρχικοποίηση, την διεύθυνση, $\log_2(\#Clauses)$ bit, η οποία κατά την διάρκεια εκτέλεσης του αλγορίθμου αντιστοιχεί στο τυχαία επιλεγμένο μη ικανοποιήσιμο clause καθός και τα σήματα ελέγχου τα οποία είναι το ρολόι και η ενεργοποίηση εγγραφής.

Πολυπλέκτης 3 προς 1

Ο πολυπλέκτης αυτός είναι υπεύθυνος για την επιλογή της σωστής μεταβλητής η οποία προωθείται στο επόμενο στάδιο. Οι τρεις εισόδοι του πολυπλέκτη είναι οι τρεις ποσότητες των $\log_2(\#Variables)$ bit που έχουμε από την έξοδο του Clause Table. Το σήμα ελέγχου πλάτους 2 bit καθορίζει ποια από τις τρεις μεταβλητές θα πάει στην έξοδο και προέρχεται από την μονάδα ελέγχου. Οι τιμές που παίρνει το σήμα ελέγχου είναι $00, 01$ και 10 σε τρεις διαδοχικούς κύκλους.

5.2.2 2ο Στάδιο - Έλεγχος μεταβλητής

Η είσοδος στο στάδιο αυτό είναι η μεταβλητή που προέρχεται από το Clause Table. Όπως φαίνεται και στο σχήμα 5.3 η βαθμίδα αυτή αποτελείται από τα Variable Arrays τα οποία κρατούν πληροφορίες για τις μεταβλητές των αντίστοιχων clauses. Κάθε Variable Array αποτελείται από τρεις καταχωρητές μήκους $\log_2(\#Variables)$ bit , ένα για κάθε μεταβλητή του clause. Ο αριθμός των Variable Arrays είναι ίσος με τον αριθμό των clauses της σχεδίασης. Το περιεχόμενο των καταχωρητών γράφεται κατά την διαδικασία αρχικοποίησης και είναι σταθερό κατά την διάρκεια εκτέλεσης του αλγορίθμου.



Σχήμα 5.3: Variable Array

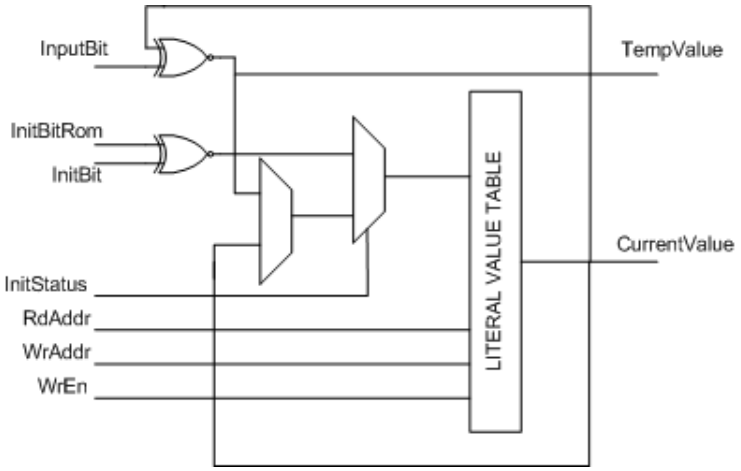
Η μεταβλητή που εισέρχεται στο στάδιο κατανέμεται σε όλα τα variable arrays όπου γίνεται σύγκριση με το περιεχόμενο του καταχωρητή τους. Η έξοδος του κάθε Variable Array είναι ένα σήμα πλάτους 3 bit , ένα bit για κάθε μεταβλητή του clause, το οποίο έχει τιμή 1 αν η μεταβλητή που θα μελετηθεί βρίσκεται στο συγκεκριμένο

clause και 0 αν δεν βρίσκεται. Η έξοδος του σταδίου αυτού είναι τα προαναφερθέντα σήματα τα οποία περνούν μέσα από ένα καταχωρητή που βρίσκεται μεταξύ του δευτέρου και τρίτου σταδίου.

Με τη χρήση των Variable Arrays μπορεί το κάθε clause να κρατά την πληροφορία για τις μεταβλητές του. Το πλεονέκτημα της μεθόδου αυτής είναι ότι αποφεύγεται η πολύπλοκη διασύνδεση των clauses με τις μεταβλητές και η ευελιξία όσο αφορά την αύξηση των μεταβλητών αυξάνοντας το μήκος των καταχωρητών στα Variable Arrays.

5.2.3 3ο Στάδιο - Literals

Όπως φαίνεται και από το σχήμα 5.4 το στάδιο αποτελείται από τα Literal Value Tables (LVT), μνήμες οι οποίες κρατούν την τρέχων κατάσταση του clause καθώς και τις 3ις προσωρινές τιμές που προκύπτουν από τα flips των μεταβλητών.



Σχήμα 5.4: Literal Value Table

Κάθε clause αποτελείται από τρία LVTs των οποίων η έξοδος τους μέσα από συνδυαστική λογική δίνει τα σήματα προς τις μονάδες του score και επιλογής του τυχαίου μη ικανοποιήσιμου clause όπως θα δούμε και πιο αναλυτικά στη συνέχεια.

Βασική είσοδος στα στοιχεία αυτά είναι το σήμα που προέρχεται από το 2ο στάδιο. Οι υπόλοιπες είσοδοι του συστήματος είναι τα σήματα ελέγχου τα οποία θα αναλυθούν στην συνέχεια. Έξοδος των LVTs είναι 3 σήματα που θα καθορίσουν την αξιολόγηση της μεταβλητής και την επιλογή του μη ικανοποιήσιμου clause για την επόμενη

επανάληψη. Τα δύο σήματα περιέχουν πληροφορία για το score (makevalue και breakvalue) και το άλλο για την τιμή των clauses που προκύπτει από την αντιστροφή της μεταβλητής.

Ο πυρήνας του LVT όπως φαίνεται και από το σχήμα 5.4 είναι μια Block RAM πλάτους 1 bit τεσσάρων θέσεων. Στην πρώτη θέση αντιστοιχεί η τιμή του literal που προκύπτει από το τρέχον μοντέλο μεταβλητών. Οι άλλες τρεις θέσεις αντιστοιχούν στην τιμή που παίρνει το literal κατά την διάρκεια μελέτης των τριών μεταβλητών του μη ικανοποιήσιμου clause.

Ανάλυση Σχήματος 5.4

Η μνήμη που χρησιμοποιείται είναι Dual port και ο λόγος είναι γιατί χρειαζόμαστε να κάνουμε ανάγνωση της τρέχων κατάστασης από την πρώτη θέση και παράλληλα να γράφουμε μια από τις άλλες τρεις. Σε τρεις διαδοχικούς κύκλους όπως θα δούμε και στο control η διεύθυνση ανάγνωσης είναι σταθερή και δείχνει στην πρώτη θέση ενώ διαδοχικά γράφεται η δεύτερη, τρίτη και τέταρτη. Όταν επιλεγεί η μεταβλητή η οποία θα γίνει μόνιμα flip θα διαβαστεί η αντίστοιχη θέση και διεύθυνση εγγραφής θα γίνει η πρώτη θέση της μνήμης.

Η μνήμη αυτή αρχικοποιείται κατά την διάρκεια αρχικοποίησης τους κυκλώματος. Τα σήματα που καθορίζουν την τιμή που θα γραφτεί στην μνήμη είναι το σήμα που δείχνει αν το literal βρίσκεται στην θετική ή αρνητική του κατάσταση σε συνδυασμό με ένα σήμα που προέρχεται από την τυχαία αρχικοποίηση των μεταβλητών.

Η τιμή που γράφεται στην μνήμη είναι η τιμή που διαβάζεται από την μνήμη σε συνδυασμό με την τιμή του bit εισόδου το οποίο προέρχεται από το Variable array και καθορίζει αν η μεταβλητή που μελετάται βρίσκεται στο συγκεκριμένο clause. Εκτός από αυτό ως εγραφή μπορεί να είναι κατευθείαν αυτό που διαβάζεται από την μνήμη το οποίο θα γραφτεί στην πρώτη θέση όπως προαναφέρεται μετά από το μόνιμο flip της επιλεχθέντας μεταβλητής ή το bit αρχικοποίησης. Ο έλεγχος καθορίζεται μέσα από τους δύο πολυπλέκτες που υπάρχουν πριν από την μνήμη.

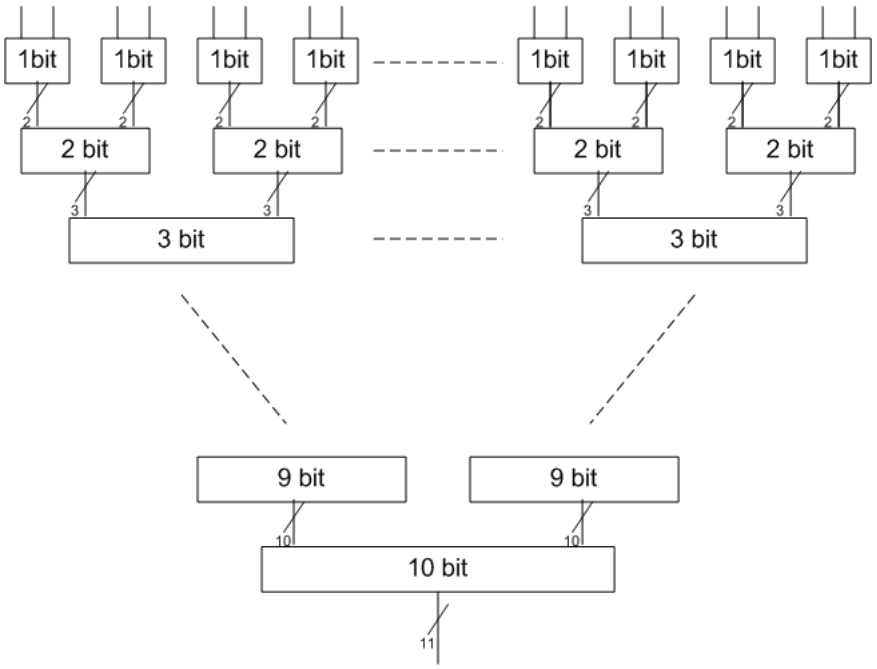
Η έξοδος του LVT είναι τα δύο σήματα όπως φαίνονται και στο διάγραμμα. Το πρώτο έχει την τιμή του literal σύμφωνα με το τρέχον μοντέλο μεταβλητών και το άλλο η προσωρινή τιμή μετά από κάθε flip. Οι έξοδοι από τα τρία LVTs που περιέχονται σε κάθε clause μέσα από μια συνδυαστική λογική μας δίνουν τις τιμές που καθορίζουν το score και η προσωρινή τιμή μέσα από πύλη or μας δίνει το σήμα για την μονάδα επιλογής τυχαίου clause.

5.2.4 4ο Στάδιο - Υπολογισμός Score

Στα δύο αυτά στάδια έχουμε δύο βασικές λειτουργίες. Η πρώτη είναι για τον υπολογισμό των scores και η δεύτερη για την επιλογή ενός clause το οποίο είναι μη ικανοποιήσιμο. Όπως είδαμε και στο σχήμα 5.1 στο στάδιο αυτό έχουμε 3ις βασικές μονάδες. Οι δύο μονάδες για το score οι οποίες αποτελούνται από ίδια σχεδίαση υλικού και η μονάδα για την επιλογή του false clause.

Makecount και Breakcount

Οι μονάδες αυτές αποτελούνται από ένα αθροιστή σε μορφή δέντρου (Binary Adder) ο οποίος δέχεται για είσοδο τα ανάλογα σήματα από το προηγούμενο στάδιο με πλάτος όσο και ο αριθμός των clauses η κάθε μονάδα, και υπολογίζει πόσα από αυτά έχουν την τιμή 1. Στο σχήμα 5.5 παρουσιάζεται το αναλυτικό διάγραμμα της μονάδας για την περίπτωση των 1024 clauses.



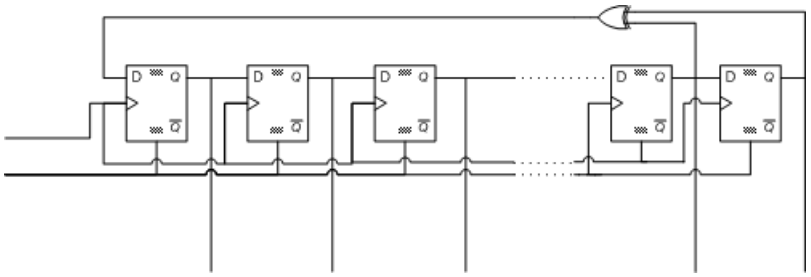
Σχήμα 5.5: Binary Tree Adder

Τα αρχικά σήματα εισόδου ανά δύο πηγαίνουν στους 512 αθροιστές 1 bit , σε κάθε ένα από αυτούς έχουμε έξοδο μια ποσότητα 2 bit η οποία είναι το άθροισμα των αντίστοιχων εισόδων. Στη συνέχεια η έξοδος αυτών των αθροιστών γίνεται είσοδος σε αθροιστές 2 bit , έπειτα σε 3 bit κ.ο.κ. Η έξοδος των 2 αυτών αθροιστών γίνονται

είσοδοι στον αφαιρετή ο οποίος κάνει την αφαίρεση $Makecount - Breakcount$ της οποίας το αποτέλεσμα είναι και το score της Novelty.

False Clause Selector

Στην μονάδα επιλογής μη ικανοποιήσιμου clause έχουμε είσοδο την πληροφορία για κάθε ένα clause μέσω ενός σήματος αν ικανοποιείται ή όχι. Η μονάδα αυτή αποτελείται από 2 στοιχεία. Την γεννήτρια ψευδοτυχαίων αριθμών η οποία υλοποιείται με την χρήση ενός shiftregister όπως φαίνεται και στο σχήμα 5.6 και το συνδυαστικό κύκλωμα το οποίο βοηθά την γεννήτρια τυχαίων αριθμών να δίνει πάντα μη ικανοποιήσιμο clause στην έξοδο. Ο λόγος που γίνεται αυτό είναι για να μπορεί μέσα σε ένα κύκλο να επιλέγεται το false clause . Στο σχήμα 5.7 φαίνεται αναλυτικά το κύκλωμα της μονάδας για την σχεδίαση με τα 1024 clauses.

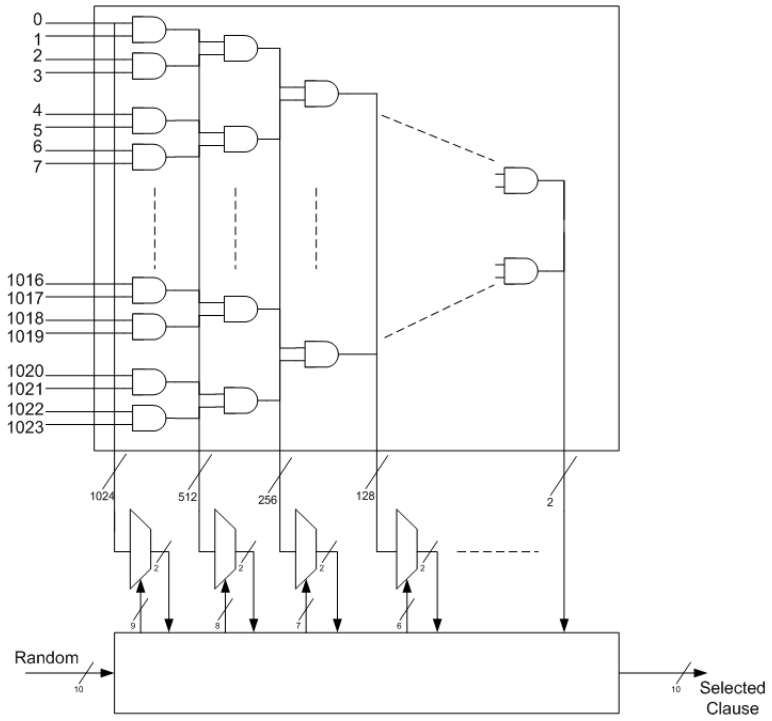


Σχήμα 5.6: Γεννήτρια Ψευδοτυχαίων Αριθμών

Βασική ιδέα του κυκλώματος είναι να εξετάζει ένα ένα τα bits που δίνει η γεννήτρια τυχαίων αριθμών σε συνδυασμό με τα σήματα που καθορίζουν την κατάσταση των clauses.Σε περίπτωση που στην ομάδα των clauses που αντιστοιχεί το εξεταζόμενο bit δεν υπάρχει μη ικανοποιήσιμο clause τότε το bit αυτό αντιστρέφεται. Έτσι αρχίζοντας από το MSB της τυχαίας τιμής και κάνοντας flip όπου χρειάζεται έχουμε μια ποσότητα η οποία αντιστοιχεί σε μη ικανοποιήσιμο clause.

5.2.5 5ο Στάδιο - Επιλογή Μεταβλητής

Το τελευταίο σημείο του αλγορίθμου. Το στάδιο αυτό χρειάζεται περισσότερο από ενα κύκλους για να ολοκληρωθεί η λειτουργία του. Στο σημείο αυτό καταλήγουν τα δεδομένα που αφορούν τις μεταβλητές και γίνεται σύγκριση τους. Στο σχήμα 5.8 παρουσιάζεται το διάγραμμα του κυκλώματος που επιλέγει την μεταβλητή. Τα βασικά στοιχεία του Variable selector ειναι οι καταχωρητές που κρατούν τα δεδομένα για την καλύτερη και δεύτερη καλύτερη μεταβλητή, η μνήμη που κρατά τα flips που έχει κάνει



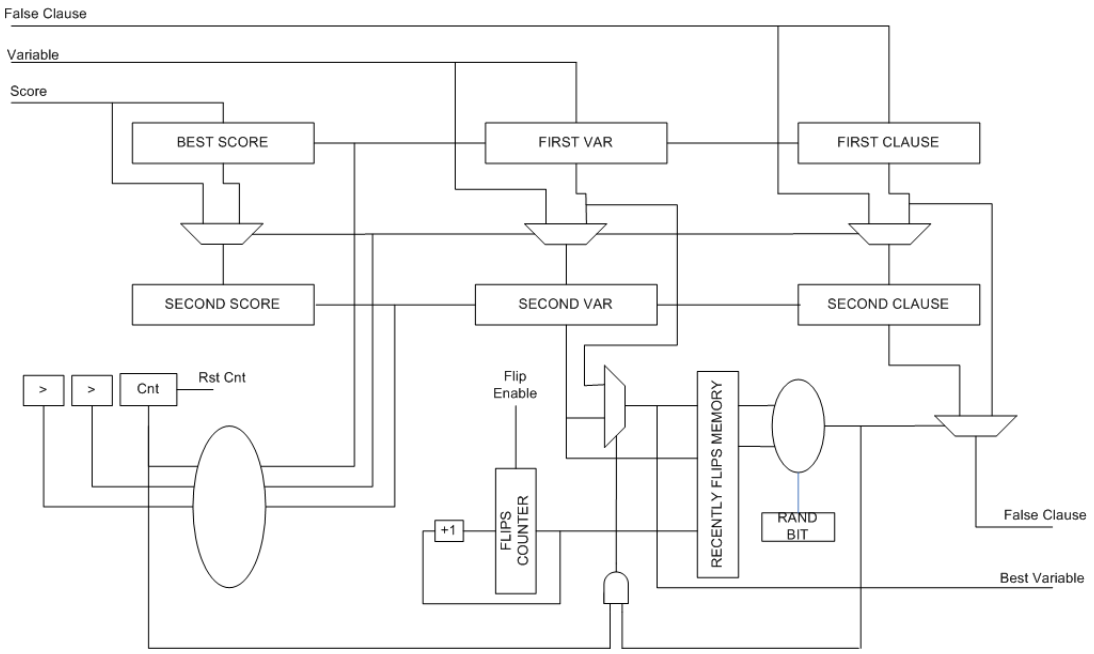
Σχήμα 5.7: Μονάδα Επιλογής Τυχαίου Clause

κάθε μεταβλητή, τον flip counter και τις μονάδες ελέγχου.

Ανάλυση Σχήματος 5.8

Υπάρχουν 2 επίπεδα καταχωρητών που αποτελούντε το κάθε ένα από 3ις καταχωρητές. Στο πρώτο επίπεδο έχουμε τους καταχωρητές για την καλύτερη μεταβλητή. Οι τρεις καταχωρητές κρατούν την τιμή του score , την τιμή της μεταβλητής και το επιλεγμένο μη ικανοποιήσιμο clause.Είσοδος των καταχωρητών είναι τα δεδομένα που έρχονται από το προηγούμενο στάδιο. Το σήμα εγγραφής του καταχωρητή προέρχεται από την μια μονάδα ελέγχου εγγραφής καταχωρητών που θα δούμε στην συνέχεια. Η έξοδοι τους οδηγούντε σε τρεις πολυπλέκτες που καθορίζουν τα δεδομένα προς εγγραφή στο δεύτερο επίπεδο καταχωρητών.

Το δεύτερο επίπεδο κρατά την πληροφορία για την δεύτερη καλύτερη μεταβλητή και αποτελείται από τους αντίστοιχους καταχωρητές με το πρώτο επίπεδο. Οι είσοδοι των καταχωρητών μπορεί να είναι είτε η πληροφορία όπως έρχεται στο στάδιο είτε τα περιεχόμενα του πρώτου επιπέδου καταχωρητών και καθορίζονται από ένα πολυπλέκτη ο οποίος ελέγχεται από την μονάδα ελέγχου εγγραφής καταχωρητών.



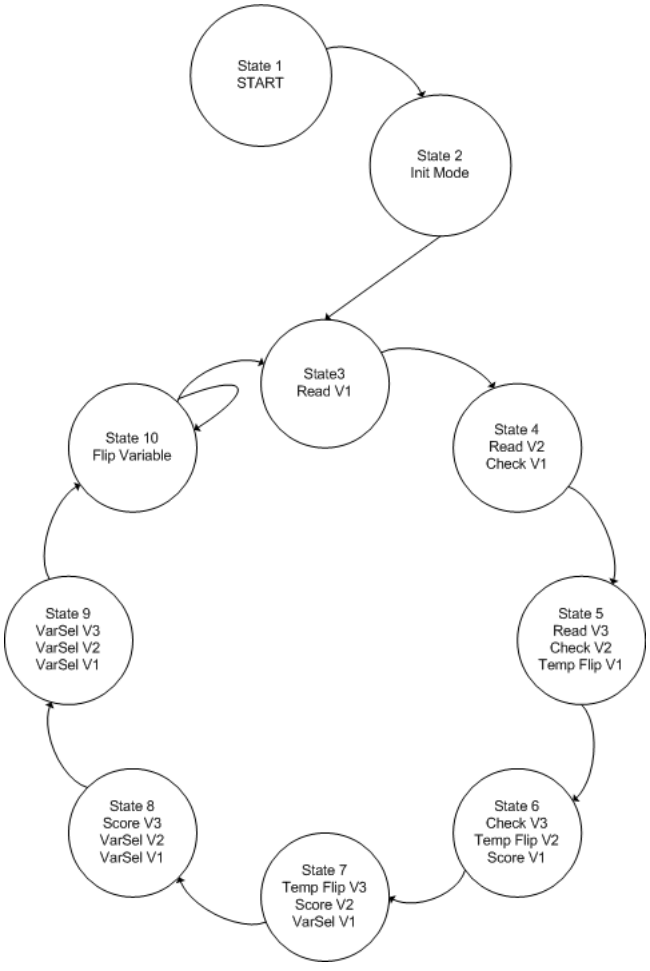
Σχήμα 5.8: Μονάδα Επιλογής Μεταβλητής

Η εγγραφή στους καταχωρητές και ο έλεγχος των πολυπλεκτών καθορίζεται από το score των μεταβλητών και είναι τα σήματα εξόδου της αντίστοιχης μονάδας ελέγχου. Το πρώτο επίπεδο καταχωρητών γράφεται όταν η εισερχόμενη μεταβλητή είναι καλύτερη από αυτή που υπάρχει ήδη στο επίπεδο. Παράλληλα γίνεται ενεργοποίηση εγγραφής στο δεύτερο επίπεδο ώστε να ενημερωθεί η δεύτερη στην κατάταξη μεταβλητή. Στην περίπτωση που η εισερχόμενη μεταβλητή είναι χειρότερη από την πρώτη αλλά καλύτερη από την δεύτερη τότε γίνεται ενεργοποίηση εγγραφής στο δεύτερο επίπεδο καταχωρητών και ως είσοδος στους καταχωρητές τα σήματα πό την είσοδο του Variable Selector. Όταν η μεταβλητή είναι χειρότερη και από τις δύο τότε δεν γίνεται ενεργοποίηση εγγραφής σε κανένα επίπεδο και η κατάσταση στους καταχωρητές μένει ως έχει.

Η τελική επιλογή μεταβλητής γίνεται σε συνδυασμό με την μνήμη που κρατά τα flips. Γίνεται σύγκριση στην τιμή που διαβάζεται από την μνήμη των flips των δύο μεταβλητών και αν η πρώτη έγινε τελευταία flip τότε επιλέγεται τυχαία μια από της δύο αλλιώς επιλέγεται η πρώτη. Η επιλογή καθορίζεται από τον πολυπλέκτη στις εξόδους των καταχωρητών που κρατούν τις τιμές για τα false clauses.

5.3 Μονάδα Ελέγχου

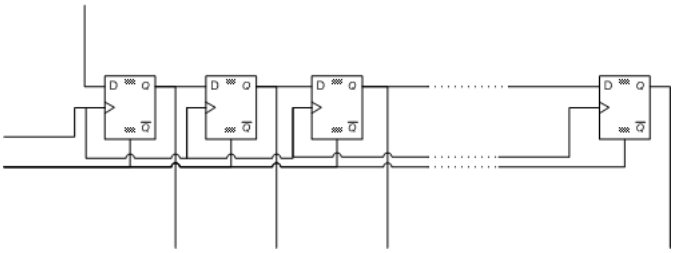
Η μονάδα ελέγχου της αρχιτεκτονικής καθορίζει τα σήματα ελέγχου όπως φαίνεται και από το σχήμα 5.1. Βασικό στοιχείο της μονάδας είναι μια μηχανή πεπερασμένων καταστάσεων (FSM) η οποία αποτελείται από 10 καταστάσεις, δύο για την αρχικοποίηση του συστήματος και 8 για την λειτουργία του αλγορίθμου όπως βλέπουμε και στο σχήμα 5.9. Οι πρώτες δύο καταστάσεις είναι υπεύθυνες για τα σήματα ελέγχου τα οποία αφορούν την αρχικοποίηση και οι υπόλοιπες για την λειτουργία του αλγορίθμου, μία κατάσταση για κάθε κύκλο.



Σχήμα 5.9: FSM για την μονάδα ελέγχου

5.3.1 Αρχικοποίηση Κυκλώματος

Παράλληλα με την μονάδα ελέγχου για την αρχικοποίηση λειτουργεί και ένας Shift Register, σχήμα 5.10, ο οποίος αρχίζει να λειτουργεί με σήμα από την μονάδα ελέγχου και παράγει ένα bit με τιμή 1 κάθε κύκλο ολισθιμένο προς τα αριστερά, αρχίζοντας από το LSB και πηγαίνοντας προς το MSB. Σκοπός του καταχωρητή αυτού είναι σειριακά να γίνει ενεργοποίηση των σημάτων εγγραφής στα clauses ώστε να γίνει εγγραφή των δεδομένων στα Variable Arrays και στα LVTs.



Σχήμα 5.10: Καταχωρητής ολίσθησης για αρχικοποίηση

Η σύνδεση του καταχωρητή αυτού με την FSM γίνεται με δύο σήματα. Το πρώτο σήμα δίνει την έναρξη λειτουργίας του και αφού γίνει η ολίσθηση του σήματος ο καταχωρητής μέσω του δεύτερου σήματος ενημερώνει την FSM. Όσο αφορά το υλικό το πρώτο σήμα, από την FSM, είναι η είσοδος του πρώτου flip flop και το δεύτερο, προς την FSM, η έξοδος του τελευταίου flip flop. Εκτός από τα δύο σήματα που συνδέονται με τον καταχωρητή ολίσθησης από την FSM έχουμε το σήμα που καθορίζει αν το σύστημα βρίσκεται σε κατάσταση αρχικοποίησης ή οχι έχοντας τιμή 1 και 0 αντίστοιχα.

Το σήμα αυτό καθορίζει την εγγραφή σε συνδυασμό με το σήμα του καταχωρητή ολίσθησης στα Variable Arrays τα οποία όπως προαναφαίρεται γράφονται μόνο στην αρχικοποίηση και στη συνέχεια μόνο διαβάζονται. Στα LVTs κατά την διάρκεια αρχικοποίησης εκτός από την εγγραφή μνήμης η οποία όπως και στα Variable Arrays καθορίζεται για κάθε clause από τον καταχωρητή ολίσθησης χρησιμοποιείται και το σήμα κατάστασης αρχικοποίησης το οποίο ελέγχει τον πολυπλέκτη που καθορίζει τα δεδομένα εγγραφής των μνημών.

Κατά την διάρκεια της αρχικοποίησης αυτό που γίνεται είναι η εισαγωγή των δεδομένων του προβλήματος. Τα δεδομένα που εισάγονται είναι η CNF , δηλαδή οι τρεις τιμές που αφορούν τις μεταβλητές $(3 \times \log_2(\#Variables)bit)$ μαζί με ένα bit για κάθε μεταβλητή το οποίο καθορίζει αν είναι στην θετική ή αρνητική της μορφή μέσα στο clause. Τα 24 bits συνδέονται με το Clause Table όπου γίνεται η εγγραφή της μνήμης και με τα 1024 Variable Arrays. Το σήμα ενεργοποίησης εγγραφής της μνήμης κατά

την διάρκεια της αρχικοποίησης γίνεται 1 και στην συνέχεια παραμένει 0. Τα άλλα τρία bits συνδέονται με τα LVTs τα οποία σε συνδυασμό με την τιμή της τυχαίας αρχικοποίησης των μεταβλητών γράφονται στην πρώτη θέση των μνημών που τα αποτελούν η οποία κρατά την πληροφορία για την τρέχων κατάσταση.

Μετά την διαδικασία αρχικοποίησης η οποία διαρκεί σε κύκλους όσο και ο αριθμός των clauses η FSM οδηγείται στην τρίτη κατάσταση με την οποία ξεκινά και η εκτέλεση του αλγορίθμου.

5.3.2 Λειτουργεία WalkSAT

Για την λειτουργία του βασικού αλγορίθμου έχουμε τις καταστάσεις 3-10 όπως φαίνονται και στο σχήμα 5.9 στις οποίες η FSM εισέρχεται μετά από την περάτωση της αρχικοποίησης. Αυτό που γίνεται εδώ είναι ανάλογα με το στάδιο του pipeline να ενεργοποιούνται και να θέτονται σωστά τα σήματα ελέγχου. Στη συνέχεια αναλύονται τα σήματα ελέγχου για την κανονική ροή του αλγορίθμου.

Clause Table

Current Variable Σήμα το οποίο καθορίζει πια από τις τρεις μεταβλητές θα επιλεγεί από τον πολυπλέκτη. Το σήμα αυτό είναι χρήσιμο στους πρώτους τρεις κύκλους ενώς flip και παίρνει διαδοχικά τις τιμές 00, 01 και 10.

Literals - LVTs

LVT Enable Σήμα το οποίο ενεργοποιεί την εγγραφή στης μνήμες των LVTs το οποίο παίρνει την τιμή ενεργοποίησης σε τέσσερις περιπτώσεις. Οι πρώτες τρεις περιπτώσεις είναι κατά την διάρκεια των προσωρινών flip των μεταβλητών και η τέταρτη φορά όταν θα γίνει το μόνιμο flip και θα ενημερωθεί η μόνιμη κατάσταση των literals.Οι κύκλοι στους οποίους γίνεται 1 είναι ο 4ος, 5ος, 6ος και 9ος.

Write Address Καθορίζει την διεύθυνση εγγραφής των LVTs και χρήσιμη τιμή έχει στους αντίστοιχους κύκλους όπου γίνεται ενεργοποίηση του σήματος εγγραφής. Οι πρώτες τρεις τιμές που παίρνει το σήμα αυτό αναφέρονται στην 2η, 3η και 4η θέση μνήμης αντίστοιχα και η τέταρτη τιμή που παίρνει είναι για την πρώτη θέση.

Read Address Σήμα για την διεύθυνση ανάγνωσης της μνήμης το οποίο είναι σταθερό και δείχνει στην πρώτη θέση για τις τρεις περιπτώσεις και για την τέταρτη καθορίζεται από την μεταβλητή που επιλέχθηκε.

Variable Selector

Reset Counter Όταν αρχίσουν να εισέρχονται οι μεταβλητές στο στάδιο αυτό θα χρειαστούν τρεις κύκλοι για την επιλογή της μεταβλητής προς αντιστροφή έτσι το σήμα αυτό αρχικοποιεί τον μετρητή ο οποίος θα παράξει τις τιμές 00,01,10 που θα καθορίσουν τους τρεις κύκλους και παράλληλα θα παράξουν κάποια σήματα ελέγχου για τους πολυπλέκτες στη μονάδα Variable Selector.

Flip Register Enable Μετά το πέρας των κύκλων για την επιλογή μεταβλητής ενεργοποιείται αυτό το σήμα το οποίο αυξάνει τον μετρητή των flips κατά 1 και παράλληλα γράφει την τιμή του στην αντίστοιχη διεύθυνση της μνήμης για τα flip κάθε μεταβλητής.

Κεφάλαιο 6

Αποτελέσματα

Στο κεφάλαιο αυτό θα μελετηθεί η αποτύπωση της αρχιτεκτονικής μας σε δύο οικογένειες FPGA της Xilinx, Virtex II και Virtex 5. Για την αποτύπωση των συστημάτων χρησιμοποιήθηκε τα εργαλείο της Xilinx, ISE 9.1 sp3 στο οποίο έγινε και προσομοίωση του συστήματος. Στις ενότητες που ακολουθούν θα παρουσιάσουμε τα αποτελέσματα τόσο σε πόρους όσο και σε ταχύτητα της αποτύπωσης της αρχιτεκτονικής μας για διάφορα μεγέθη και πως αλλάζουν τα δεδομένα αυξάνοντας το μέγεθος της σχεδίασης. Στη συνέχεια θα συγκρίνουμε τα αποτελέσματα μας με αποτελέσματα από άλλες αρχιτεκτονικές και λογισμικό.

6.1 Συμπεριφορά Hardware σε σχέση με λογισμικό

Σε αυτή την ενότητα θα μελετήσουμε την συμπεριφορά της αρχιτεκτονικής σε σχέση με το λογισμικό που περιγράψαμε στην ενότητα 3.3 όσο αφορά τα βήματα του αλγορίθμου προς την λύση ενός προβλήματος. Γενικά το σημείο που καθορίζει την ροή του αλγορίθμου είναι η επιλογή της μεταβλητής προς αντιστροφή. Πιο αναλυτικά είναι τα σημεία που υπολογίζουν το Score μαζί με το σημείο που επιλέγει το τυχαίο μη ικανοποιήσιμο Clause. Στην αρχιτεκτονική μας αυτό που κάναμε είναι να σχεδιάσουμε τις μονάδες αυτές να λειτουργούν παρόμοια με το λογισμικό και σε συνδυασμό με την μονάδα που κάνει της συγκρίσεις έχουμε την ίδια ροή στην διαδικασία εύρεσης λύσης.

Στον πίνακα 6.1 παρουσιάζουμε την σύγκριση λογισμικού και υλικού όπως ελέγχθεικε μέσα από simulations που έγιναν με την χρήση του Xilinx ISE 9.1.03. Στο σημείο αυτό να αναφέρουμε ότι κατά την διάρκεια προσομοίωσης του συστήματος με μεγάλα benchmarks δεν μπόρεσε να ολοκληρωθεί η προσομοίωση στο εργαλείο της Xilinx και αυτό μας οδήγησε στην επαλύθευση λειτουργίας του συστήματος με μικρά προβλήματα και στη συνέχια να κάνουμε εκτίμιση για τα flips και τον χρόνο εκτέλεσης

με χρήση του λογισμικού και της συχνότητας του συστήματος.

Benchmarks	Flips with Software	Flips with Hardware
uf20-010	21.12	19
uf20-016	36.4	32
uf20-020	36.29	40
uf20-023	60.32	52
uf20-029	62.88	60

Πίνακας 6.1: Επαλήθευση λειτουργίας Υλικού

6.2 Αποτύπωση Αρχιτεκτονικής σε Virtex II και Virtex 5

Στην ενότητα αυτή θα παρουσιάσουμε τα αποτελέσματα που αφορούν τους πόρους και τη συχνότητα λειτουργίας της αποτύπωσης για την οικογένεια Virtex II και Virtex 5 . Στους πίνακες που θα ακολουθήσουν θα παρουσιάσουμε τα αποτελέσματα κάθε μονάδας και ολόκληρης της αρχιτεκτονικής για όλα τα μεγέθη που μελετήσαμε και υλοποιήσαμε. Όσο αφορά τις Virtex II και Virtex 5 έχουμε τα μοντέλα Virtex XC2V6000 και XC5VLX110T αντίστοιχα.

6.2.1 Κατανάλωση πόρων από επί μέρους μονάδες

Clause Table

Στους πίνακες 6.2 και 6.3 παρουσιάζονται οι πόροι που χρειάζεται η μονάδα Clause Table για Virtex 2 και 5 αντίστοιχα.

	Available	128/32	256/64	512/128	1024/256
Slices	33792	5(0%)	6(0%)	7(0%)	8(0%)
4 input LUTs	67584	10(0%)	12(0%)	14(0%)	14(0%)
BRAMS	144	1(0%)	1(0%)	1(0%)	1(0%)

Πίνακας 6.2: Πόροι για Clause Table σε Virtex II

Logic Utilization	Available	128/32	256/64	512/128	1024/256
Slice Registers	69120	5 (0%)	6 (0%)	7 (0%)	8 (0%)
Slice LUTs	69120	5 (0%)	6 (0%)	7 (0%)	8 (0%)
Block RAM/FIFO	148	1 (0%)	1 (0%)	1 (0%)	1 (0%)

Πίνακας 6.3: Πόροι για Clause Table σε Virtex 5

Όσο αυξάνουμε τον αριθμό των Clauses και Variables παρατηρούμε την μονάδα αυτή να αυξάνει ελάχιστα τις απαιτήσεις της. Ο λόγος που συμβαίνει αυτό ειναι γιατί το κύριο στοιχείο της μονάδας είναι η μνήμη που κρατά την πληροφορία για τη CNF η οποία βρίσκετε στην Block RAM που χρησιμοποιείται. Η αύξηση που παρατηρείται στα Slices οφείλεται στην αύξηση του μεγέθους του πολυπλέκτη που χρησιμοποιείται για την επιλογή της μεταβλητής.

Clause Evaluator

Στους πίνακες 6.4 και 6.5 παρουσιάζονται οι πόροι που χρειάζεται η μονάδα Clause Evaluator για Virtex 2 και 5 αντίστοιχα.

	Available	128/32	256/64	512/128	1024/256
Slices	33792	2786 (8%)	6018 (17%)	12920 (38%)	27603 (81%)
Slice Flip Flops	67584	2817 (4%)	6401 (9%)	14337 (21%)	31745 (46%)
4 input LUTs	67584	3584 (5%)	7936 (11%)	17408 (25%)	34816 (51%)

Πίνακας 6.4: Πόροι για Clause Evaluator σε Virtex II

Logic Utilization	Available	128/32	256/64	512/128	1024/256
Slice Registers	69120	2817 (4%)	6401 (9%)	14337 (20%)	31745 (45%)
Slice LUTs	69120	2432 (3%)	5632 (8%)	11264 (16%)	22528 (32%)
Block RAM/FIFO	148	0 (0%)	0 (0%)	0 (0%)	0 (0%)

Πίνακας 6.5: Πόροι για Clause Evaluator σε Virtex 5

Η βασικότερη μονάδα της σχεδίασης μας και η πιο απαιτητική σε πόρους. Σε αυτό το σημείο και στην Virtex 2 όπως και στην Virtex 5 παρατηρούμε σταθερή αύξηση των πόρων της τάξης 2× όσο διπλασιάζουμε τα Clauses και Variables που υποστηρίζει

η αρχιτεκτονική μας. Ουσιαστικά γίνεται διπλασιασμός των LVTs και των Variable Arrays.

False Clause Selector

Στους πίνακες 6.6 και 6.7 παρουσιάζονται οι πόροι που χρειάζεται η μονάδα Clause Evaluator για Virtex 2 και 5 αντίστοιχα.

	Available	128/32	256/64	512/128	1024/256
Slices	33792	124(0%)	180 (0%)	328 (0%)	623 (1%)
Slice Flip Flops	67584	10(0%)	10 (0%)	10 (0%)	10 (0%)
4 input LUTs	67584	226(0%)	579 (0%)	1092 (1%)	2117 (1%)

Πίνακας 6.6: Πόροι για False Clause Selector σε Virtex II

Logic Utilization	Available	128/32	256/64	512/128	1024/256
Slice Registers	69120	10 (0%)	10 (0%)	10 (0%)	10 (0%)
Slice LUTs	69120	156 (0%)	490 (0%)	1067 (1%)	2092 (3%)
Block RAM/FIFO	148	0 (0%)	0 (0%)	0 (0%)	0 (0%)

Πίνακας 6.7: Πόροι για False Clause Selector σε Virtex 5

Όπως στην μονάδα του Clause Evaluator και εδώ παρατηρούμε μια γραμική αύξηση των πόρων της τάξης των 2×. Όσο διπλασιάζουμε τα Clauses χρειαζόμαστε άλλο ένα πολυπλέκτη ο οποίος θα είναι διπλάσιος από τον μεγαλύτερο της αμέσως μικρότερης σχεδίασης μαζί με ένα άλλο επίπεδο πυλών και για το λόγο αυτό παρατηρούμε λίγο περισσότερους από διπλάσιους πόρους.

Score Units

Στους πίνακες 6.8 και 6.9 παρουσιάζονται οι πόροι που χρειάζεται η μονάδα που είναι υπεύθυνη για τον υπολογισμό του Score για Virtex 2 και 5 αντίστοιχα.

Άλλη μια μονάδα που απαιτεί διπλάσιους πόρους κατά τον διπλασιασμό των Clauses. Ο λόγος που έχουμε διπλασιασμό εδώ είναι γιατί κάθε φορά που αυξάνουμε τα Clauses χρειαζόμαστε δύο αντίτυπα των μονάδων αυτών από την προηγούμενη σχεδίαση. Ένα για τα πρώτα μισά bits και ένα για τα άλλα μισά. Επίσης εκτός από τα δύο αντίτυπα χρειαζόμαστε ένα επιπλέον αθροιστή ο οποίος θα αθροίσει τα επιμέρους αθροίσματα. Στον πίνακα 6.10 παρατηρούμε τους πόρους που χρειάζεται η μονάδα αυτή με

	Available	128/32	256/64	512/128	1024/256
Slices	33792	325(0%)	654 (1%)	1320 (3%)	2647 (7%)
4 input LUTs	67584	587(0%)	1183 (1%)	2384 (3%)	4779 (7%)

Πίνακας 6.8: Πόροι για Score Units σε Virtex II

Logic Utilization	Available	128/32	256/64	512/128	1024/256
Slice Registers	69120	0 (0%)	0 (0%)	0 (0%)	0 (0%)
Slice LUTs	69120	521 (0%)	1051 (1%)	2120 (3%)	4251 (6%)
Block RAM/FIFO	148	0 (0%)	0 (0%)	0 (0%)	0 (0%)

Πίνακας 6.9: Πόροι για Score Units σε Virtex 5

1024 clauses και 256 variables στην περίπτωση που έχει καταχωρητές για μείωση του critical path και υποστηρίζει flip σε 9 κύκλους σε σύγκριση με την σχεδίαση χωρίς καταχωρητές.

Logic Utilization	Available	1024/256 (8)	1024/256 (9)
Slice Registers	69120	0 (0%)	72 (0%)
Slice LUTs	69120	4251 (6%)	4330 (6%)
Block RAM/FIFO	148	0 (0%)	0 (0%)

Πίνακας 6.10: Πόροι για Score Units σε Virtex 5

Variable Selector

Στους πίνακες 6.11 και 6.12 παρουσιάζονται οι πόροι που χρειάζεται η μονάδα που είναι υπεύθυνη για την επιλογή μεταβλητής, Variable Selector, για Virtex 2 και 5 αντίστοιχα.

Στην μονάδα αυτή παρατηρούμε αύξηση των πόρων μικρότερης της τάξης των 2×. Σε αυτή τη μονάδα αυτό που αλλάζει και προκαλεί την αύξηση των πόρων είναι οι καταχωρητές που κρατούν τις τιμές των Scores, False Clauses και Variable Numbers καθώς και οι συγκριτές. Κατά τον διπλασιασμό των Clauses οι καταχωρητές αυξάνονται κατά 1 bit.

	Available	128/32	256/64	512/128	1024/256
Slices	33792	120(0%)	174 (0%)	305 (0%)	527 (1%)
Slice Flip Flops	67584	91(0%)	97 (0%)	104 (0%)	115 (0%)
4 input LUTs	67584	224(0%)	330 (0%)	587 (0%)	1031 (1%)

Πίνακας 6.11: Πόροι για Variable Selector σε Virtex II

Logic Utilization	Available	128/32	256/64	512/128	1024/256
Slice Registers	69120	90 (0%)	97 (0%)	104 (0%)	111 (0%)
Slice LUTs	69120	156 (0%)	171 (0%)	276 (0%)	432 (0%)
Block RAM/FIFO	148	0 (0%)	0 (0%)	0 (0%)	0 (0%)

Πίνακας 6.12: Πόροι για Variable Selector σε Virtex 5

6.2.2 Κατανάλωση πόρων ολόκληρης της Αρχιτεκτονικής

Σε αυτή την παράγραφο θα δούμε τα χαρακτηριστικά ολόκληρης της αρχιτεκτονικής. Στον πίνακα 6.13 παρουσιάζονται όπως και στις μεμονωμένες μονάδες οι πόροι που χρειάζεται η αρχιτεκτονική για κάθε μέγεθος.

	Available	128/32	256/64	512/128	1024/256
Slices	33792	3428(10%)	7283 (21%)	15341 (45%)	32653 (97%)
Slice Flip Flops	67584	2925(4%)	6517 (9%)	14462 (21%)	31881 (47%)
4 input LUTs	67584	4524(6%)	9800 (14%)	21085 (31%)	42367 (62%)
BRAMS	144	1(0%)	1 (0%)	1 (0%)	1 (0%)

Πίνακας 6.13: Πόροι για ολόκληρη αρχιτεκτονική σε Virtex II

Στα σχήματα 6.1(α′) και 6.1(β′) παρουσιάζονται γραφικά οι πόροι που χρειάζεται η σχεδίαση για να αποτυπωθεί σε Virtex 2 και Virtex 5 αντίστοιχα. Από τα σχήματα παρατηρήται η γραμική αύξηση των πόρων κατά 2×.

6.2.3 Συχνότητα λειτουργίας συστήματος

Στον πίνακα 6.15 παρουσιάζουμε την μέγιστη συχνότητα που μπορεί να πετύχει κάθε σύστημα ανάλογα με τον αριθμό των Clauses και Variables που υποστηρίζει τα οποία παρουσιάζονται και γραφικά στο σχήμα 6.2.

Logic Utilization	Available	128/32	256/64	512/128	1024/256
Slice Registers	69120	2924 (4%)	6515 (9%)	14458 (20%)	31873 (46%)
Slice LUTs	69120	3348 (4%)	7423 (10%)	14831 (21%)	29659 (42%)
Block RAM/FIFO	148	1 (0%)	1 (0%)	1 (0%)	1 (0%)

Πίνακας 6.14: Πόροι για ολόκληρη αρχιτεκτονική σε Virtex 5

Device	128/32	256/64	512/128	1024/256
Virtex II	46.34	45.79	44.26	43.74
Virtex 5	105.74	90.53	84.84	75.43

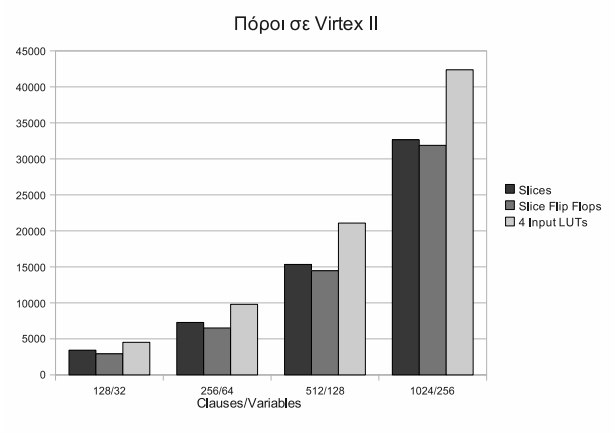
Πίνακας 6.15: Συχνότητα λειτουργίας συστήματος

Όσο αυξάνουμε το μέγεθος τις σχεδίασης (διπλασιασμός Clauses/Variables) παρατηρούμε μια μείωση τις συχνότητας. Αυτό οφείλετε στην μονάδα που υπολογίζει το Score η οποία καθορίζει και το critical path της σχεδίασης. Όσο διπλασιάζουμε την σχεδίαση έχουμε την προσθήκη ενός ακόμα αθροιστή στο δέντρο αθροιστών ο οποίος προσθέτει καθυστέρηση στο critical path. Όσο αφορά τις Virtex II και Virtex 5 παρατηρούμε την Virtex 5 να έχει πολύ καλύτερη συχνότητα αλλά να είναι πιο ευαίσθητη στο critcal path. Σε μεγαλύτερο μέγεθος μπορούμε να βελτιώσουμε το critical path χρησιμοποιώντας καταχωρητές στην μονάδα του score και ο υπολογισμός του score να γίνεται σε δύο κύκλους. Αυτό έχει ως αποτέλεσμα την μείωση του critical path από την μία και από την άλλη αυξάνει κατά ένα τους κύκλους που χρειάζεται ένα flip.

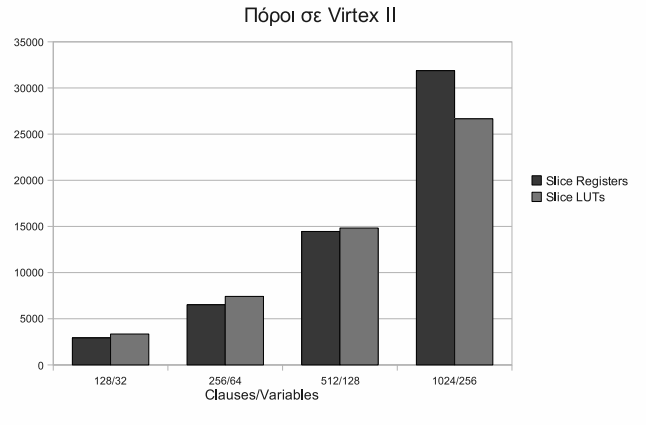
Στη συνέχεια παρουσιάζουμε τους πίνακες 6.16 και 6.17 στους οποίους φαίνονται ο αριθμός των flips που μπορεί το σύστημα να υποστηρίζει σύμφωνα με τους κύκλους που χρειάζεται για ένα flip και την συχνότητα λειτουργίας του για Virtex II και Virtex 5 αντίστοιχα.

Clauses/Vars	Cycles/Flip	Time for Flip	KFlips/s
1024/256	8	0.1829	5467.88
512/128	8	0.1807	5533
256/64	8	0.1747	5723.75
128/32	8	0.1727	5792

Πίνακας 6.16: KFlips/s για κάθε σύστημα σε Virtex II



(α′) Virtex II



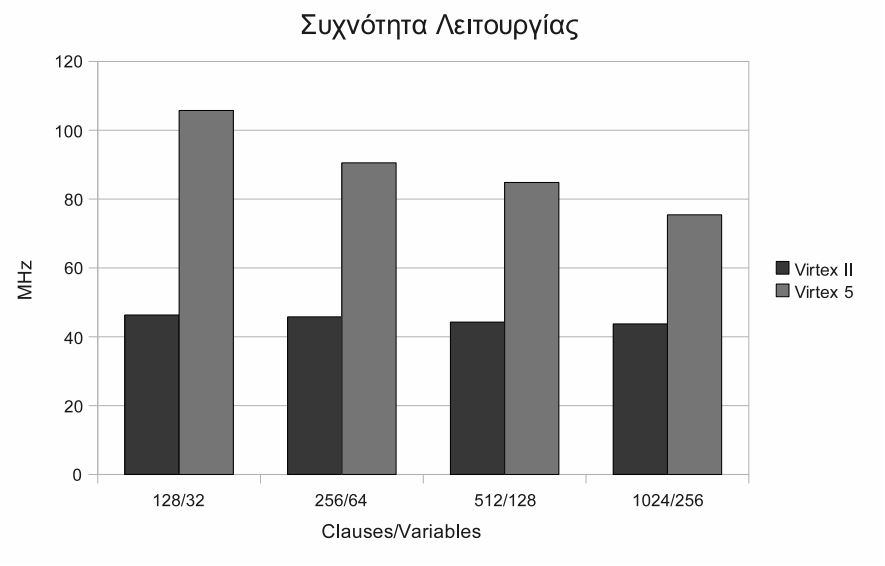
(β′) Virtex 5

Σχήμα 6.1: Πόροι που καταναλώνονται κατά την αποτύπωση

Στον πίνακα 6.16 στον οποίο φαίνονται ο αριθμός των Flips που μπορεί να κάνει το κάθε σύστημα ανά δευτερόλεπτο. Η τιμή αυτή υπολογίζεται από τον αριθμό των κύκλων που χρειάζονται για ένα flip και την συχνότητα λειτουργίας όπως παρουσιάζεται στον πίνακα 6.15.

6.3 Απόδοση σε διάφορα Benchmarks

Στην ενότητα αυτή θα δούμε την συμπεριφορά της σχεδίασης μας για διάφορα προβλήματα. Στον πίνακα 6.18 παρουσιάζονται τα προβλήματα μαζί με τα flips που χρειάζονται για να λυθούν όπως αυτά υπολογίστηκαν χρησιμοποιώντας την μοντελοποίηση του αλγορίθμου σε γλώσσα προγραμματισμού C που περιγράφεται στην ενότητα 3.3



Σχήμα 6.2: Συχνότητα λειτουργίας συστήματος

Clauses/Vars	Cycles/Flip	Time for Flip	KFlips/s
1024/256	8	0.1061	9428.25
512/128	8	0.0943	10605
256/64	8	0.0884	11315.75
128/32	8	0.0757	13216.88

Πίνακας 6.17: KFlips/s για κάθε σύστημα σε Virtex 5

για την Virtex II και Virtex 5 .

6.4 Σύγκριση με άλλες Αρχιτεκτονικές

Στο σημείο αυτό θα παρουσιάσουμε και θα συγκρίνουμε τα αποτελέσματα σε σχέση με την αρχιτεκτονική που περιγράφεται στην [KM05]. Μαζί με την σύγκριση σε επίπεδο υλικού θα γίνει και σύγκριση σε επίπεδο λογισμικού όπως αυτή παρουσιάστηκε στην δημοσίευση [KM05]. Στον πίνακα 6.19 παρουσιάζουμε τα αποτελέσματα για τα πέντε προβλήματα που αναφέραμε και στην ενότητα 3.4. Στους χρόνους από την αρχιτεκτονική που αναφέρεται [KM05] γίνεται αφαίρεση του χρόνου που χρειάζεται για είσοδο - έξοδο και ο οποίος σύμφωνα με την δημοσίευση είναι $5.5ms$

Benchmarks	Flips	T (ms) Virtex II	T (ms) Virtex 5	Speedup
uf225-087	1163.41	0.21	0.12	1.72
uf225-026	1387.09	0.25	0.15	1.72
uf225-028	10156.05	1.86	1.08	1.72
uf225-091	10887.16	1.99	1.15	1.72
uf225-039	1278917.95	233.9	135.65	1.72

Πίνακας 6.18: Χρόνος επίλυσης προβλημάτων για Virtex II και Virtex 5

Benchmarks	Hardware [KM05]			Our WalkSAT			SpeedUp
	Mean Flips	KFlips /sec	T(ms)	Mean Flips	KFlips /sec	T(ms)	
uf225-087	1936	331	0.35	1162	5467	0.22	1.3
uf225-026	2024	345	0.37	1387	5467	0.25	1.48
uf225-028	11723	1536	2.13	10156	5467	1.86	1.15
uf225-091	14962	1820	2.72	10887	5467	1.99	1.4
uf225-039	2690011	5439	489.08	1278917	5467	233.9	2.10

Πίνακας 6.19: Σύγκριση της σχεδίασης με [KM05] σε XC2V6000

Η καλύτερη απόδοση οφείλεται κατα κύριο λόγο στην ευριστική που χρησιμοποιούμε η οποία έχει περίπου 2× καλύτερη απόδοση ώς προς τον αριθμό των flips. Αυτό το αποτέλεσμα παρατηρείται στο δύσκολο πρόβλημα και φαίνεται καθαρά η χρήση της ευριστικής novelty πως επιρρεάζει θετικά την απόδοση αντικαθιστόντας την ευριστική skc. Η σύγκριση γίνεται στην αποτύπωση της σχεδίασης σε Virtex II στην οποία είναι αποτυπωμένη και η σχεδίαση της [KM05] με σκοπό να συγκρίνουμε τις ευριστικές και να μην επιρρεαστούν τα αποτελέσματα λόγο νεότερης τεχνολογίας.

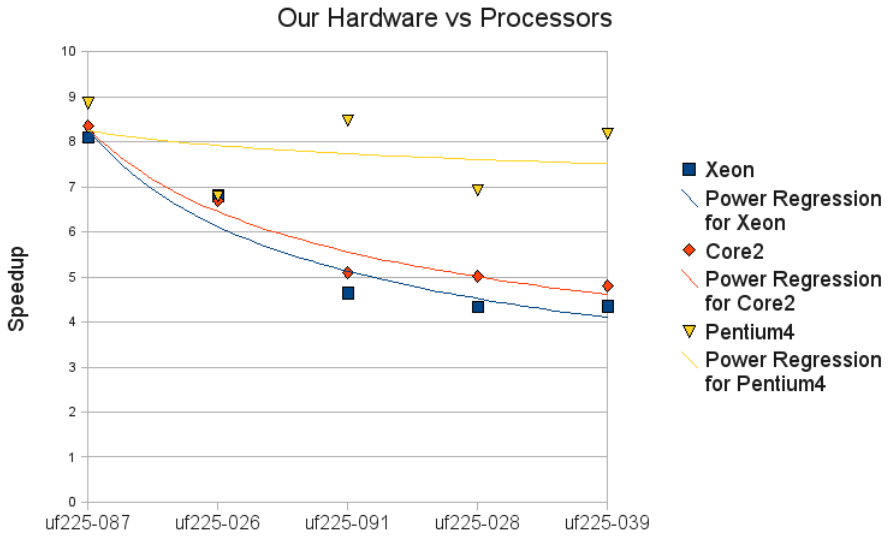
6.5 Σύγκριση με λογισμικό

Στην ενότητα αυτή θα συγκρίνουμε την απόδοση της αρχιτεκτονικής μας με το λογισμικό WalkSAT v46 [ws4] το οποίο χρησιμοποιά την ευριστική Novelty. Στον πίνακα 6.20 παρουσιάζουμε τα 5 προβλήματα που μελετάμε και τις αποδόσεις των δύο συστημάτων. Η εκτέλεση του λογισμικού έγινε σε υπολογιστή με επεξεργαστή Intel(R) Xeon(R) E5430 CPU 2.66GHz με 8 πυρήνες και 12 GB RAM.Στο σημείο αυτό να αναφέρουμε ότι η εκτέλεση του λογισμικού έγινε και σε υπολογιστή με επεξεργαστή Intel(R) Pentium(R) 4 CPU 2.53GHz και μνήμη 1 GB RAM και σε Intel(R) Core(TM)2 Duo CPU T8300 2.40GHz με μνήμη 4 GB RAM των οποίων τα αποτελέσματα φαίνονται

Benchmarks	Walksat v46 [ws4]			Our WalkSAT			SpeedUp
	Mean Flips	KFlips /sec	T(ms)	Mean Flips	KFlips /sec	T(ms)	
uf225-087	1306	1306	1	1163	9428	0.12	8.1
uf225-026	1558	1558	1	1387	9428	0.15	6.8
uf225-028	8242	1648	5	10156	9428	1.08	4.64
uf225-091	10342	2068	5	10887	9428	1.15	4.33
uf225-039	1384998	2347	589	1278917	9428	135.65	4.35

Πίνακας 6.20: Σύγκριση Λογισμικού - Υλικού

στο γράφημα 6.3.



Σχήμα 6.3: Απόδοση συστήματος σε σχέση με λογισμικό

Στο γράφημα 6.3 παρουσιάζεται το Speedup για τους τρεις επεξεργαστές. Παρατηρείται στα μικρά προβλήματα μεγάλο Speedup και αυτό οφείλεται στον τρόπο που το Software υπολογίζει το flips ανά δευτερόλεπτο. Υπολογίζει τον συνολικό χρόνο επεξεργασίας και σε συνδιασμό με τα flips που χρειαστίκανε υπολογίζει την τιμή των flips ανά δευτερόλεπτο. Στα δυσκολότερα προβλήματα παρατηρούμε το Speedup να σταθεροποιείτε και να είναι γύρο στο 4×.

Η απόδοση εδώ οφείλεται καθαρά στον παραλληλισμό που μπορούμε να πετύχουμε με την αποτύπωση της αρχιτεκτονικής σε αναδιατασσόμενη λογική εφόσον και το

λογισμικό καί η αρχιτεκτονική μας χρησιμοποιούν την ίδια ευριστική. Επίσης είναι χρήσιμο να αναφαίρουμε ότι και το λογισμικό σύμφωνα με τον πηγαίο κώδικα που το υλοποιεί κάνει βελτιστοποίηση στον πρόπο που ελέγχει την cnf εκμεταλλευόμενο μνήμη και κρατόντας περισσότερη πληροφορία για κάθε μεταβλητή όπως το score που μπορεί να πετύχει χωρίς τον έλεγχο ολόκληρης της cnf με αποτέλεσμα την καλύ επίδοση του. Στην αρχιτεκτονική μας συμαντικό παράγοντα στην απόδοση παίζει η χρήση pipeline η οποία επιτρέπει το flip να γίνεται σε 8 κύκλους αντί 18 που θα ήθελε χωρίς χρήση pipeline, πέντε για τον έλεγχο κάθε μεταβλητής και τρις για την επιλογή της μεταβλητής.

Κεφάλαιο 7

Συμπεράσματα και μελλοντικές επεκτάσεις

Στο κεφάλαιο αυτό θα παρουσιάσουμε τα συμπεράσματα που προέκυψαν από την εκπόνηση της παρούσας διπλωματικής και τις μελλοντικές επεκτάσεις που μπορεί να γίνουν για καλύτερη απόδοση και μεγαλύτερα προβλήματα.

7.1 Συμπεράσματα

7.1.1 Δυναμική Αναδιάταξη

Το πρώτο συμπέρασμα που καταλήξαμε προέρχεται από την πρώτη προσέγγιση του αλγορίθμου WalkSAT και την χρήση δυναμικής αναδιάταξης. Στην περίπτωση που μελετήσαμε παρατηρήθηκε η χρήση δυναμικής αναδιάταξης μη αποδοτική όπως αναφέρεται και στην ενότητα 4.1 λόγο του μεγάλου αριθμού επαναλήψεων για μεγάλα προβλήματα και δεδομένου του χρόνου που χρειάζεται να γίνει αναδιάταξη σε ένα μέρος της FPGA.

Επίσης μέσα από τα πειράματα παρατηρήθηκε εφόσον υπάρχει μια ευριστική καλύτερη από κάποια άλλη τότε δεν υπάρχει λόγος να χρησιμοποιηθεί η χειρότερη ευριστική, έτσι η προσέγγιση του προβλήματος χρησιμοποιώντας δυναμική αναδιάταξη στις ευριστικές δεν χρειάζεται.

7.1.2 Στατική Υλοποίηση

Το δεύτερο συμπέρασμα αφορά τις ευριστικές συναρτήσεις και την απόδοση τους. Παρατηρήσαμε την ευριστική novelty να έχει καλύτερη απόδοση από τις υπόλοιπες που

μελετήσαμε η οποία δεν είχε υλοποιηθεί σε υλικό μέχρι τώρα.

Στην συνέχεια μελετήσαμε τον αλγόριθμο Walksat και συγκεκριμένα το σημείο που γίνεται έλεγχος κατά πόσο ικανοποιήται η πρόταση cnf και πως αυτό μπορεί να παραλληλισθεί με σκοπό να εκμεταλλευθούμε την αναδιατασσόμενη λογική.

Τέλος έχοντας υπόψιν τα αποτελέσματα από την απόδοση του αλγορίθμου και συγκρίνοντας το με λογισμικό και άλλες αρχιτεκτονικές παρατηρήθηκε ότι η ευριστική novelty αν και χρειάζεται περισσότερη λογική και η επιλογή μεταβλητής είναι πιο σύνθετη στο υλικό παρουσιάζει καλύτερη απόδοση.

7.2 Μελλοντικές Επεκτάσεις

Σχεδίαση του Clause Evaluator ώστε να μπορεί να εκμεταλλευθεί περισσότερο της μνήμες BRAMs που προσφέρονται και μένουν ανεκμετάλλευτες στην παρούσα σχεδίαση. Αυτό θα βοηθήσει στην επέκταση της αρχιτεκτονικής και την υποστήριξη περισσότερων Clauses/Variables.

Σχεδίαση και υλοποίηση της ευριστικής R-Novelty η οποία φαίνεται να είναι καλύτερη από την novelty με σκοπό να αυξηθεί ακόμα περισσότερο η αποδοση του συστήματος.

Bibliography

[dim] *DIMACS Challenge–Satisfiability: Suggested Format.*

[DP] A. Dandalis and V.K. Prasanna. Run-Time Performance Optimization of an FPGA-Based Deduction Engine for SAT Solvers.

[DTYZ08] John D. Davis, Zhangxi Tan, Fang Yu, and Lintao Zhang. A Practical Reconfigurable Hardware Accelerator for Boolean Satisfiability Solvers. In *DAC-2008*, 2008.

[KM05] Kenji Kanazawa and Tsutomu Maruyama. An FPGA Solver for WSAT Algirithms. In *FPL05*, pages 83–88, 2005.

[KM06] Kenji Kanazawa and Tsutomu Maruyama. An FPGA Solver for large SAT Problems. In *FPL06*, pages 243–258, 2006.

[KM07] Kenji Kanazawa and Tsutomu Maruyama. An FPGA Solver for very large SAT Problems. In *FPL07*, pages 493–496, 2007.

[LP97] Harry Lewis and Christos H. Papadimitriou. *Elements of the Theory of Computation (2nd Edition).* Prentice Hall, 1997.

[LSW$^+$] P.H.W. Leong, C.W. Sham, W.C. Wong, H.Y. Wong, W.S. Yuen, and M.P. Leong. A Bitstream Reconfigurable FPGA Implementation of the WSAT Algorithm.

[MSK97] David McAllester, Bart Selman, and Henry Kautz. Evidence for Invariants in Local Search. In *AAAI-97*, pages 321–326, 1997.

[PM] M. Platzner and G. De Micheli. Acceleration of Satisfiability Algorithms by Reconfigurable Hardware.

[RN02] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 2002.

[sat] *http://www.cs.ubc.ca/ hoos/SATLIB/benchm.html.*

[SdBF04] Iouliia Skliarova and Antonio de Brito Ferrari. Reconfigurable Hardware SAT Solvers: A Survey of Systems. *IEEE Transactions on Computers*, 53(11):1449–1461, November 2004.

[SF] I. Skliarova and A.B. Ferrari. A Software/Reconfigurable Hardware SAT Solver.

[SYSN] T. Suyama, M. Yokoo, H. Sawada, and A. Nagoya. Solving Satisfiability Problems Using Reconfigurable Computing.

[ws4] *http://www.cs.rochester.edu/u/kautz/walksat/.*

[YSLL99] Wong Hiu Yung, Yuen Wing Seung, Kin Hong Lee, and Philip Heng Wai Leong. A Runtime Reconfigurable Implemantation of the GSAT Algorithm. In *FPL99*, pages 526–531, 1999.

[YWH03] Roland H.C. Yap, Stella Z.Q. Wang, , and Martin J. Henz. Hardware Implementations of Real-Time Reconfigurable WSAT Variants. In *FPL03*, pages 488–496, 2003.

[Zho] P. Zhong. Using Configurable Computing to Accelerate Boolean Satisfiability.