ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Τομέας Τεχνολογίας Πληροφορικής & Υπολογιστών
Εργαστήριο Υπολογιστικών Συστημάτων

Αυτοματοποίηση Συσχεδίασης Υλικού/Λογισμικού

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Χρήστος Γ. Παυλάτος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός & Τεχνολογίας Υπολογιστών
Πανεπιστήμιου Πατρών (2002)

Αθήνα, Νοέμβριος 2007

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΥΠΟΛΟΓΙΣΤΩΝ

ΠΡΑΚΤΙΚΟ ΕΞΕΤΑΣΗΣ ΔΙΔΑΚΤΟΡΙΚΗΣ ΔΙΑΤΡΙΒΗΣ

«ΑΥΤΟΜΑΤΟΠΟΙΗΣΗ ΣΥΣΧΕΔΙΑΣΗΣ ΥΛΙΚΟΥ/ΛΟΓΙΣΜΙΚΟΥ»

Χρήστος Παυλάτος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός
Πανεπιστήμιου Πατρών

Τριμελής Συμβουλευτική Επιτροπή:

Γεώργιος Παπακωνσταντίνου, Επιβλέπων, Καθηγητής ΕΜΠ
Παναγιώτης Τσανάκας, Καθηγητής ΕΜΠ
Νεκτάριος Κοζύρης, Επίκουρος Καθηγητής ΕΜΠ

Επταμελής Εξεταστική Επιτροπή

Γεώργιος Παπακωνσταντίνου
Καθηγητής ΕΜΠ

Παναγιώτης Τσανάκας
Καθηγητής ΕΜΠ

Νεκτάριος Κοζύρης
Επίκουρος Καθηγητής ΕΜΠ

Πεκμετζή Κεμάλ
Καθηγητής ΕΜΠ

Ανδρέας - Γεώργιος Σταφυλοπάτης
Καθηγητής ΕΜΠ

Θεόδωρος Ανδρόνικος
Λέκτορας
Ιόνιο Πανεπιστήμιο

Θέμις Παναγιωτόπουλος
Αναπληρωτής Καθηγητής
Πανεπιστημίου Πειραιώς

Αθήνα, 2 Νοεμβρίου 2007

.....

Χρήστος Γ. Παυλάτος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός & Τεχνολογίας Υπολογιστών
Πανεπιστήμιου Πατρών

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τον συγγραφέα και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν επίσημες τις θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

Αυτή η διδακτορική διατριβή χρηματοδοτήθηκε από το πρόγραμμα «ΠΕΝΕΔ 2003». Το «ΠΕΝΕΔ 2003» είναι μέρος του προγράμματος «ΑΝΤΑΓΩΝΙΣΤΙΚΟΤΗΤΑ» και είναι συγχρηματοδοτούμενο από την Ευρωπαϊκή Ένωση (80%) και Εθνικούς πόρους (20%).

Το έργο συγχρηματοδοτείται:

- *80% της Δημόσιας Δαπάνης από την Ευρωπαϊκή Ένωση – Ευρωπαϊκό Κοινωνικό Ταμείο*
- *20% της Δημόσιας Δαπάνης από το Ελληνικό Δημόσιο –Υπουργείο Ανάπτυξης – Γενική Γραμματεία Έρευνας και Τεχνολογίας*
- *Και από Ιδιωτικό Τομέα*

στο πλαίσιο του Μέτρου 8.3 του Ε.Π. Ανταγωνιστικότητα – Γ΄ Κοινοτικό Πλαίσιο Στήριξης

Περιεχόμενα*

*Οι αναφορές παρατίθενται σε ξεχωριστή ενότητα του κεφαλαίου στο οποίο αναφέρονται, με σκοπό την διευκόλυνση του αναγνώστη.

Αυτοματοποίηση Συσχεδίασης Υλικού/Λογισμικού

Χρήστος Γ. Παυλάτος

Περίληψη: Ένα Ενσωματωμένο Σύστημα (Ε.Σ.) (Embedded System) αποτελεί υπολογιστική μονάδα με αρχιτεκτονική και αρχές λειτουργίας παρόμοιες με αυτές των συμβατικών υπολογιστών, η οποία ωστόσο προσαρμόζεται στις ανάγκες και απαιτήσεις της εκάστοτε εφαρμογής. Βασικό δομικό στοιχείο ενός Ε.Σ., συνήθως, αποτελεί ένας μικροεπεξεργαστής, ο οποίος βρίσκεται συνδεδεμένος μέσω μιας ιεραρχίας διαύλων με στοιχεία προσωρινής και μόνιμης αποθήκευσης (μνήμες RAM, EPPROM, Flash), καθώς και με στοιχεία εξειδικευμένου υλικού τα οποία επικοινωνούν με τα βασικά δομικά στοιχεία και καλούνται να επιτελέσουν συγκεκριμένες εργασίες ανάλογα με τις απαιτήσεις της εκάστοτε εφαρμογής. Τα στοιχεία αυτά υλοποιούνται είτε σε μη επαναπρογραμματιζόμενο υλικό (VLSI, ASICs) είτε σε προγραμματιζόμενο υλικό (PLDs, FPGAs). Γίνεται συνεπώς κατανοητό ότι η σχεδίαση ενός Ε.Σ. προϋποθέτει, αφού καθοριστούν τα βασικά δομικά στοιχεία, να διαχωριστεί ο αλγόριθμος σε τμήματα που θα απεικονιστούν στο υλικό (FPGA, PLD κτλ) και σε τμήματα που θα απεικονιστούν στο λογισμικό (μικροεπεξεργαστής), καθώς και να διαμορφώσει κατάλληλο λειτουργικό σύστημα που θα διαχειρίζεται τον μικροεπεξεργαστή. Στην παρούσα Διδακτορική Διατριβή το χρησιμοποιούμενο μοντέλο για την αυτοματοποίηση της συσχεδίασης Υλικού/Λογισμικού είναι αυτό των Γραμματικών (Γραμματικών χωρίς συμφραζόμενα και Κατηγορηματικών Γραμματικών). Οι τελευταίες μπορούν να χρησιμοποιηθούν σε πλείστες όσες εφαρμογές Τεχνητής Νοημοσύνης και Συντακτικής Αναγνώρισης. Η προτεινόμενη μεθοδολογία βασίζεται στην απεικόνιση του συντακτικού τμήματος της γραμματικής σε υλικό (FPGA), του δε

σημασιολογικού σε έναν μικροεπεξεργαστή. Η πρώτη προσπάθεια στηρίχθηκε στον ακολουθιακό αλγόριθμο συντακτικής αναγνώρισης του R. Floyd. Στη συνέχεια, χρησιμοποιήθηκε ο πλέον αποδοτικός αλγόριθμος συντακτικής αναγνώρισης που υπάρχει, δηλαδή ο παράλληλος αλγόριθμος του J. Earley. Ο τελευταίος βελτιώθηκε και επεκτάθηκε για να καλύψει και τη σημασιολογία της γραμματικής. Οι παραπάνω αλγόριθμοι απεικονίστηκαν σε διάφορες αρχιτεκτονικές, που το FPGA επικοινωνεί με εξωτερικό μικροεπεξεργαστή, ή ο μικροεπεξεργαστής έχει επίσης απεικονιστεί στο FPGA όπου γίνεται και η συντακτική αναγνώριση διαχειριζόμενο πάντα από κατάλληλα τροποποιημένο λειτουργικό σύστημα.

Automated Hardware/Software Co-design

Christos J. Pavlatos

Abstract: An Embedded System (E.S.) constitutes a processing unit with architecture and principals similar with those of conventional computers, which is adapted to the needs and requirements of each application. Basic structural element of an E.S. is usually a microprocessor, which is connected, via busses, with elements of temporary and permanent storage (memories RAM, EPPROM, Flash), as well as with hardware modules which communicate with the basic structural elements and are used to carry out specific work, depending on the requirements of each application. These elements are implemented either in not reconfigurable hardware (VLSI, ASICs) or in reconfigurable hardware (PLDs, FPGAs). Consequently, the design of E.S. presupposes, after the specification of the basic structural elements, the division of the algorithm in parts that will be mapped in hardware (FPGA, PLD e.t.c.) and in parts that will be mapped in software (microprocessor). In addition appropriately modified Operating System should be used. In this PhD Thesis the underlying model for the automation of the hardware-software co-design is that of Grammars (Context-free grammars and Attribute Grammars). The last ones can be used in a considerable number of applications of Artificial Intelligence and Syntactic Pattern Recognition. The proposed methodology is based on the mapping of the syntactic part of grammar on hardware (FPGA), while the semantic part on a conventional microprocessor. The first effort was based on the sequential syntactic algorithm of R. Floyd. Afterwards the most efficient algorithm of syntactic recognition was used, that is the parallel algorithm of J. Earley. The last one was improved and extended in order to concurrently evaluate the semantics of the grammar. The aforementioned algorithms

were implemented in various architectures that either the FPGA cooperate with exterior microprocessor or the microprocessor has been mapped in the same FPGA where the syntactic recognition takes place, using in each case appropriately modified Operating System.

1. Πρόλογος

Στην παρούσα διατριβή επιχειρείται η συμβολή στην έρευνα της συσχεδίασης Ενσωματωμένων Συστημάτων και της βελτιστοποίησης αλγορίθμων Συντακτικής Αναγνώρισης Προτύπων, πραγματοποιώντας ένα ακόμα μικρό βήμα προς την αντιμετώπιση των προβλημάτων σχεδίασης Ενσωματωμένων Συστημάτων που εξειδικεύονται σε εφαρμογές που απαιτούν τη χρήση τόσο του διαδικαστικού, όσο και του δηλωτικού μοντέλου προγραμματισμού, στοχεύοντας στην βελτιστοποίηση της απόδοσής τους και στην απλοποίηση του τρόπου προγραμματισμού τους. Τέτοιες εφαρμογές απαντώνται κυρίως σε μεθοδολογίες Συντακτικής Αναγνώρισης Προτύπων και Τεχνητής Νοημοσύνης, σε πολλές εφαρμογές της σύγχρονης ζωής όπως στην ιατρική, στον αυτόματο έλεγχο, στη συλλογή και επεξεργασία πληροφοριών κ.α. Ο γνώμονας ήταν η πρακτική εφαρμογή όσων παρουσιάζονται και επιχειρήθηκε να δοθούν απλές καινοτόμες λύσεις σε ένα μικρό κομμάτι του προβλήματος.

Ο τομέας της σχεδίασης ενσωματωμένων συστημάτων, επιβάλλεται να ισορροπήσει μεταξύ πυκνότητας σχεδίασης, πολυπλοκότητας, χρόνου προς την αγορά και ικανοποίηση των απαιτήσεων της ζητούμενης προς υλοποίηση εφαρμογής. Ένα σύνολο παραγόντων που καθιστούν το πρόβλημα της σχεδίασης εξαιρετικά δύσκολο να αντιμετωπιστεί και λυθεί με τον καλύτερο δυνατό τρόπο. Έτσι, από τις αρχές του 1990 και μετά, ο τομέας αυτός άρχισε να αποτελεί ένα από τα πιο φλέγοντα θέματα έρευνας για την πανεπιστημιακή κοινότητα.

Σήμερα, στη σχετική επιστημονική βιβλιογραφία, απαντάται ένας μεγάλος όγκος ερευνητικού έργου που στοχεύει στην ανάλυση και επίλυση των προαναφερθέντων προβλημάτων σχεδίασης. Δυστυχώς, παρά τις μεγάλες προσπάθειες που γίνονται, η έρευνα στον τομέα αυτό εξακολουθεί να βρίσκεται στα αρχικά

της στάδια, κυρίως λόγω της πολυπλοκότητας του προβλήματος που αντιμετωπίζεται. Ευελπιστούμε, με τη συμβολή μας αυτή, να προκύψει ένα ακόμα βήμα προς την αποτελεσματική αντιμετώπιση των προβλημάτων σχεδίασης.

Θα αποτελούσε σοβαρή παράληψη να μην αναφερθεί η πολύτιμη συμβολή πολλών ανθρώπων που βοήθησαν, ο καθένας με τον τρόπο του, στην ολοκλήρωση αυτής της διατριβής. Πρωτίστως, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Γεώργιο Παπακωνσταντίνου για την πολύτιμη βοήθεια του στον ερευνητικό τομέα, αλλά κυρίως για την υπευθυνότητα και ηθική του, που αποτελούν για εμένα πρότυπο κοινωνικής και επαγγελματικής συμπεριφοράς. Η εμπιστοσύνη που επέδειξε στο πρόσωπό μου, ήταν η σημαντικότερη βοήθεια στις δυσκολίες και απογοητεύσεις που είχα κατά την διάρκεια εκπόνησης της διατριβής μου. Επίσης, θα ήθελα να ευχαριστήσω και τους κ. Τσανάκα και κ. Κοζύρη οι οποίοι συνέβαλλαν στην δημιουργία ενός επιστημονικά γόνιμου και ευχάριστου περιβάλλοντος. Επιπρόσθετα, θα ήθελα να ευχαριστήσω τους ανθρώπους που συνεργάστηκα στο εργαστήριο, αρχίζοντας από τον Γιάννη Παναγόπουλο που με βοήθησε σημαντικά στα πρώτα μου ερευνητικά βήματα, τον Αλέξανδρο Δημόπουλο που αποδεικνύεται καθημερινά πολύτιμος συνεργάτης, τον Γιάννη Ρυακιωτάκη και τον Μαρίνο Σαμψών για τις πολλές φορές που γελάσαμε παρέα κάνοντας διάλειμμα από τη δουλειά, τον Ανδρέα Κουλούρη και τον Θεόδωρο Ανδρόνικο για τη βοήθεια που μου προσέφεραν, τον Βαγγέλη Κούκη που ασχολείται καθημερινά με την επίβλεψη του δικτύου αλλά και όλα τα παιδιά του εργαστηρίου. Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου για την στήριξη που μου προσέφερε και εκείνη που ξέρει να με ηρεμεί και έχει ένα μοναδικό τρόπο να με εμψυχώνει και να με στηρίζει.

2. Εισαγωγή

Η παρούσα Διδακτορική Διατριβή παρουσιάζει μεθόδους αυτοματοποίησης της συσχεδίασης Υλικού/Λογισμικού, που αποσκοπούν στην επιτάχυνση της διαδικασίας σχεδιασμού Ενσωματωμένων Συστημάτων (Ε.Σ) καθώς και στην βελτιστοποίηση της απόδοσης τους. Στην ενότητα αυτή, παρουσιάζετε το γνωστικό αντικείμενο της Διδακτορικής Διατριβής. Πιο συγκεκριμένα παρουσιάζονται η βασική δομή των Ε.Σ και οι γενικότερες αρχές της συσχεδίασης Υλικού/Λογισμικού. Τέλος, παρατίθεται μια σύντομη περιγραφή των επιμέρους ενοτήτων που απαρτίζουν την Διδακτορική Διατριβή.

2.1 Ενσωματωμένα Συστήματα

Ένα «Ενσωματωμένο Σύστημα» (Ε.Σ.) (Embedded System) [2.1] αποτελεί υπολογιστική μονάδα με αρχιτεκτονική και αρχές λειτουργίας παρόμοιες με αυτές των συμβατικών υπολογιστών, η οποία ωστόσο προσαρμόζεται στις ανάγκες και απαιτήσεις της εκάστοτε εφαρμογής. Έτσι, και στην περίπτωση των Ε.Σ., βασικό δομικό στοιχείο αποτελεί ένας μικροεπεξεργαστής, ο οποίος βρίσκεται συνδεδεμένος μέσω μιας ιεραρχίας διαύλων με στοιχεία προσωρινής και μόνιμης αποθήκευσης (μνήμες RAM, EPPROM, Flash). Παράλληλα, στα Ε.Σ. απαντώνται και στοιχεία εξειδικευμένου υλικού τα οποία επικοινωνούν με τα βασικά δομικά στοιχεία και καλούνται να επιτελέσουν συγκεκριμένες εργασίες ανάλογα με τις απαιτήσεις της εκάστοτε εφαρμογής σε απόδοση, κατανάλωση ισχύος, λειτουργίες Ε/Ε κ.α. Τα στοιχεία αυτά υλοποιούνται είτε σε μη επαναπρογραμματιζόμενο υλικό (VLSI, ASICs) είτε σε προγραμματιζόμενο υλικό (PLDs, FPGAs) και δια-

συνδέονται μέσω μιας ιεραρχίας (πιθανώς πολλών επιπέδων) διαύλων με τον μικροεπεξεργαστή και τη μνήμη. Τα βασικά λειτουργικά στοιχεία και οι επιλογές αρχιτεκτονικής και υλοποίησης, οι οποίες και επηρεάζουν τα τελικά χαρακτηριστικά του Ε.Σ. ως προς την απόδοση, κόστος, επιφάνεια, κατανάλωση ενέργειας κ.α., είναι αρκετά και οι συνδυασμοί αυτών οδηγούν σε έναν μεγάλο χώρο σχεδίασης (design space). Επιγραμματικά μπορούν να αναφερθούν το επίπεδο αρχιτεκτονικής (η τεχνολογία του επεξεργαστή και των υπολοίπων δομικών στοιχείων του Ε.Σ., το επίπεδο ιεραρχίας των διαύλων και η ιεραρχίας της μνήμης), το επίπεδο παραμετρικών χαρακτηριστικών (η συχνότητα λειτουργίας στοιχείων και διαύλων, το μέγεθος των διαύλων, μεγέθη μνήμης και καταχωρητών) και το επίπεδο συμπεριφορικής λειτουργίας (το πρωτόκολλο επικοινωνίας των διαύλων, ο μηχανισμός καθορισμού προτεραιοτήτων, οι μέθοδοι επικοινωνίας μεταξύ των δομικών στοιχείων κα). Το μεγάλο μέγεθος του χώρου σχεδίασης που προκύπτει από τις διαφορετικές πιθανές επιλογές των προαναφερθέντων χαρακτηριστικών, καθιστά επιτακτική την ανάγκη χρήσης συγκεκριμένων μεθοδολογιών κατά τη σχεδίαση Ε.Σ, με στόχο τον εντοπισμό της βέλτιστης υλοποίησης βάσει της ζητούμενης εφαρμογής και των καθορισμένων προδιαγραφών. Πολύ περισσότερο, η δυνατότητα επιλογής υλοποίησης συγκεκριμένων λειτουργιών της αρχικής εφαρμογής σε Λογισμικό ή Υλικό συντελεί ακόμα περισσότερο στην αύξηση του χώρου σχεδίασης. Ως αποτέλεσμα, δεδομένης της ανάγκης για όσο το δυνατό μεγαλύτερη εξερεύνηση του χώρου σχεδίασης (design space exploration) στο μικρότερο δυνατό χρόνο (χρόνο προς την αγορά), οι μεθοδολογίες σχεδίασης και επιλογές υλοποίησης αναδεικνύονται στους βασικότερους παράγοντες που καθορίζουν το βαθμό επιτυχίας του τελικού προϊόντος.

Συμπερασματικά, κατά τη σχεδίαση των Ε.Σ., προκύπτει η ανάγκη για «Συσχεδίαση Υλικού/Λογισμικού» (Hardware/Software Codesign) [2.2] [2.3] [2.4], όπου βάσει των αρχικών προδιαγραφών και απαιτήσεων, το Ε.Σ. υλοποιείται επιλέγοντας λειτουργίες που θα πραγματοποιούνται από τον μικροεπεξεργαστή(ες) (Λογισμικό) και λειτουργίες που θα πραγματοποιούνται από τις εξειδικευμένες υπολογιστικές μονάδες (Υλικό). Επίσης, απαιτούνται μεθοδολογίες σχεδίασης, οι οποίες επιτρέπουν εξερεύνηση του χώρου σχεδίασης σε υψηλό επίπεδο αφαίρε-

σης, παρουσιάζοντας προσεγγίσεις των λειτουργικών χαρακτηριστικών της τελικής υλοποίησης και επιτρέποντας το σταδιακό περιορισμό των σχεδιαστικών επιλογών. Οι μεθοδολογίες αυτές, επιπρόσθετα, οδηγούν σε όλο και χαμηλότερα επίπεδα αφαίρεσης (design refinement) μέχρι την περιγραφή (σε συνθέσιμο κώδικα γλώσσας υλικού) του τελικού συστήματος. Έτσι, αφενός καθίσταται δυνατή η εξερεύνηση του χώρου σχεδίασης σε υψηλά επίπεδα αφαίρεσης σε αποδεκτό χρόνο και αφετέρου μειώνεται ο χρόνος υλοποίησης μέσω αυτοματοποιημένων διαδικασιών σύνθεσης υλικού. Ως αποτέλεσμα, προκύπτουν Ε.Σ. τα οποία καλύπτουν τις προδιαγραφές και ανάγκες της ζητούμενης εφαρμογής και παράλληλα δεν παραβιάζεται το χρονικό όριο μέχρι την αγορά (time to market design limit) που επιβάλλει σήμερα ο ανταγωνισμός της βιομηχανίας σχεδίασης Ε.Σ.

2.2 Συσχεδίαση Υλικού/Λογισμικού

Στην επιστημονική βιβλιογραφία και βιομηχανική πρακτική, απαντάται σήμερα μια πληθώρα μεθοδολογιών σχεδίασης Ε.Σ. η οποία συνοδεύεται από ένα μεγάλο αριθμό συμπληρωματικών εργαλείων, τα οποία στοχεύουν στην επίλυση του προβλήματος σχεδίασης Ε.Σ. Οι μεθοδολογίες αυτές ακολουθούν στενά την εξέλιξη της τεχνολογίας στην πολυπλοκότητα και πυκνότητα των δομικών στοιχείων της τελικής υλοποίησης η οποία μάλιστα εξελίσσεται με τάχιστους ρυθμούς. Έτσι, κατά τα μέσα της δεκαετίας του 90, η πυκνότητα και πολυπλοκότητα των ψηφιακών συστημάτων οδήγησε την επιστημονική κοινότητα στην αναζήτηση μεθόδων «Σύνθεσης Υλικού από Υψηλό επίπεδο» (High Level Synthesis) [2.5]. Η προσέγγιση αυτή, ουσιαστικά βασίζεται στην υλοποίηση μόνο σε Υλικό ολόκληρου του επιθυμητού συστήματος, βάσει αρχικών προδιαγραφών που δίνονται σε υψηλό επίπεδο αφαίρεσης. Αρχικά, η ζητούμενη εφαρμογή εκφράζεται με τη χρήση γλωσσών προγραμματισμού υψηλού επιπέδου, όπως είναι η C ή η SystemC. Με τον τρόπο αυτό επιβεβαιώνεται, αφενός η ορθή λειτουργία της ζητούμενης εφαρμογής (verification) και αφετέρου είναι δυνατή η αξιολόγηση της λειτουργίας της σε λογισμικό με τη χρήση κατάλληλου μεταγλωττιστή που συνοδεύει την επιλογή του επιθυμητού επεξεργαστή. Κατόπιν, η μεθοδολογία ε-

φαρμόζει τεχνικές οι οποίες επιτελούν τα τρία βασικότερα στάδια της σύνθεση υλικού τα οποία ονομάζονται: Χρονοδρομολόγηση (Scheduling), Επιλογή υπολογιστικών στοιχείων (Allocation) και Ανάθεση (Binding). Κατά το πρώτο στάδιο (Scheduling), χρονοδρομολογούνται όλοι οι υπολογισμοί της συμπεριφορικής περιγραφής βάσει των ζητούμενων προδιαγραφών σε απόδοση, κόστος, κατανάλωση ενέργειας κ.α. Στο επόμενο στάδιο (Allocation), επιλέγονται τα δομικά στοιχεία υλικού και ο αριθμός τους, τα οποία καλούνται να εκτελέσουν τους υπολογισμούς της ζητούμενης εφαρμογής. Στο τρίτο στάδιο (Binding), σε καθένα από τα δομικά στοιχεία τα οποία επιλέχθηκαν, ανατίθεται και η επιτέλεση συγκεκριμένων υπολογισμών. Ο τρόπος επιτέλεσης των παραπάνω σταδίων και οι τεχνικές βελτιστοποίησης που ακολουθεί η εκάστοτε μεθοδολογία είναι αυτά που καθορίζουν την ποιότητα του τελικού συστήματος. Οι τελικές σχεδιαστικές επιλογές εκφράζονται με τη χρήση μιας γλώσσας σύνθεσης υλικού, σε επίπεδο καταχωρητών (όπως η Verilog [2.6]) και μπορούν να τροφοδοτηθούν σε αυτοματοποιημένα εμπορικά εργαλεία κατασκευής του τελικού συστήματος.

Καθώς η υπολογιστική πολυπλοκότητα και η πυκνότητα των ψηφιακών συστημάτων αυξάνονταν με εκθετικούς ρυθμούς, τα ψηφιακά συστήματα έφτασαν σε τέτοιο βαθμό ολοκλήρωσης, που τεχνικές σύνθεσης υλικού από υψηλό επίπεδο αδυνατούσαν να καλύψουν τις ζητούμενες ανάγκες υλοποίησης και του περιορισμένου χρόνου προς την αγορά. Δύο βασικοί παράγοντες συντέλεσαν σε αυτό: i) ο βαθμός ολοκλήρωσης καθιστούσε εξαιρετικά χρονοβόρα τη σχεδίαση ψηφιακών συστημάτων από την αρχή (design from scratch) με αποτέλεσμα το τελικό προϊόν να παρουσιάζει μη αποδεκτή χρονική καθυστέρηση μέχρι την υλοποίησή του, ii) η πυκνότητα των ενσωματωμένων συστημάτων, καθιστούσε την καθυστέρηση των δομικών στοιχείων σε επίπεδο πυλών, συγκρίσιμη με την καθυστέρηση που παρατηρείται στη μεταφορά των ψηφιακών σημάτων πάνω στους διαύλους με αποτέλεσμα μέθοδοι χρονοδρομολόγησης να οδηγούν σε εσφαλμένα συμπεράσματα ως προς την απόδοση του τελικού συστήματος.

Για τους λόγους αυτούς, οι μεθοδολογίες σχεδίασης υλικού από υψηλό επίπεδο αντικαταστάθηκαν από μεθοδολογίες «Σχεδίασης Συστημάτων από Υψηλό Επίπεδο» (System Level Synthesis) [2.7]. Ειδοποιός διαφορά, αποτελεί το γεγονός ότι

στις νέες μεθοδολογίες, τα βασικά δομικά στοιχεία αποτελούσαν τώρα πολυπλοκότερες δομικές μονάδες (επεξεργαστές, μνήμες, εξειδικευμένο υλικό) οι οποίες συνδυάζονταν για την υλοποίηση του τελικού συστήματος (Ενσωματωμένο Σύστημα). Με τον τρόπο αυτό, περιορίζονταν το πρόβλημα σύνθεσης υλικού σε επιμέρους κομμάτια της τελικής υλοποίησης καθιστώντας εφικτή την εφαρμογή μεθόδων «Σχεδίασης από Υψηλό Επίπεδο» σε αποδεκτά χρονικά πλαίσια, ενώ παράλληλα, με την μετάβαση σε ένα υψηλότερο επίπεδο αφαίρεσης, επιτυγχάνονταν μεγαλύτερη ταχύτητα σχεδίασης με αποδεκτό βαθμό απώλειας στην τελική απόδοση του συστήματος. Τα βασικά στάδια της «Σχεδίαση Συστημάτων από Υψηλό Επίπεδο» είναι τα εξής: i) οι προδιαγραφές της επιθυμητής εφαρμογής δίνονται σε μια γλώσσα προγραμματισμού υψηλού επιπέδου, ii) επιλέγονται κομμάτια της εφαρμογής που θα υλοποιηθούν σε υλικό και κομμάτια που θα υλοποιηθούν σε λογισμικό και επιλέγονται τα δομικά στοιχεία του τελικού συστήματος σε υψηλό επίπεδο. Η επιλογή αυτή, αποτελεί σχεδιαστικό στιγμιότυπο του χώρου σχεδίασης και αξιολογείται ως προς την απόδοσή της. Σε αυτό το στάδιο δοκιμάζονται επαναληπτικά διάφορα στιγμιότυπα έως ότου εντοπιστεί αυτό που ικανοποιεί τις αρχικές προδιαγραφές. Η διαδικασία αυτή αποτελεί το στάδιο της εξερεύνησης του χώρου σχεδίασης (design space exploration) και η έκταση εφαρμογής της εξαρτάται άμεσα από τα χρονικά πλαίσια που επιβάλλει ο ζητούμενος χρόνος προς την αγορά και ο χρόνος που η εκάστοτε μεθοδολογία απαιτεί για την ανάλυση του κάθε επιλεγμένου στιγμιότυπου από τον χώρο σχεδίασης. Όταν το επιθυμητό σύστημα έχει εντοπιστεί, μεθοδολογίες σύνθεσης υλικού αναλαμβάνουν την υλοποίηση των μερών που επιλέχθηκαν να απεικονιστούν σε υλικό ενώ το λογισμικό εκφράζεται σε μια γλώσσα παρόμοια της C ώστε να προγραμματιστεί στους επεξεργαστές.

Καθώς η πυκνότητα των βασικών δομικών υλικών αυξήθηκε ακόμα περισσότερο, μεθοδολογίες «Σύνθεσης Συστημάτων από Υψηλό Επίπεδο» άρχισαν να αδυνατούν και αυτές με τη σειρά τους να επιτύχουν επιθυμητές υλοποιήσεις σε αποδεκτά χρονικά περιθώρια. Βασική αιτία αποτελούσε η εξαιρετικά χρονοβόρα διαδικασία υλοποίησης της διεπαφής (interface) μεταξύ των επιλεγμένων δομικών μονάδων και η ανάγκη επανασχεδίασης δομικών μονάδων σε οποιαδήποτε

νέα ανάγκη υλοποίησης συστήματος (μη επαρκής επαναχρησιμοποίηση υλικού). Έτσι κατά τις αρχές του 2000, εμφανίστηκαν νέες μεθοδολογίες σχεδίασης, γνωστές με το όνομα «Σχεδίαση συστημάτων βάσει πλατφόρμας» (Platform Based Design) [2.8]. Η βασικότερη διαφορά τους από τις προηγούμενες προσεγγίσεις αποτελεί το γεγονός ότι προεπιλεγμένες πλατφόρμες (δηλαδή παραμετρικές υλοποιήσεις ενσωματωμένων συστημάτων), επιλέγονται ανάλογα με τα γενικά χαρακτηριστικά τους και η διαδικασία σχεδίασης αποσκοπεί να απεικονίσει την ζητούμενη εφαρμογή στις υπολογιστικές τους μονάδες. Τα βασικότερα στάδια των μεθοδολογιών αυτών είναι τα εξής: i) αρχικά, μαζί με τη συμπεριφορική περιγραφή της ζητούμενης εφαρμογής, επιλέγεται και συγκεκριμένη πλατφόρμα στην οποία πρόκειται να υλοποιηθεί. Η επιλογή της πλατφόρμας γίνεται βάσει γενικών χαρακτηριστικών και απαιτήσεων της ζητούμενης υλοποίησης, ii) κατόπιν, επιλέγονται κομμάτια της αρχικής περιγραφής που θα απεικονιστούν σε υλικό της πλατφόρμας και κομμάτια που θα προγραμματιστούν στους επεξεργαστές. Κάθε μία από τις επιλογές απεικόνισης αποτελεί στιγμιότυπο του χώρου σχεδίασης και αξιολογείται βάσει των αρχικών προδιαγραφών (εξερεύνηση του χώρου σχεδίασης). Μόλις επιλεγεί το επιθυμητό στιγμιότυπο, κομμάτια της εφαρμογής, είτε απεικονίζονται στο ήδη υπάρχον υλικό, είτε σχεδιάζονται από την αρχή, ενώ κομμάτια προς υλοποίηση σε λογισμικό προγραμματίζουν τους μικροεπεξεργαστές της πλατφόρμας. Δεδομένου του κοινού και προεπιλεγμένου τρόπου επικοινωνίας μεταξύ των δομικών μονάδων του συστήματος από την πλατφόρμα και της δυνατότητας επαναχρησιμοποίησης δομικών στοιχείων από προηγούμενες υλοποιήσεις στην ίδια πλατφόρμα, επιτυγχάνεται γρηγορότερη εξερεύνηση του χώρου σχεδίασης μα αποτέλεσμα την υλοποίηση αποδοτικότερων συστημάτων. Στο σημείο αυτό θα πρέπει να σημειωθεί ότι η σημερινή πυκνότητα των ψηφιακών συστημάτων επιτρέπει την υλοποίηση ολόκληρων ενσωματωμένων συστημάτων σε μία μόνο ψηφίδα (System On Chip-SoC). Η πρακτική αυτή, επέτρεψε την εμφάνιση ενός νέου εμπορεύσιμου προϊόντος, αυτού των «Στοιχείων Πνευματικής Ιδιοκτησίας» (Intellectual Property Components-IP). Αυτά αποτελούν συνθέσιμες περιγραφές πολύπλοκων δομικών στοιχείων που απαντώνται σε ένα ενσωματωμένο σύστημα όπως επεξεργαστές, μνήμες, δίαυ-

λοι, εξειδικευμένο υλικό, οι οποίες μπορούν να διατεθούν από διαφορετικούς κατασκευαστές (IP vendors) στο σχεδιαστή, ο οποίος και τα συνδυάζει στην τελική σχεδίαση της ψηφίδας (System-On-Chip). Το τελικό προϊόν, εκφρασμένο σε γλώσσα περιγραφής υλικού, μπορεί κατόπιν να απεικονιστεί, μέσω αυτοματοποιημένης διαδικασίας, σε μία μόνο ψηφίδα (π.χ. FPGA). Η προσέγγιση αυτή άνοιξε νέους ερευνητικούς δρόμους στη σχεδίαση συστημάτων και καθιέρωσε μια νέα οικογένεια μεθοδολογιών σχεδίασης, που λίγο διαφέρουν όμως από αυτές της «Σχεδίασης Συστήματος βάσει πλατφόρμας» γνωστές με το όνομα «IP-Based Design". Ένα από τα βασικότερα πλεονεκτήματα που επέφερε αυτή η πρακτική, είναι και μεγάλη δυνατότητα, πλέον, επαναχρησιμοποίησης υλικού, μιας και η μετάβαση σε περιγραφές λογισμικού των δομικών συστημάτων καθιστά απλούστατη την διάθεσή τους μεταξύ σχεδιαστών ιδιαίτερα μέσω του διαδικτύου.

2.3 Ερευνητικοί Στόχοι Διδακτορικής Διατριβής

Η παρούσα διατριβή βασίζεται στην μεθοδολογία σχεδίασης Ε.Σ βάση πλατφόρμας και εστιάζεται στην σχεδίαση Ενσωματωμένων Συστημάτων που εξειδικεύονται σε εφαρμογές που απαιτούν τη χρήση τόσο του διαδικαστικού όσο και του δηλωτικού μοντέλου προγραμματισμού, στοχεύοντας στην βελτιστοποίηση της απόδοσής τους και στην απλοποίηση του τρόπου προγραμματισμού τους. Τέτοιες εφαρμογές απαντώνται κυρίως σε μεθοδολογίες Συντακτικής Αναγνώρισης Προτύπων και Τεχνητής Νοημοσύνης, σε πολλές εφαρμογές της σύγχρονης ζωής όπως στην ιατρική, στον αυτόματο έλεγχο, στη συλλογή και επεξεργασία πληροφοριών κ.α. Το χρησιμοποιούμενο μοντέλο για την αυτοματοποίηση της συσχεδίασης Υλικού/Λογισμικού είναι αυτό των Γραμματικών (Γραμματικών χωρίς συμφραζόμενα και Κατηγορηματικών Γραμματικών). Η προτεινόμενη μεθοδολογία βασίζεται στην απεικόνιση του συντακτικού τμήματος της γραμματικής σε υλικό (FPGA), του δε σημασιολογικού σε έναν μικροεπεξεργαστή. Η πρώτη προσπάθεια στηρίχθηκε στον ακολουθιακό αλγόριθμο συντακτικής αναγνώρισης του R. Floyd [2.9]. Στη συνέχεια, χρησιμοποιήθηκε ο πλέον αποδοτικός αλγόριθμος συντακτικής αναγνώρισης που υπάρχει, δηλαδή ο παράλληλος αλγό-

ριθμος του J. Earley. Ο τελευταίος βελτιώθηκε και επεκτάθηκε για να καλύψει και τη σημασιολογία της γραμματικής. Οι παραπάνω αλγόριθμοι απεικονίστηκαν σε διάφορες αρχιτεκτονικές, που το FPGA επικοινωνεί με εξωτερικό μικροεπεξεργαστή, ή ο μικροεπεξεργαστής έχει επίσης απεικονιστεί στο FPGA όπου γίνεται η συντακτική αναγνώριση.

Η υλοποίηση υβριδικών ακολουθιακών-δηλωτικών εφαρμογών σήμερα σε Ενσωματωμένα Συστήματα, παρουσιάζει δύο σημαντικά μειονεκτήματα:

- Η αρχική υβριδική διαδικαστική-δηλωτική εφαρμογή πρέπει αρχικά να χωριστεί σε δύο μέρη: το δηλωτικό και το ακολουθιακό. Το δηλωτικό μέρος απαιτεί την παρουσία ενός μεταγλωττιστή ο οποίος το μετατρέπει σε διαδικαστικό. Το παραγόμενο διαδικαστικό κομμάτι θα πρέπει κατόπιν να «συρραφτεί» με το αρχικό διαδικαστικό μέρος της εφαρμογής με στόχο τη μεταγλώττισή του για την τελική του εκτέλεση στον επεξεργαστή του Ενσωματωμένου Συστήματος. Η διαδικασία αυτή, αφενός αυξάνει την πολυπλοκότητα του παραγόμενου κώδικα δυσχεραίνοντας την αποσφαλμάτωση και τροποποίησή του και αφετέρου δυσχεραίνει τον εύκολο διαμοιρασμό δεδομένων μεταξύ διαδικαστικού και δηλωτικού κώδικα.

- Η εξολοκλήρου εκτέλεση της τελικής εφαρμογής σε λογισμικό (στον επεξεργαστή του Ενσωματωμένου Συστήματος) επιφέρει σημαντική μείωση στην απόδοση του τελικού συστήματος, λόγω των πολλών εντολών απόφασης και άλματος που απαιτεί η εκτέλεση του κώδικα. Η πρακτική αυτή, αποτρέπει την εκμετάλλευση παράλληλων τεχνικών που απαντώνται σε σύγχρονους επεξεργαστές.

Η μεθοδολογία που ακολουθήθηκε στην παρούσα Διδακτορική Διατριβή είναι το υβριδικό διαδικαστικό-δηλωτικό πρόβλημα να μετασχηματίζεται σε ισοδύναμη Κατηγορηματική Γραμματική. Οι Κατηγορηματικές Γραμματικές παρέχουν αυτή την δυνατότητα έκφρασης τόσο δηλωτικών όσο και διαδικαστικών χαρακτηριστικών. Συνεπώς, το πρόβλημα ανάγεται στη συντακτική και σημασιολογική

ανάλυση της παραγόμενης γραμματικής και στου καταλληλότερου τρόπου απεικόνισης του. Ως εκ τούτου, η ανάπτυξη της μεθοδολογίας διαχωρισμού συνιστωσών Υλικού/Λογισμικού καταλήγει στη διερεύνηση του εάν η συντακτική ανάλυση θα εκτελεστεί στο υλικό και η σημασιολογική στο λογισμικό ή αντίστροφα. Δύο είναι οι παράμετροι που επηρέασαν την απόφαση μας στον διαχωρισμό συνιστωσών Υλικού/Λογισμικού:

- Το πιο υπολογιστικά χρονοβόρο κομμάτι κώδικα είναι αυτό που πρέπει να εκτελεστεί στο υλικό.

- Το λιγότερο εξαρτώμενο από την εφαρμογή κομμάτι κώδικα είναι αυτό που μπορεί εύκολα να απεικονιστεί στο υλικό με την χρήση μιας γλώσσας περιγραφής υλικού.

Λαμβάνοντας υπόψη, αφενός ότι η συντακτική ανάλυση θα εκτελεστεί με τη χρήση ενός σταθερού αλγορίθμου συντακτικής ανάλυσης, ενώ οι σημασιολογικοί κανόνες μπορούν να εξαρτώνται από την εφαρμογή, όπως επίσης και το γεγονός ότι η συντακτική ανάλυση είναι μια αρκετά χρονοβόρα διαδικασία, επιλέχθηκε η συντακτική ανάλυση να εκτελείται σε υλικό και η σημασιολογική ανάλυση σε λογισμικό. Το παραπάνω θεωρητικό συμπέρασμα, επαληθεύτηκε σε αρκετές εφαρμογές.
Δεδομένης της απόφασης αυτής, δηλαδή η συντακτική ανάλυση να εκτελείται σε υλικό και η σημασιολογική ανάλυση σε λογισμικό, στη Μεθοδολογία απεικόνισης δηλωτικού και διαδικαστικού κώδικα σε Υλικό-Λογισμικό, δύο ήταν τα προβλήματα που έπρεπε να αντιμετωπιστούν

- Η απεικόνιση του αλγορίθμου συντακτικής ανάλυσης σε επαναπρογραμματιζόμενο Υλικό (Field Programmable Gate Arrays, FPGA) με χρήση γλώσσας περιγραφής υλικού.

- Η επέκταση του συντακτικού αναλυτή έτσι ώστε να καλύπτει την σημασιολογία της εφαρμογής, η επιλογή μικροεπεξεργαστή (εξωτερικού - hardcore, εσωτερικού - softcore/hard-wired) στον οποίο θα εκτελείται η σημασιολογική

ανάλυση καθώς και η αρχιτεκτονική επικοινωνίας του μικροεπεξεργαστή με την μονάδα που αναλαμβάνει την συντακτική ανάλυση.

Στα δύο αυτά προβλήματα θα επικεντρωθεί η ανάλυση στις παρακάτω ενότητες

Η παρούσα ερευνητική πρόταση εστιάζεται στην επίλυση των προαναφερθέντων προβλημάτων προτείνοντας τα παρακάτω:

- Στην ανάπτυξη ενός νέου εργαλείου [2.13], [2.14], [2.22] που συνδυάζει τόσο το δηλωτικό όσο και το διαδικαστικό μοντέλο προγραμματισμού υλοποιώντας σε υλικό έναν ακολουθιακό συντακτικό αναλυτή [2.9].

- Τη βελτίωση του καλλίτερου στην βιβλιογραφία , παράλληλου συντακτικού αναλυτή [2.10], [2.11] και υλοποίηση του σε υλικό [2.12], [2.16], [2.21].

- Την επέκταση και βελτίωση του παραπάνω παράλληλου συντακτικού αναλυτή για να καλύψει και τη σημασιολογία της γραμματικής [2.18], [2.20], [2.23].

- Την ανάπτυξη ενός νέου εργαλείου, με τεχνικές Συσχεδίασης Υλικού/Λογισμικού και ορίζοντας κατάλληλο interface μεταξύ των δύο, που συνδυάζει τόσο το δηλωτικό όσο και το διαδικαστικό μοντέλο προγραμματισμού βασισμένο στον παραπάνω βελτιωμένο παράλληλο συντακτικό αναλυτή [2.15], [2.17], [2.19].

- Την ανάπτυξη μιας πλατφόρμας αυτόματης παραγωγής Ενσωματωμένων Συστημάτων για τέτοιου είδους υβριδικές εφαρμογές, βασισμένης στο παραπάνω εργαλείο .

- Στην εφαρμογή των παραπάνω εργαλείων σε πρακτικές εφαρμογές [2.15], [2.16], [2.17], [2.18], [2.19].

2.4 Αναφορές

[2.1]. "Embedded System Design: A unifiedHw/Sw Introduction", John Wiley and SonsFrank Vahid, Tony Givargis, 2000

[2.2]. "A Framework for Hardware-Software Co-Design of Embedded Systems", available at: www-cad.eecs.berkeley.edu/~polis

[2.3]. "Specification and design of Embedded Systems", Daniel D. Gajski, Frank Vahid, Sanjiv Narayan, Jie Gong, Prentice Hall, 1994.

[2.4]. "Hardware Software Codesign Techniques for Embedded Systems", I. Panagopoulos, PhD thesis, National Technical University of Athens, 2004

[2.5]. "Introduction to High Level Synthesis", Daniel D. Gadjski, Longanath Ramachandran, Proc. IEEE Design and Test of Computers, pp.44-54,Winter, 1994

[2.6]. "Fundamentals of Digital Logic With VERILOG", Stephen Brown, Zvonko Vranesic, Design, Mc Graw Hill, 2003

[2.7]. "A New Approach for System-Level Architecture Exploration", Zivkovic, Vladimir and van der Wolf, Peter and Deprettere, Ed and de Kock, Erwin,, 2002

[2.8]. "Platform-Based Design and Software Design Methodology for Embedded Systems", Sangiovanni-Vincentelli, Alberto and Martin, Grant, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2002,December,volume 19,number 12,pp. 1523-1533

[2.9]. "The Syntax of Programming Languages-A Survey", R.W. Floyd, IEEE Transactions on Electr. Comp., Vol EC 13, No 4, August 1964

[2.10]. "An efficient context–free parsing algorithm", J. Earley, Communications of ACM, vol.13, pp. 94-102, 1970.

[2.11]. "Parallel parsing algorithms and VLSI implementation for syntactic pattern recognition", Y. Chiang and K. Fu, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-6, May 1984.

[2.12]. "Optimal Reconfigurable Embedded Parsers", C. Pavlatos, Alexandros Dimopoulos, Andrew Koulouris, Theodore Andronikos, Ioannis Panagopoulos and George Papakonstantinou, Computer Languages, Systems & Structures, DOI link: http://dx.doi.org/10.1016/j.cl.2007.08.001

[2.13]. "An Embedded System for Intelligent Control", I. Panagopoulos, C. Pavlatos and G. Papakonstantinou, Journal of Intelligent and Robotics Systems,vol.42,p.179-211,2005

[2.14]. "An Embedded System for Artificial Intelligence", I. Panagopoulos, C. Pavlatos, G. Papakonstantinou, International Journal of Computational Intelligence, vol.1 no1-4, 2004

[2.15]. "Hardware Natural Language Interface", Christos Pavlatos, Alexandros Dimopoulos and George Papakonstantinou, AIAI 2007, Athens 2007

[2.16]. "Efficient Signal Processing Using Syntactic Pattern Recognition Methods", Andrew Koulouris, Theodore Andronikos, Christos Pavlatos, Alexandros Dimopoulos, Ioannis Panagopoulos and George Papakonstantinou, SIP 2007, Honolulu USA,Aug. 2007

[2.17]. "An Efficient Hardware Implementation for AI applications", A. Dimopoulos, C. Pavlatos, I. Panagopoulos, G. Papakonstantinou,SETN'06, Heraklion, Crete, May 2006

[2.18]. "An Intelligent Embedded System for Control Applications", C. Pavlatos, A. Dimopoulos, G. Papakonstantinou, Workshop on Modeling and Control of Complex Systems, Cyprus, July 2005

[2.19]. "An embedded system for the electrocardiogram recognition", C. Pavlatos, A. Dimopoulos, G. Papakonstantinou, EMBEC'05, Prague, Czech Republic, November 2005

[2.20]. "Knowledge Representation using a Modified Earley's Algorithm", C. Pavlatos, I. Panagopoulos, G, Papakonstantinou, Lectures of Notes in Artificial Intelligence, Springer Verlag, pp.321-330, 2004

[2.21]. "A programmable Pipelined Coprocessor for Parsing Applications", C. Pavlatos, I. Panagopoulos, G, Papakonstantinou, Workshop on Application Specific Processors (WASP), Stockholm, Sept. 2004

[2.22]. "An extended RISC microprocessor for Attribute Grammar Evaluation", I. Panagopoulos, C. Pavlatos, G. Papakonstantinou, SAC2004, ACM Symposium on Applied Computers, Nicosia, Cyprus, 2004

[2.23]. "Hardware Implementation of Syntactic Pattern Recognition Algorithms", C.Pavlatos, I. Panagopoulos, G Papakonstantinou, The International Association of Science and Technology for Development Conference (SPPRA 2003), Rhodes, Greece, 2003

3. Χρησιμοποιούμενο Μοντέλο Αυτοματοποίησης Υλικού/Λογισμικού

Στην παρούσα Διδακτορική Διατριβή το χρησιμοποιούμενο μοντέλο για την αυτοματοποίηση της συσχεδίασης Υλικού/Λογισμικού είναι αυτό των Γραμματικών. Οι τελευταίες μπορούν να χρησιμοποιηθούν σε πλείστες όσες εφαρμογές Τεχνητής Νοημοσύνης και Συντακτικής Αναγνώρισης Προτύπων. Η προτεινόμενη μεθοδολογία βασίζετε στην απεικόνιση του συντακτικού τμήματος της γραμματικής σε υλικό (FPGA) ενώ του σημασιολογικού σε έναν μικροεπεξεργαστή.

Οι Γραμματικές χωρίς Συμφραζόμενα [3.1] (Context Free Grammars CFGs) αποτελούν ένα πολύ ισχυρό εκφραστικό εργαλείο στη διάθεση της επιστήμης των υπολογιστών. Παρουσιάστηκαν για πρώτη φορά τη δεκαετία του '50 και έκτοτε έχουν βρει εφαρμογή σε μια πλειάδα θεμάτων. Η πιο σημαντική από αυτές είναι σίγουρα η συνεισφορά τους στο σχεδιασμό και στην υλοποίηση των γλωσσών προγραμματισμού υψηλού επιπέδου. Οι πιο γνωστές γλώσσες προγραμματισμού σήμερα, περιγράφονται με την βοήθεια των CFGs. Επιπλέον, εφαρμογές που ανήκουν σε ερευνητικές περιοχές όπως ανάλυση εικόνας, αναγνώριση φωνής, ανάλυση ηλεκτροκαρδιογραφήματος, αναπαράσταση γνώσης κτλ, βασίζονται σε μεγάλο βαθμό στη θεωρία συντακτικής αναγνώρισης.

Λόγω του μεγάλου εύρους εφαρμογών, γρήγορα αναπτύχθηκαν αποδοτικοί αλγόριθμοι συντακτικής αναγνώρισης. Πολλοί απ' αυτούς καταφέρνουν αναγνώριση με γραμμική χρονική πολυπλοκότητα, εφαρμόζονται όμως σε υπό-κλάσεις του συνόλου των CFGs. Οι δυο πιο γνωστοί αλγόριθμοι συντακτικής αναγνώρισης γενικών CF γλωσσών παρουσιάστηκαν στις εργασίες των Earley [3.2] και

Cocke-Younger-Kassami (CYK) [3.3]. Και οι δύο έχουν χρονική πολυπλοκότητα $O(n^3)$, όπου n το μήκος της συμβολοσειράς εισόδου. Παρόλα αυτά ο CYK απαιτεί τον μετασχηματισμό της γραμματικής σε Chomsky Normal Form (CNF). Όλες οι CFGs μπορούν να μετασχηματιστούν σε CNF με το απαιτούμενο υπολογιστικό κόστος όμως να αυξάνεται. Αντίθετα ο αλγόριθμος του Earley δεν απαιτεί οι γραμματικές να υποστούν κανένα μετασχηματισμό. Οι παραπάνω αλγόριθμοι αποτέλεσαν αντικείμενο μελέτης από πολλούς μεταγενέστερους ερευνητές και ποικίλες παραλλαγές αυτών έχουν παρουσιαστεί κατά καιρούς [3.4]. Η σημαντικότερη αυτών παρουσιάστηκε από τους Chiang-Fu [3.5], οι οποίοι πρότειναν τον Weakened Earley αλγόριθμο, μια πλήρως παραλληλοποιήσιμη παραλλαγή του αλγορίθμου του Earley.

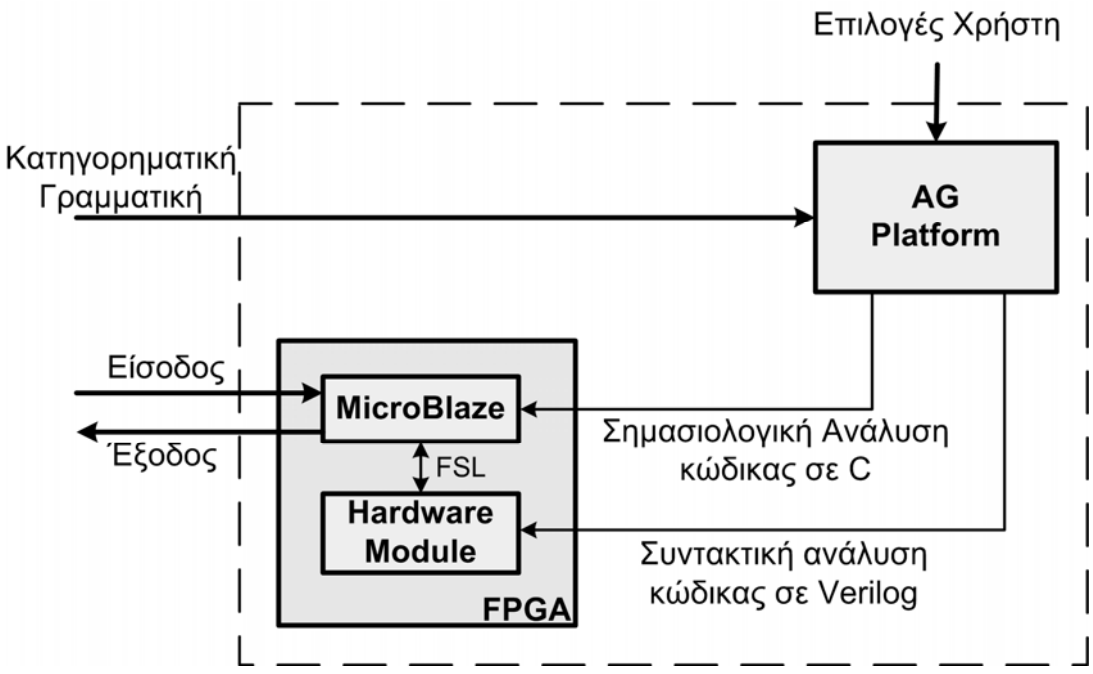
Δύο ήταν οι βασικοί στόχοι της διατριβής αυτής, ο πρώτος στόχος της διατριβής ήταν η δημιουργία ενός εργαλείου το οποίο θα δέχεται ως είσοδο τις προδιαγραφές μια CF γραμματικής και θα παράγει αυτόματα την περιγραφή του CF συντακτικού αναλυτή (parser) σε συνθέσιμο πηγαίο κώδικα Verilog-HDL. Αυτό το εργαλείο χρησιμοποιεί τρεις προτεινόμενες αρχιτεκτονικές (βλέπε κεφάλαιο 4), σε μια μορφή προτύπων. Το εργαλείο τροποποιεί κατάλληλα την επιλεγμένη από τον χρήστη πρότυπη αρχιτεκτονική, σύμφωνα με τη CFG εισόδου, προκειμένου να κατασκευαστεί ο κατάλληλος hardware parser για τη συγκεκριμένη γραμματική.

Ο χρήστης παρέχει τις προδιαγραφές της γραμματικής και μπορεί να επιλέξει τη αρχιτεκτονική του παραγόμενου hardware parser μεταξύ τριών διαφορετικών αρχιτεκτονικών υλοποίησης : μιας σειριακής, μιας με pipelined τεχνικές, και μιας παράλληλης που θα ενσωματώνει την pipelined υλοποίηση [2.6]. Και οι τρεις παραπάνω αρχιτεκτονικές είναι πλήρως ανεξάρτητες από τη γραμματική προκειμένου να χρησιμοποιηθούν ως πρότυπα για το αυτοματοποιημένο εργαλείο σύνθεσης των CFG hardware parsers. Σε περίπτωση που η απόδοση είναι η κρίσιμη πτυχή της εφαρμογής και δεν υπάρχει κανένας περιορισμός σχετικά με τον απαιτούμενο αριθμό στοιχείων επεξεργασίας, η παράλληλη αρχιτεκτονική θα πρέπει να προτιμηθεί. Οι άλλες δύο αρχιτεκτονικές (σειριακή και pipelining) επιτυγχάνουν χαμηλότερη απόδοση αλλά απαιτούν σαφώς λιγότερο υλικό προκει-

μένου να εφαρμοστούν. Τέλος ο παράλληλος αλγόριθμος συντακτικής ανάλυσης απεικονίστηκε σε ένα προτεινόμενο συνδυαστικό κύκλωμα αυξάνοντας περαιτέρω την απόδοση του συντακτικού αναλυτή.

Ο δεύτερος στόχος της διατριβής αυτής ήταν η ανάπτυξη αλγορίθμου (βλέπε Κεφάλαιο 5) αποτίμησης των κατηγορημάτων (σημασιολογική ανάλυση) που θα εκτελείται παράλληλα με την συντακτική ανάλυση στον μικροεπεξεργαστή που θα επικοινωνεί με το hardware module του συντακτικού αναλυτή.

Η αρχιτεκτονική υλοποίησης της Διδακτορικής Διατριβής και κατ' επέκταση των δυο παραπάνω στόχων φαίνεται στο Σχήμα 1. Ο χρήστης παρέχει τις προδιαγραφές της Κατηγορηματικής Γραμματικής και το εργαλείο (AG Platform) αυτόματα παράγει κώδικα C που υλοποιεί την σημασιολογική ανάλυση και κώδικα σε γλώσσα περιγραφής υλικού (Verilog) που υλοποιεί την συντακτική ανάλυση. Και τα δυο τμήματα κώδικα απεικονίζονται σε επαναπρογραμματιζόμενο υλικό (FPGA). Δεδομένης της συγγένειας των Κατηγορηματικών Γραμματικών και των προγραμμάτων PROLOG, παράλληλος στόχος της διδακτορικής διατριβής είναι η αποδοτική εκτέλεση προγραμμάτων PROLOG μέσω του παραγόμενου εργαλείου.



Σχήμα 1. Προτεινόμενη αρχιτεκτονική υλοποίησης διατριβής

Στην συνέχεια του κεφαλαίου αυτού δίνονται βασικοί ορισμοί και έννοιες οι οποίοι θα συνεισφέρουν στην κατανόηση της θεωρίας της συντακτικής αναγνώρισης προτύπων από τον αναγνώστη.

3.1 Γραμματικές χωρίς Συμφραζόμενα

Ένα οποιοδήποτε μη κενό και πεπερασμένο σύνολο Σ αποτελούμενο από σύμβολα ονομάζεται αλφάβητο. Κάθε στοιχείο του αλφάβητου Σ ονομάζεται σύμβολο. Μια πεπερασμένη παράθεση από σύμβολα ονομάζεται συμβολοσειρά (string). Το σύνολο όλων των συμβολοσειρών ενός αλφαβήτου Σ παριστάνεται με Σ^*.

Ορισμός 3.1.

Έστω ένα αλφάβητο Σ. **Γλώσσα (*language*)** επί του αλφάβητου Σ ονομάζουμε κάθε σύνολο συμβολοσειρών του Σ δηλαδή κάθε υποσύνολο του Σ^*.

Ο παραπάνω ορισμός μιας γλώσσας ως ένα οποιοδήποτε σύνολο ενός αλφάβητου είναι πολύ ευρύς. Στην επιστήμη των υπολογιστών περιοριζόμαστε σε στενότερες κατηγορίες γλωσσών για τις οποίες μπορούν να κατασκευαστούν προγράμματα που να αποφαίνονται αν μια τυχαία συμβολοσειρά ανήκει ή όχι σε αυτές. Ονομάζουμε γραμματική (*grammar*) ένα σύστημα παραγωγής συμβολοσειρών. Ο τυπικός ορισμός μιας γραμματικής είναι ο εξής :

Ορισμός 3.2.

Γραμματική είναι μια διατεταγμένη τετράδα της μορφής (T,N,P,S) όπου :

- Τ είναι ένα αλφάβητο του οποίου τα μέλη ονομάζονται *τερματικά σύμβολα (terminal symbols*).
- Ν είναι ένα αλφάβητο του οποίου τα μέλη ονομάζονται *μη-τερματικά σύμβολα* (*non-terminal symbols*).
- Τα σύνολα Τ και Ν είναι ξένα μεταξύ τους.

- P είναι ένα πεπερασμένο σύνολο *κανόνων παραγωγής*. Οι κανόνες παραγωγής είναι διατεταγμένα ζεύγη (α,β) συμβολοσειρών του αλφαβήτου που προκύπτει από την ένωση των συνόλων Τ,Ν.
- S είναι ένα στοιχείο του Ν, το οποίο ονομάζεται *αρχικό σύμβολο* της γραμματικής.

Το αλφάβητο Τ είναι το αλφάβητο της γλώσσας που θα παραχθεί. Τα σύμβολα του αλφαβήτου Ν δεν εμφανίζονται στην τελική γλώσσα, αλλά έχουν το ρόλο των μεταβλητών και χρησιμοποιούνται μόνο για τον ορισμό των κανόνων.
Ένας κανόνας παραγωγής (α,β) συμβολίζεται συνήθως με α→β. Η συμβολοσειρά α ονομάζεται *αριστερό* ή *παράγον* μέλος του κανόνα, ενώ η συμβολοσειρά β ονομάζεται *δεξιό* ή *παραγόμενο* μέλος του κανόνα.

Η γραμματική παράγει συμβολοσειρές ως εξής :
Αρχίζουμε με τη συμβολοσειρά που περιέχει μόνο το αρχικό σύμβολο. Από τη τρέχουσα συμβολοσειρά παράγουμε μια καινούργια, αντικαθιστώντας κάποια υπό-συμβολοσειρά της που αντιστοιχεί σε αριστερό μέλος κανόνα, με το αντίστοιχο δεξιό μέλος. Επαναλαμβάνουμε όσες φορές χρειαστεί. Αν καταλήξουμε σε συμβολοσειρά που να αποτελείται μόνο από τερματικά σύμβολα, τότε λέμε ότι αυτή η συμβολοσειρά παράγεται από την γραμματική

Ορισμός 3.3.
Μια γλώσσα L θα λέγεται ***τυπική (formal)*** αν υπάρχει γραμματική G που να την παράγει.
Οι γραμματικές, και αντίστοιχα οι γλώσσες που παράγονται από αυτές κατατάσσονται σε τέσσερις κλάσεις ανάλογα με τη μορφή των κανόνων τους. Οι κλάσεις αυτές συνθέτουν μια ιεραρχία με την έννοια ότι κάθε κλάση είναι υπερσύνολο των επόμενων της. Η ιεραρχία αυτή ονομάζεται ιεραρχία Chomsky, από το γλωσσολόγο Noam Chomsky που την όρισε :

- ***Γραμματικές τύπου 0 :*** Στην κλάση αυτή ανήκουν όλες οι γραμματικές χωρίς κανένα περιορισμό.
- ***Γραμματικές τύπου 1 ή γραμματικές με συμφραζόμενα(context-sensitive grammars):*** Στην κλάση αυτή ανήκουν οι γραμματικές με κανόνες της μορφής $\alpha \rightarrow \beta$, όπου η συμβολοσειρά α περιέχει τουλάχιστον ένα μη-τερματικό σύμβολο και $|\alpha| < |\beta|$. Προκειμένου να μπορούν οι παραγόμενες γλώσσες να περιέχουν την κενή συμβολοσειρά επιτρέπεται ο κανόνας $S \rightarrow \varepsilon$, όπου S το *αρχικό σύμβολο* της γραμματικής υπό την προϋπόθεση ότι αυτό δεν βρίσκεται στο δεξί μέλος κανενός κανόνα παραγωγής.
- ***Γραμματικές τύπου 2 ή γραμματικές χωρίς συμφραζόμενα (context-free grammars):*** Στην κλάση αυτή ανήκουν οι γραμματικές με κανόνες μορφής $A \rightarrow \alpha$, όπου A ένα μη τερματικό σύμβολο και α συμβολοσειρά. Στις γραμματικές αυτές επιτρέπεται οι κανόνες να έχουν κενό δεξιό μέλος καθώς αυτό δεν επηρεάζει την εκφραστικότητα του μοντέλου.
- ***Γραμματικές τύπου 3 ή κανονικές γραμματικές (regular grammars):*** Στην κλάση αυτή ανήκουν οι γραμματικές με κανόνες που έχουν μία από τις ακόλουθες μορφές $A \rightarrow aB$ ή $A \rightarrow a$ ή $A \rightarrow \varepsilon$, όπου A, B μη τερματικά σύμβολα και a τερματικό σύμβολο.

Γενικότερα ισχύουν οι παρακάτω συμβάσεις για την χρήση γραμμάτων :

- Τα πεζά γράμματα του λατινικού αλφαβήτου a, b, c, d…, συμβολίζουν τερματικά σύμβολα.
- Τα κεφαλαία γράμματα του λατινικού αλφαβήτου A, B, C, D…, συμβολίζουν μη-τερματικά σύμβολα.
- Τα πεζά γράμματα του ελληνικού αλφαβήτου α, β, γ, δ…, αναφέρονται σε συμβολοσειρά που μπορεί να περιέχει τόσο τερματικά όσο και μη-τερματικά σύμβολα.

Ορισμός 3.4.

Γραμματική χωρίς συμφραζόμενα (context-free grammar CFG) είναι μια διατεταγμένη τετράδα της μορφής (V,N,P,S) όπου :

- V (*vocabulary*) είναι ένα πεπερασμένο σύνολο όλων των συμβόλων της γραμματικής
- N είναι το σύνολο των *non-terminal* συμβόλων. Ισχύει T=V-N.
- $P \subseteq N \times V^{\star}$ είναι ένα πεπερασμένο σύνολο κανόνων της μορφής $A \rightarrow \alpha$, όπου $A \in N$ και $\alpha \in V^{\star}$. Το A ονομάζεται σύμβολο αριστερού μέρους (*lhss*) του κανόνα και το α ονομάζεται δεξιό μέρος του κανόνα (*rhss*).
- S είναι το αρχικό (*non-terminal*)σύμβολο της γραμματικής.

Για να περιγράψουμε πώς με μια γραμματική προκύπτει μια συμβολοσειρά εφαρμόζουμε συνήθως δυο τακτικές : i) κατασκευή *ακολουθίας παραγωγής*, ii)κατασκευή *συντακτικού δένδρου*.

Ορισμός 3.5.

Έστω οι συμβολοσειρές $\gamma_1 \alpha \gamma_2$ και $\gamma_1 \beta \gamma_2$ και μια γραμματική G. Λέμε ότι η $\gamma_1 \alpha \gamma_2$ *παράγει* την $\gamma_1 \beta \gamma_2$ και συμβολίζουμε με $\gamma_1\alpha\gamma_2 \rightarrow \gamma_1\beta\gamma_2$ αν ο $\alpha \rightarrow \beta$ είναι κανόνας παραγωγής της γραμματικής G.

Ορισμός 3.6.

Έστω G={N,T,P,S} context-free γραμματική. Ένα δένδρο είναι συντακτικό δένδρο της G αν :

- Κάθε κόμβος του δένδρου έχει μια επιγραφή η οποία είναι ένα σύμβολο που ανήκει στην ένωση των συνόλων N, T, ε.
- Η επιγραφή της ρίζας είναι το S.
- Αν ένας κόμβος είναι εσωτερικός και έχει επιγραφή A τότε θα πρέπει το A να ανήκει στο N.

- Αν ο κόμβος n έχει επιγραφή A και οι κόμβοι n_1, n_2, n_3… είναι παιδιά του n, σε διάταξη από τα αριστερά προς τα δεξιά, με επιγραφές X_1, X_2, X_3… αντίστοιχα τότε ο $A \rightarrow X_1X_2X_3$… πρέπει να είναι κανόνας παραγωγής του P.
- Αν ο κόμβος έχει επιγραφή ε, τότε είναι φύλλο και είναι το μοναδικό παιδί του γονέα του.

Μια ιδιότητα των context-free γραμματικών *G={V,T,P,S}* είναι ότι το *S* μπορεί να παράγει μετά από τα απαραίτητα βήματα την συμβολοακολουθία *α* αν και μόνο αν υπάρχει συντακτικό δένδρο με φύλλωμα το *α*.

Ορισμός 3.7.

Μια γραμματική χωρίς συμφραζόμενα ονομάζεται διφορούμενη (ambiguous context-free grammar) όταν μπορούν να παραχθούν περισσότερα του ενός συντακτικά δέντρα για την ίδια συμβολοακολουθία

3.2 Κατηγορηματικές Γραμματικές

Οι Κατηγορηματικές Γραμματικές (ΚΓ) [3.8] είναι βασισμένες στις γραμματικές χωρίς συμφραζόμενα.

Ορισμός 3.8.

Μια ΚΓ είναι μια διατεταγμένη τετράδα της μορφής AG = {G, A, SR, d} όπου:

- G είναι μια CFG.
- $A = \cup A(X)$ όπου A(X) είναι ένα πεπερασμένο σύνολο από κατηγορήματα που σχετίζονται με το σύμβολο $X \in V$.
- Κάθε μια από τις παραγωγές $p \in P$ (p: X0 → X1 … Xn) έχει ένα σύνολο SR(p) από σημασιολογικούς κανόνες που βοηθούν στη αποτίμηση των κατηγορημάτων των συμβόλων του κανόνα.
- d είναι μια συνάρτηση που δίνει για κάθε ένα κατηγόρημα a το πεδίο ορισμού του d (a).

Ο συμβολισμός X.a χρησιμοποιείται για να δείξει ότι το κατηγόρημα α είναι ένα στοιχείο του A (X). Το A(X) χωρίζεται σε δύο σύνολα; το σύνολο των συντιθέμενων κατηγορημάτων AS(X) και το σύνολο των κληρονομούμενων κατηγορημάτων AI(X). Συντιθέμενα κατηγορήματα X.s είναι εκείνα των οποίων οι τιμές καθορίζονται από κατηγορήματα των κόμβων παιδίων του κόμβου X στο αντίστοιχο συντακτικό δέντρο. Κληρονομούμενα κατηγορήματα X.i είναι εκείνα των οποίων οι τιμές καθορίζονται από κατηγορήματα του κόμβου γονέα του κόμβου X στο αντίστοιχο συντακτικό δέντρο. Το σύμβολο έναρξης δεν έχει κληρονομούμενα κατηγορήματα.

Οι συντακτικοί κανόνες της κατηγορικής γραμματικής ορίζουν όλες τις πιθανές παραγωγές από ένα συγκεκριμένο μη τερματικό σύμβολο. Αν μόνο τερματικά σύμβολα χρησιμοποιούνται για τον προσδιορισμό της επιτυχίας ή αποτυχίας της συντακτικής ανάλυσης (βάσει συγκρίσεων των τερματικών συμβόλων με τις δομές της συμβολοσειράς εισόδου), τότε η διαδικασία ονομάζεται συντακτική ανάλυση (parsing). Σε περίπτωση που μαζί με τα τερματικά σύμβολα χρησιμοποιούνται και οι τιμές κάποιων κατηγορημάτων, η διαδικασία ονομάζεται σημασιολογικά ελεγχόμενη συντακτική ανάλυσης (semantically driven parsing). Είναι δυνατό να απαλειφθούν όλα τα τερματικά σύμβολα της κατηγορικής γραμματικής (να αντικατασταθούν με το σύμβολο nil, δηλαδή το κενό σύμβολο) και η πληροφορία τους να αποθηκευτεί σε τιμές κατηγορημάτων στα φύλλα του δέντρου (κατηγορικές γραμματικές οριστικών προτάσεων (definite clause) [3.17]) ούτως ώστε να είναι δυνατό να πραγματοποιηθεί σημασιολογικά ελεγχόμενη συντακτική ανάλυση χωρίς τη χρήση τερματικών συμβόλων. Τέλος, αν δεν υπάρχει συμβολοσειρά εισόδου και ούτε τερματικά σύμβολα, τότε η συντακτική ανάλυση είναι εκφυλισμένη (degenerate) και όλες οι παραγωγές δέντρων πραγματοποιούνται βάσει των τιμών των κατηγορημάτων.

3.3 Λογικά Προγράμματα & Κατηγορηματικές Γραμματικές

Εκτενείς προσπάθειες στην υλοποίηση μηχανών για εφαρμογές λογικού προγραμματισμού απαντώνται κυρίως στις προσπάθειες υλοποίησης υπολογιστικών μηχανών 5ης γενεάς, οι οποίες οραματίζονταν ένα σύνολο από συνδεδεμένες παράλληλες μηχανές κατάλληλες για εφαρμογές Τεχνητής Νοημοσύνης [3.10], [3.11]. Παρουσιάστηκαν ισχυρές μηχανές βασισμένες σε υπολογιστές UMA και NUMA [3.9], [3.10] στην προσπάθεια αύξησης της απόδοσης μέσω παραλληλισμού των δηλωτικών προγραμμάτων λογικής σε PROLOG. Αν και η συνολική απόδοση του συστήματος αυξήθηκε, μέσω αυτών των προσπαθειών, το κόστος αυτών των υλοποιήσεων μαζί με το μέγεθός τους, απέτρεψε τη χρησιμοποίησή τους σε εφαρμογές μικρότερου μεγέθους που απαιτούνται σε Ε.Σ. Επιπροσθέτως, οι μηχανές αυτές βελτιστοποιήθηκαν μόνο για εφαρμογές λογικού προγραμματισμού με αποτέλεσμα να μην είναι πάντα ικανοποιητικές για άλλες πιθανές εφαρμογές. Η εμφάνιση των Ε.Σ φαίνεται να οδηγεί σε νέες δυνατότητες και απαιτήσεις στην υλοποίηση επεξεργαστών με βελτιστοποιημένα στοιχεία για εφαρμογές λογικού προγραμματισμού. Τα ενσωματωμένα συστήματα δεν στοχεύουν στη γενικότητα αφού είναι περισσότερο προσανατολισμένα σε υλοποιήσεις εξειδικευμένων εφαρμογών. Επιπροσθέτως, οι περιορισμοί που εφαρμόζονται σε τέτοια συστήματα και η απαιτούμενη υπολογιστική δύναμη καθιστά προσεγγίσεις που στοχεύουν σε αύξηση της απόδοσης χρήσιμες και επιτακτικές. Ως αποτέλεσμα, η προσπάθεια σχεδίασης επεξεργαστικών μονάδων ικανές να υποστηρίξουν αποδοτικά τον δηλωτικό τρόπο προγραμματισμού που χρησιμοποιείται σε λογικά προγράμματα μπορεί να οδηγήσει στην υλοποίηση αποδοτικών «έξυπνων» συστημάτων σε σχέση με τα παραδοσιακά που υποστηρίζουν τον ακολουθιακό τρόπο προγραμματισμού. Οι κατηγορικές γραμματικές χρησιμοποιήθηκαν εκτενώς σε εφαρμογές λογικού προγραμματισμού [3.13], [3.18] [3.19]. Στις εργασίες [3.15] [3.16] παρουσιάζεται μία αποδοτική μέθοδος βασισμένη στον συντακτικό αναλυτή του Floyd [3.20] η οποία μετατρέπει το αρχικό λογι-

κό πρόγραμμα στο συμπεριφορικά του ισοδύναμο εκφρασμένο με κατηγορικές γραμματικές. Η μέθοδος αυτή εισάγει έναν αριθμό από συμπεριφορικά ισοδύναμους συντακτικούς κανόνες για τους λογικούς κανόνες και έναν αριθμό από κατηγορήματα και σημασιολογικούς κανόνες για τη διαδικασίας ενοποίησης (unification process). Το τελικό σύστημα μπορεί να χρησιμοποιηθεί ως πλήρης μηχανή λογικού προγραμματισμού και είναι υλοποιημένο σε λογισμικό. Οι βασικές αρχές που διέπουν την προσέγγιση αυτή δίνονται παρακάτω: Κάθε λογικός κανόνας, στο αρχικό λογικό πρόγραμμα, μπορεί να μετατραπεί σε έναν ισοδύναμο συντακτικό κανόνα ο οποίος αποτελείται μόνο από μη τερματικά σύμβολα. Για παράδειγμα ο κανόνας:

$R_O(...) \leftarrow R_1(...) \wedge ... \wedge R_m(...)$ μετατρέπεται στον $R_O \rightarrow R_1 ... R_m |$.

(το σύμβολο "|." εκφράζει το τέλος του συντακτικού κανόνα). Στην περίπτωση που υπάρχουν δύο ή περισσότερες εναλλακτικές λύσεις για τον ίδιο λογικό κανόνα, αυτός μεταφράζεται σε έναν συντακτικό κανόνα με ίσο αριθμό εναλλακτικών τρόπων ανάλυσης.

Τέλος, τα γεγονότα (facts) των λογικών κανόνων μετατρέπονται σε τερματικούς κανόνες που δεν έχουν τερματικό σύμβολο. Για παράδειγμα, τα γεγονότα:

$R_g(a,b), R_g(c,d), R_g(e,f)$ μετατρέπονται στον κανόνα: $R_g \rightarrow ||||$.

Προφανώς η συντακτική ανάλυση είναι εκφυλισμένη αφού δεν υπάρχουν τερματικά σύμβολα.

Για κάθε μεταβλητή στους λογικούς κανόνες, ορίζονται δύο κατηγορήματα, ένα synthesized και ένα inherited. Τα κατηγορήματα αυτά χρησιμοποιούνται στη διαδικασία ενοποίησης. Οι κανόνες αποτίμησης κατηγορημάτων σχηματίζονται βάσει του αρχικού λογικού προγράμματος. Μία λεπτομερέστερη περιγραφή του τρόπου με τον οποίο πραγματοποιείται ο μετασχηματισμός αυτός δίνεται στις εργασίες [3.15] [3.16] [3.19], ο οποίος μάλιστα μπορεί εύκολα να πραγματοποιηθεί με τη χρήση αυτοματοποιημένου εργαλείου. Τα κατηγορήματα στα «φύλλα» του δέντρου παίρνουν τις τιμές των σταθερών στα γεγονότα του λογικού προγράμματος. Η λογική διαδικασία πραγματοποιείται κατά τη δημιουργία του συντακτικού δέντρου, και μία συνάρτηση EVAL καλείται σε κάθε δημιουργία/προσπέλαση ενός κόμβου και υπολογίζει τα συσχετισμένα με αυτόν κατηγο-

ρήματα. Σημασιολογικοί έλεγχοι βασίζονται στις τιμές των κατηγορημάτων και ελέγχονται ώστε να ορίσουν την επιτυχία ή αποτυχία της λογικής διαδικασίας (μία μεταβλητή κατάστασης με όνομα flag χρησιμοποιείται για αυτόν ακριβώς τον σκοπό). Γενικά, η χρήση κανόνων αποτίμησης κατηγορημάτων τα οποία ελέγχουν τη δημιουργία του συντακτικού δέντρου, οδηγούν στη δημιουργία ενός πλήρους εκφυλισμένου σημασιολογικά ελεγχόμενου συντακτικού αναλυτή, ο οποίος μπορεί να χρησιμοποιηθεί αποδοτικά σε εφαρμογές λογικού προγραμματισμού. Επιπρόσθετοι σημασιολογικοί έλεγχοι μπορούν επίσης να υπάρξουν στις λογικές παραγωγές. Οι επιπρόσθετοι αυτοί κανόνες οδηγούν τη λογική διαδικασία πέρα από την τυπική που απαντάται στη γλώσσα PROLOG, οδηγώντας στην υλοποίηση αποδεικτών θεωρημάτων (theorem provers) [3.21] και λογικών μηχανών με ασάφεια και αβεβαιότητα [3.22]. Για την καλύτερη κατανόηση του προαναφερθέντος μετασχηματισμού, παραθέτουμε ένα παράδειγμα λογικού προγραμματισμού και του τρόπου μετασχηματισμού του στο συμπεριφορικά ισοδύναμό του σε αποτιμητή κατηγορικών γραμματικών.

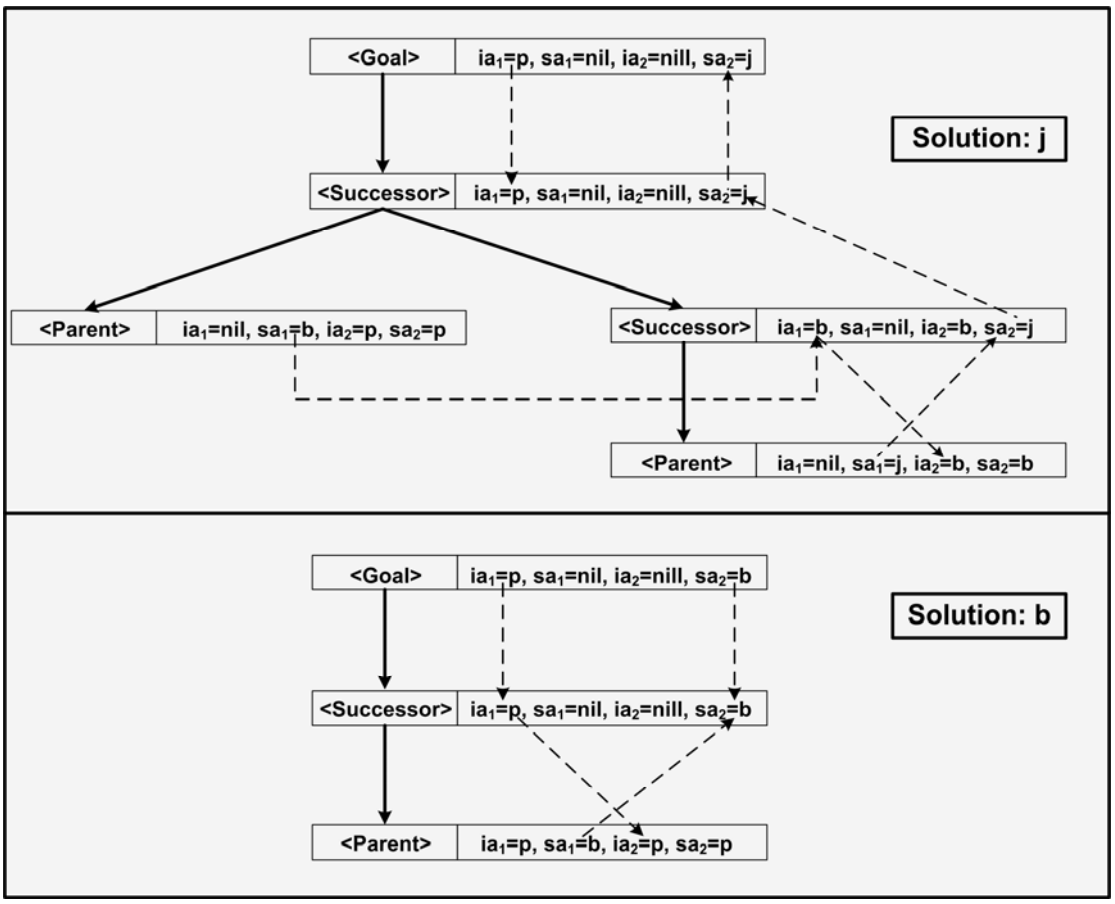
Πίνακας 1. Λογικό πρόγραμμα και συμπεριφορικά ισοδύναμο πρόγραμμα κατηγορηματικής γραμματικής

Ορισμός της βάσης γνώσης	**Ισοδύναμο πρόγραμμα σε κατηγορηματική γραμματική**

Inference Rules
Goal (X,Y) if Successor (X,Y)
Successor (X,Y) if Parent (Z,X) and Successor (Z,Y)
Successor(X,Y) if Parent (Y,X)
Facts
Parent (j,b)
Parent (j,l)
Parent (b,a)
Parent (b,p)
	Goal = Successor |.
	Successor.ia1=Goal.ia1;
	Goal.sa2=Successor.sa2;
	Successor = Parent Successor | Parent |.
	Parent[1].ia2=Successor[1].ia1;
	Successor[2].ia1=Parent.sa1;
	Parent[2].ia2=Successor[1].ia1;
	Successor[1].sa2=Parent[2].sa1;
	Parent = ||||.
	nil1: if ((Parent.ia1!=nil) && (Parent.ia1!="j"))
	flag=0; else Parent.sa1="j";
	if ((Parent.ia2!=nil) && (Parent.ia2!="b"))
	flag=0; else Parent.sa2="b";
	nil2: if ((Parent.ia1!=nil) && (Parent.ia1!="j"))
	flag=0; else Parent.sa1="j";
	if ((Parent.ia2!=nil) && (Parent.ia2!="l"))
	flag=0; else Parent.sa2="l";
	…

Δίνεται μία βάση γνώσης, η οποία παρουσιάζεται στον Πίνακας 1 (πρώτη στήλη) και θέλουμε να θέσουμε την ερώτηση: "ποιος είναι ο απόγονος (successor) του p;" δηλαδή Successor (p,?). Οι συντακτικοί κανόνες που σχηματίζουν το ισοδύναμο πρόγραμμα κατηγορηματικής γραμματικής παρουσιάζονται στον Πίνακας 1 (Δεύτερη στήλη) μαζί με τους κανόνες αποτίμησης κατηγορημάτων και τους σημασιολογικούς ελέγχους που χρησιμοποιούνται για τη διαδικασία ενοποίησης.

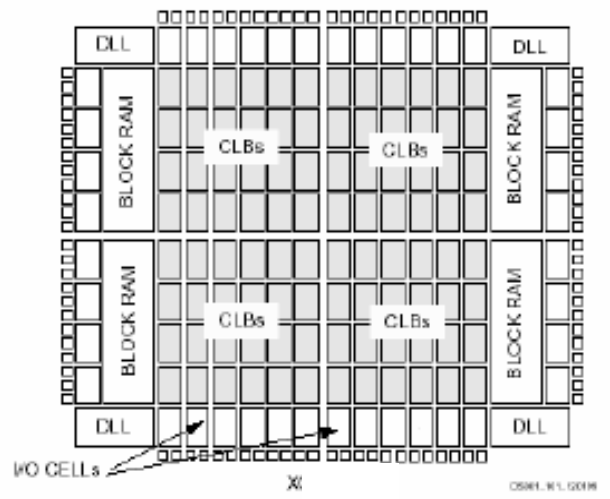
Η ερώτηση που ζητείται έχει δύο λύσεις οι οποίες είναι οι "j" και "b". Τα αντίστοιχα συντακτικά δέντρα μαζί με τα κατηγορήματα που χρησιμοποιήθηκαν για τη λογική διαδικασία δίνονται στο Σχήμα 4.



Σχήμα 2. Παραγόμενα συντακτικά δέντρα που δίνουν λύση στο λογικό Πρόγραμμα

3.4 Χρησιμοποιούμενα εργαλεία

Οι πρότυπες αρχιτεκτονικές του συντακτικού αναλυτή σχεδιάστηκαν στο ολοκληρωμένο περιβάλλον σχεδίασης Xilinx ISE 8.1 [2.7]. Κατόπιν με το εργαλείο XST (Xilinx Synthesis Tool) παράχθηκε αρχείο συνθέσιμου κώδικα το οποίο, αφού μετασχηματιστεί σε κατάλληλη bit-μορφή φορτώνεται στην FPGA συσκευή στόχου (FPGA target device). Η FPGA συσκευή στόχου, που χρησιμοποιήθηκε, ανήκει στην οικογένεια Spartan της Xilinx (βλέπε Σχήμα 3).



Σχήμα 3. Το FPGA Spartan2

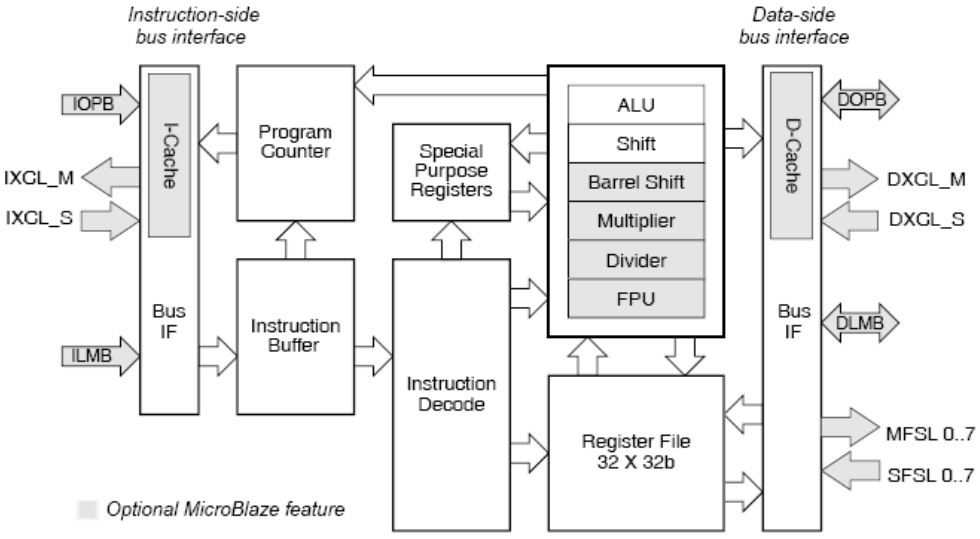
Τα FPGAs είναι ειδικά κατασκευασμένα chips που έχουν τη δυνατότητα επαναπρογραμματισμού ανάλογα με τη λειτουργία που φορτώνεται σε αυτά μέσω κάποιου περιβάλλοντος σχεδίασης. Αποτελούνται από κελιά όπου καθένα από αυτά περιέχει ένα ή περισσότερα LUTs (Lookup Tables) τα οποία παρέχουν τη δυνατότητα να υλοποιούν οποιαδήποτε λογική συνάρτηση και FSM(Finite State Machine).

Αναφορικά με την επιλογή μικροεπεξεργαστή (εξωτερικού - hardcore, εσωτερικού - softcore/hard-wired) στον οποίο θα εκτελείται η σημασιολογική ανάλυση, καθώς και η αρχιτεκτονική επικοινωνίας του μικροεπεξεργαστή με την μονάδα που αναλαμβάνει την συντακτική ανάλυση, έπειτα από αρκετές υλοποιήσεις

προτιμήθηκε η επιλογή του Soft-core microprocessor Microblaze [2.7] λόγω των πλεονεκτημάτων του, που περιγράφονται πιο κάτω.

Ο επεξεργαστής MicroBlaze είναι ένας 32-bit RISC επεξεργαστής soft core, βελτιστοποιημένος για υλοποίηση σε FPGAs της Xilinx. Είναι άκρως παραμετροποιήσιμος, επιτρέποντας στον χρήστη να επιλέγει τα συγκεκριμένα χαρακτηριστικά που είναι αναγκαία για την υλοποίησή του. Τα κυριότερα χαρακτηριστικά του, όπως φαίνεται και στο Σχήμα 4, δίδονται παρακάτω:

- 32 καταχωρητές γενικού σκοπού μήκους 32 bit
- Μήκος εντολής 32 bit με τρεις τελεστές και δύο μεθόδους διευθυνσιοδότησης
- Αριθμητική και λογική μονάδα (ALU)
- Μονάδα ολίσθησης (shift unit)
- 2 δύο επίπεδα διακοπών (interrupts)
- Διάδρομος εντολών (instruction bus) των 32 bit
- Διάδρομος δεδομένων (data bus) των 32 bit
- Χρήση pipeline κατά την εκτέλεση των εντολών



Σχήμα 4. Διάγραμμα του πυρήνα του MicroBlaze

Επιπλέον, στα βασικά χαρακτηριστικά του μπορούν να προστεθούν επιπρόσθετες δυνατότητες όπως :

- Μονάδα κινητής υποδιαστολής (FPU)
- Διαιρετής hardware
- Πολλαπλασιαστής hardware
- Δυνατότητα ολίσθησης πολλών bit σε έναν κύκλο (barrel shift)
- Γρήγορη μνήμη (cache) τόσο για τις εντολές όσο και για τα δεδομένα

Η επικοινωνία μεταξύ MicroBlaze και hardware modules επιτυγχάνεται με διεπαφές FSL. Ο MicroBlaze μπορεί να σχηματιστεί με μέχρι 16 διεπαφές FSL, οκτώ εισόδου και οκτώ εξόδου. Τα κανάλια FSL είναι αποκλειστικές συνδέσεις μιας κατεύθυνσης, σημείου προς σημείο, μεταξύ του MicroBlaze και των περιφερειακών. Το μήκος του διαδρόμου FSL είναι 32 bit. Ένα ξεχωριστό bit υποδηλώνει εάν μία λέξη που αποστέλλεται ή λαμβάνεται, είναι δεδομένο ή εντολή ελέγχου. Κάθε FSL παρέχει μια διεπαφή χαμηλού latency με τον επεξεργαστή. Συνεπώς είναι ιδανικά για την επέκταση των λειτουργιών του επεξεργαστή με χρήση προσαρμοσμένων τμημάτων hardware. Η απόδοση αυτή εξαρτάται από τη συσκευή υλοποίησης. Τα κυριότερα χαρακτηριστικά της διεπαφής FSL είναι:

- Επικοινωνία από σημείο σε σημείο, μονόδρομης κατεύθυνσης
- Υποστήριξη επικοινωνίας δεδομένων και ελέγχου
- Βασισμένη σε FIFO επικοινωνία
- Διαμορφώσιμο μέγεθος δεδομένων
- Λειτουργία στα 600MHz

3.5 Αναφορές

[3.1] "The Theory of Parsing, Translation and Compiling", A. Aho, J. Ullman, volume 1. Prentice-Hall, 1972

[3.2] "An efficient context–free parsing algorithm", J. Earley, Communications of ACM, vol.13, pp. 94-102, 1970.

[3.3] "Recognition of context free languages in time n^3", D. Younger Information and Control, 10:2, 189-208, 1967

[3.4] "An Improved context – free Recognizer", S. Graham, M. Harrison and W. Ruzzo, ACM Transactions on Programming Languages and Systems, vol.2, no. 3, pp. 415-462, July 1980.

[3.5] "Parallel parsing algorithms and VLSI implementation for syntactic pattern recognition", Y. Chiang and K. Fu, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-6, May 1984.

[3.6] "A programmable Pipelined Coprocessor for Parsing Applications" C. Pavlatos, I. Panagopoulos, G, Papakonstantinou, Workshop on Application Specific Processors (WASP), Stockholm, Sept. 2004

[3.7] www.xilinx.com officiale website

[3.8] "Attribute Grammar Paradigms – A high level methodology in language implementation" J. Paaki, ACM Computing Surveys, Vol.27, no.2, (1995)

[3.9] "A High Performance OR-parallel Prolog System", PhD thesis, The Royal Institute of Technology, Stockholm, March 1992.

[3.10] "Parallel execution of prolog programs: a survey", G. Gupta, E. Pontelli, Khayri A. M. Ali, M. Carlsson, Manuel V. Hermenegildo, Journal of Programming Languages and Systems, Vol. 23, No 4, pages 472-602, 2001

[3.11] Communications of the ACM, Volume 36, Issue 3, (March 1993) , 1993, ISSN:0001-0782

[3.12] "Embedded System Design: A Unified Hardware/Software Introduction", Frank Vahid, Tony Givargis, WILEY, 2002

[3.13] "A grammatical view of logic programming", P. Deransart and J. Maluszynski, MIT Press, 1993.

[3.14] "Attribute Grammar Paradigms – A high level methodology in language implementation", J. Paaki, ACM Computing Surveys, vol. 27, no. 3, June 1995

[3.15] "An attribute grammar interpreter as a knowledge engineering tool", G.Papakonstantinou, C. Moraitis and T. Panayiotopoulos, Angewandte Informatik 9/86, pp. 382-388, 1986.

[3.16] "Knowledge Based System Diagnosis, Supervision and Control", Edited By S. G. Tzafestas, Plenum Press, 1989

[3.17] "Definite Clause grammars for language analysis – a survey of the formalism and comparison with augmented transition networks.", Pereira, F.C.N, and Warren, D.H.D. , Journal in Artificial Intelligence, vol. 13, pp 231-278, 1981

[3.18] "Attribute Grammar Theorem Prover", T. Panayiotopoulos, G. Papakonstantinou and G. Stamatopoulos, Information and Software Technology, vol. 30, no. 9, pp: 553-560, November 1988

[3.19] "Knowledge represantation with attribute grammars", G. Papakonstantinou, J. Kontos, The Computer Journal, vlo. 29, no. 3, 241-245, 1986.

[3.20] "The Syntax of Programming Languages-A Survey.", Floyd, R.W. , in IEEE Transactions on Electr. Comp., Vol EC 13, No 4, August 1964.

[3.21] "A full theorem prover under uncertainty.", Panayiotopoulos, T., andPapakonstantinou, G. , Journal of Intelligence and Robotic Systems,vol. 7, pp 139-149,1993

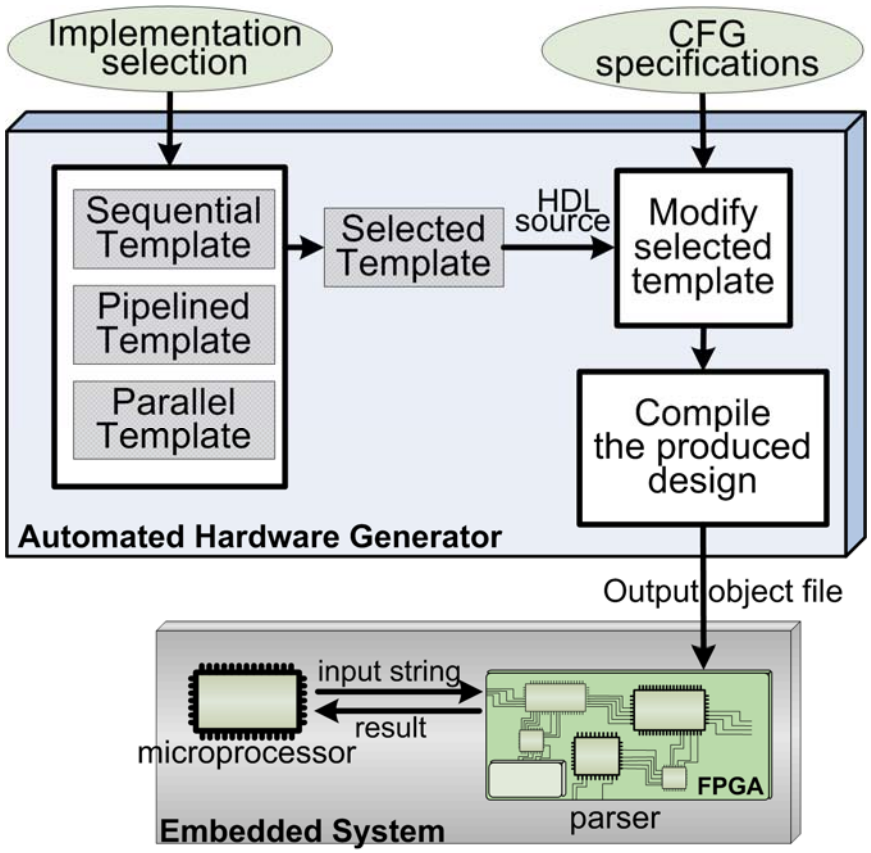
[3.22] "An attribute grammar interpreter for inexact reasoning", Panayiotopoulos, T., Papakonstantinou, G. and Sgouros, N. , Information and Software Technology, vol. 32, no.5, 1989

4. Υλοποιήσεις Αλγορίθμων Συντακτικής Αναγνώρισης

Πρώτος στόχος της διατριβής αυτής είναι η δημιουργία ενός εργαλείου το οποίο θα δέχεται ως είσοδο τις προδιαγραφές μιας CF γραμματικής και θα παράγει αυτόματα την περιγραφή του CF συντακτικού αναλυτή (parser) σε συνθέσιμο πηγαίο κώδικα Verilog-HDL. Αυτό το εργαλείο (βλέπε Σχήμα 5) χρησιμοποιεί τρεις προτεινόμενες αρχιτεκτονικές, σε μια μορφή προτύπων. Το εργαλείο τροποποιεί κατάλληλα την επιλεγμένη από τον χρήστη πρότυπη αρχιτεκτονική, σύμφωνα με τη CFG εισόδου, προκειμένου να κατασκευαστεί ο κατάλληλος hardware parser για τη συγκεκριμένη γραμματική.

Ο χρήστης παρέχει τις προδιαγραφές της γραμματικής και μπορεί να επιλέξει τη αρχιτεκτονική του παραγόμενου hardware parser μεταξύ τριών διαφορετικών αρχιτεκτονικών υλοποίησης : μιας σειριακής, μιας με pipelined τεχνικές, και μιας παράλληλης που θα ενσωματώνει την pipelined υλοποίηση. Και οι τρεις παραπάνω αρχιτεκτονικές είναι πλήρως ανεξάρτητες από τη γραμματική προκειμένου να χρησιμοποιηθούν ως πρότυπα για το αυτοματοποιημένο εργαλείο σύνθεσης των CFG hardware parsers. Σε περίπτωση που η απόδοση είναι η κρίσιμη πτυχή της εφαρμογής και δεν υπάρχει κανένας περιορισμός σχετικά με τον απαιτούμενο αριθμό στοιχείων επεξεργασίας, η παράλληλη αρχιτεκτονική θα πρέπει να προτιμηθεί. Οι άλλες δύο αρχιτεκτονικές (σειριακή και pipelining) επιτυγχάνουν χαμηλότερη απόδοση αλλά απαιτούν σαφώς λιγότερο υλικό προκειμένου να εφαρμοστούν.

Τέλος ο παράλληλος αλγόριθμος συντακτικής ανάλυσης απεικονίστηκε σε ένα προτεινόμενο συνδυαστικό κύκλωμα αυξάνοντας περαιτέρω την απόδοση του συντακτικού αναλυτή.



Σχήμα 5. Αρχιτεκτονική αυτοματοποίησης συντακτικού αναλυτή

4.1 Υλοποίηση του αλγορίθμου του Earley

Το 1970 ο Earley [4.4] παρουσίασε έναν top-down αλγόριθμο συντακτικής ανάλυσης context-free γραμματικών. Ο αλγόριθμος αυτός είναι γενικός υπό την έννοια ότι λειτουργεί για κάθε context-free γραμματική και όχι για περιορισμένο υποσύνολο της κλάσης των γραμματικών αυτών.

Στη γενική περίπτωση ο αλγόριθμος είναι ένας συντακτικός αναγνωριστής ο οποίος αποδέχεται ή απορρίπτει την συμβολοσειρά εισόδου σε χρόνο ανάλογο με το n^3, όπου n το μέγεθος της συμβολοσειράς εισόδου. Για μη διφορούμενες γραμ-

ματικές και για διφορούμενες γραμματικές περιορισμένου βαθμού πετυχαίνουμε αναγνώριση σε χρόνο ανάλογο με το n^2.

Η βασική καινοτομία του αλγόριθμου του Earley είναι η εισαγωγή του συμβόλου "●" το οποίο ονομάζεται τελεία, εφαρμόζεται στους κανόνες της γραμματικής και δεν ανήκει στα σύμβολα αυτής. Η χρησιμότητα της τελείας είναι να χωρίζει το δεξιό μέρος του κανόνα σε δυο σύνολα συμβόλων. Για το σύνολο αριστερά του "●" ισχύει ότι μπορεί να παράγει τη συμβολοσειρά εισόδου που έχει εξεταστεί μέχρι τώρα. Για το σύνολο δεξιά του "●" πρέπει να ελεγχθεί για το αν μπορεί να παράγει το υπόλοιπο της συμβολοσειράς εισόδου.

Οι κανόνες στους οποίους εφαρμόζεται η τελεία ονομάζονται dotted rules. Επίσης το σύμβολο δεξιά της τελείας "●" θα συμβολίζεται με rhs (right hand symbol).

Ο αλγόριθμος διατρέχει τη συμβολοσειρά εισόδου a_1 a_2 a_3… a_n από τα αριστερά προς τα δεξιά . Καθώς κάθε σύμβολο a_i ελέγχεται, κατασκευάζεται ένα σύνολο S_i από καταστάσεις . Μια κατάσταση ορίζεται από την τετράδα {r, j, f, h} όπου :

- το r είναι ο αριθμός του κανόνα παραγωγής .
- το j είναι η θέση της τελείας "●" στο δεξί μέρος του κανόνα .
- το f είναι ένας δείκτης στο σύνολο S_f όπου η κατάσταση δημιουργήθηκε αρχικά.
- το h είναι μια συμβολοσειρά που ονομάζεται k-lookahead string , η οποία ουσιαστικά αποθηκεύει τα k επόμενα σύμβολα του string εισόδου .

Ο αλγόριθμος του Earley κατασκευάζει (n+1) σύνολα (αριθμούνται από 0 μέχρι n) και αρχικοποιείται τοποθετώντας στο αρχικό σύνολο, set 0 , την κατάσταση {0,0,0}. Ο κανόνας με αύξοντα αριθμό 0 είναι αυτός που στο αριστερό του μέρος έχει το αρχικό σύμβολο.

Σε κάθε κατάσταση { r, j, f } , ανάλογα με την μορφή της μπορεί να εφαρμοστεί μια από τις τρεις λειτουργίες Predictor, Scanner, Completer που αποτελούν και τον κορμό του αλγορίθμου. Οι λειτουργίες αυτές και οι συνθήκες εφαρμογής τους περιγράφονται στη συνέχεια .

- Predictor: Η λειτουργία predictor ελέγχει εάν υπάρχει κάποια κατάσταση στο σύνολο S_i, που εξετάζεται, η οποία να έχει ένα μη-τερματικό σύμβολο δεξιά της τελείας και αν υπάρχει προσθέτει στο S_i όλους τους κανόνες που έχουν αυτό το μη-τερματικό σύμβολο στο αριστερό μέρος τους προσθέτοντας πρώτα ένα "●" στην αρχή του δεξιού μέρους του κανόνα.

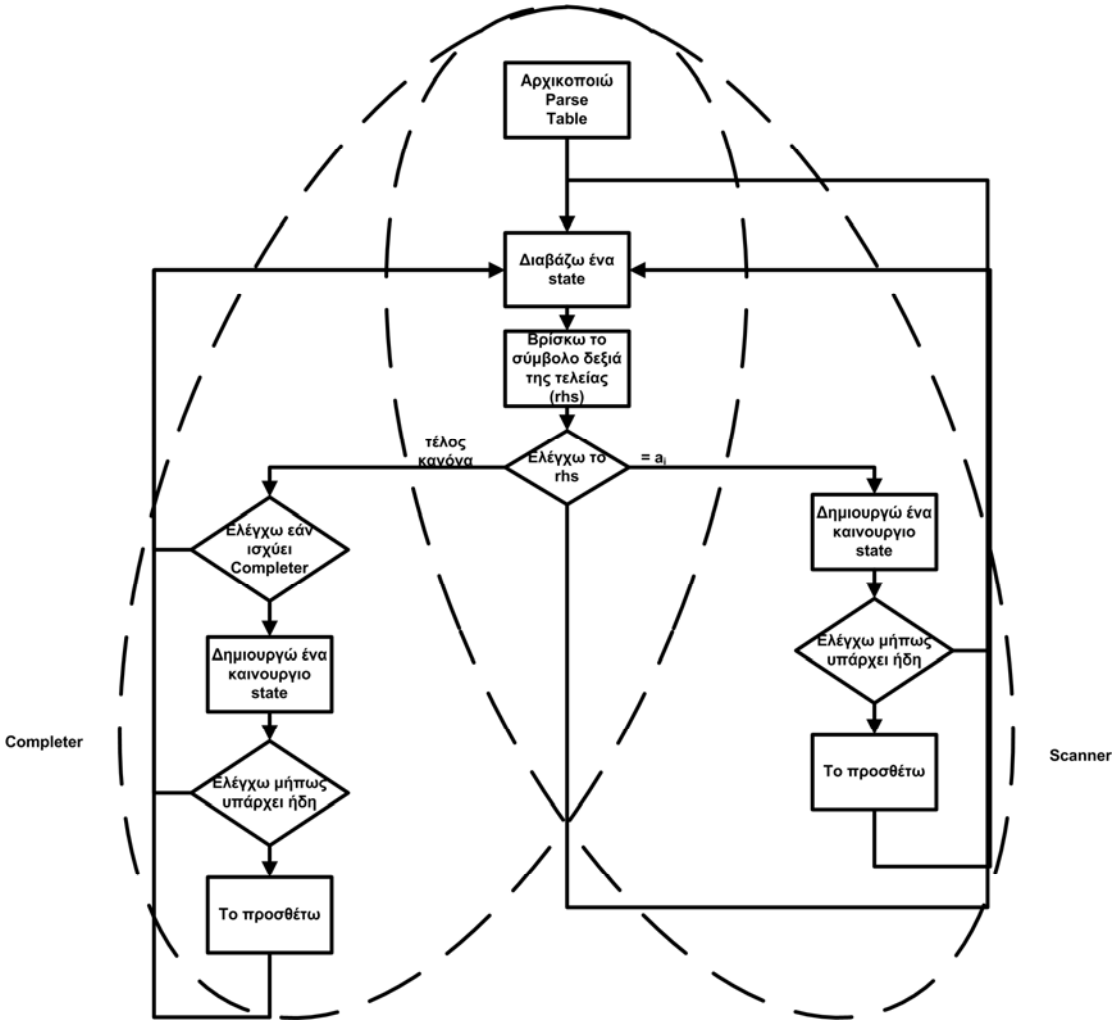
- Scanner: Η λειτουργία scanner ελέγχει εάν υπάρχει κατάσταση στο σύνολο S_i που έχει ένα τερματικό σύμβολο δεξιά της τελείας "●". Εάν ναι τότε συγκρίνει το rhs με το επόμενο σύμβολο της συμβολοσειράς εισόδου , και εάν ταιριάζουν ο scanner προσθέτει αυτόν τον κανόνα στο σύνολο S_{i+1}, αφού πρώτα κινήσει την τελεία μετά το τερματικό σύμβολο.

- Completer: Η λειτουργία completer ελέγχει εάν υπάρχει κάποια κατάσταση στο σύνολο S_i, που εξετάζεται, η οποία να έχει την τελεία στο τέλος του κανόνα. Κατόπιν ο completer συγκρίνει το μη-τερματικό σύμβολο του αριστερού μέρους του κανόνα που εξετάζεται (lhss) , με το σύμβολο αριστερά της τελείας σε κάθε κανόνα που ανήκει στο σύνολο που υποδεικνύεται από το δείκτη f. Οι κανόνες για τους οποίους τα συγκρινόμενα σύμβολα ταιριάζουν προστίθενται στο τρέχον σύνολο S_i , αφού πρώτα η τελεία μετακινηθεί μια θέση δεξιότερα

Το 1980 η S. Graham [4.6] πρότεινε τη χρήση ενός πίνακα PT (Parse Table) αντί της καθορισμένης δομής set του Earley. Το στοιχείο pt(i, j) του πίνακα PT περιέχει όλους τους κανόνες που ανήκουν στο σύνολο S_j και αρχικά δημιουργήθηκαν στο σύνολο S_i. Συγκεκριμένα η στήλη j του πίνακα PT αντιστοιχεί στο σύνολο S_j. Μόνο τα στοιχεία που ανήκουν στην κύρια διαγώνιο και πάνω από αυτή χρησιμοποιούνται. Επιπλέον οι Chiang & Fu [4.5] παρατήρησαν ότι η λειτουργία predictor μπορεί να παραληφθεί προσθέτοντας το σύνολο Predict(N) στα διαγώνια στοιχεία, όπου

Predict (R) = $\{A \rightarrow \gamma \bullet \delta \mid A \rightarrow \gamma\delta \in P, \gamma \xrightarrow{*} \lambda, A \in R\}$ και R είναι ένα σύνολο από μη τερματικά σύμβολα.

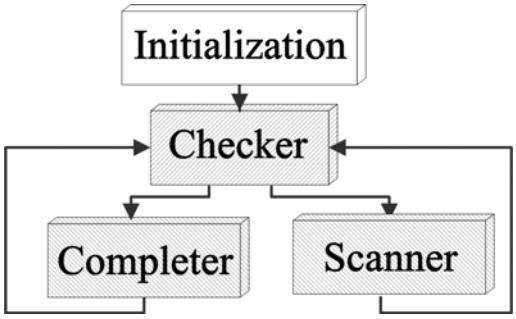
Ο αλγόριθμος χωρίς τη λειτουργία predictor καλείται weakened Earley's algorithm και οι υλοποιήσεις μας είναι βασισμένες σε αυτόν τον αλγόριθμο. Το διάγραμμα ροής του weakened αλγορίθμου Earley παρουσιάζεται στο Σχήμα 6.



Σχήμα 6. Το διάγραμμα ροής του weakened αλγορίθμου Earley

Από αυτό το διάγραμμα ροής, παρατηρούμε ότι η τομή των δύο οβάλ σχημάτων μοιράζεται από τις λειτουργίες του scaner και completer. Επομένως, το εισάγουμε εδώ ως τρίτη θεμελιώδη λειτουργία που καλούμε checker. Με αυτόν τον τρόπο, χωρίζουμε τη διαδικασία συντακτικής ανάλυσης σε τρεις τμήματα, αυτά του checker, completer, scanner όπως παρουσιάζονται απλά στο Σχήμα 7. Θα αναλυθεί αργότερα πώς η προτεινόμενη λειτουργία θα μας επιτρέψει να εφαρμόσουμε τεχνικές pipeline στην υλοποίηση μας.

Αρχικά μια λειτουργία checker εκτελείται και μετά ανάλογα με τη θέση της τέλειας και το σύμβολο δεξιά αυτής στην κατάσταση που εξετάζουμε, ακολουθεί είτε μια λειτουργία scanner είτε μια λειτουργία completer (Σχήμα 7), έως ότου έχουν ελεγχθεί όλα τα state. Τελικά η ύπαρξη μιας κατάστασης της μορφής S→α. στο set S_n ελέγχει εάν input string γίνεται αποδεκτό ή όχι από τη γραμματική.



Σχήμα 7. Οι τρεις λειτουργίες της συντακτικής ανάλυσης

Στην υλοποίηση μας όλα τα σύνολα Si αποθηκεύονται σε τοπική μνήμη ενώ οι τρεις λειτουργίες (checker, scanner και completer) εκτελούνται από τον συντακτικό αναλυτή υλοποιημένο σε υλικό (hardware parser). Ο hardware parser διαβάζει διαδοχικά όλα τα state που είναι αποθηκευμένα στην τοπική μνήμη, τα εξετάζει, και δημιουργεί νέα state εάν αυτό επιβάλλεται από τις λειτουργίες scanner και completer. Όλες οι λειτουργίες (completer, scanner, checker) επαναληπτικά διαβάζουν, εξετάζουν, και γράφουν ένα state σε κάποιο set, συνεπώς η δομή που χρησιμοποιείται για να απεικονιστεί η γραμματική, οι κανόνες και τα state είναι κρίσιμη.

4.1.1 Απεικόνιση των δεδομένων στην μνήμη

Έστω G μια CFG η οποία έχει s σύμβολα (non-terminal και terminals), w συντακτικούς κανόνες με μέγιστο δεξί μέλος (rhss: right hand side symbols) m και n είναι το μήκος του input string. Χρειαζόμαστε ένα πρόσθετο σύμβολο '|' το οποίο δείχνει το τέλος των κανόνων. Ο τρόπος τα σύμβολα, οι κανόνες και τα state απεικονίζονται περιγράφεται παρακάτω.

- κάθε σύμβολο του λεξιλογίου V μπορεί να απεικονιστεί με την χρήση $\lceil \log_2(s+1) \rceil$ bits,
- κάθε κανόνας της γραμματικής μπορεί να αριθμηθεί με την χρήση $\lceil \log_2(w) \rceil$ bits,
- κάθε κανόνας μπορεί να απεικονιστεί με την χρήση ((m + 2) * $\lceil \log_2(s+1) \rceil$) bits,
- κάθε state μπορεί να απεικονιστεί με την χρήση ($\lceil \log_2(w) \rceil + \lceil \log_2(n) \rceil + \lceil \log_2(m+1) \rceil$) bits, όπου ο πρώτος όρος δείχνει τον κανόνα, ο δεύτερος δείχνει το σύνολο που το state ανήκει (S_f) και το τρίτο δείχνει τη θέση της τελείας.

Παράδειγμα 1. Θεωρήστε την CFG G = {V, N, P, S} όπου:
V = {S, E, T, P, +, a, b},
N = {E, T, P} και
P = {S→E, E→E+T, E→T, T→P, P→a, P→b}

Για αυτήν την γραμματική s = 7, w = 6 και m =3 και ας θεωρήσουμε n = 6, τότε χρειαζόμαστε 3-bit bit-vectors ($\lceil \log_2(7+1) \rceil$)για την απεικόνιση των συμβόλων, 3-bit bit-vectors ($\lceil \log_2(6) \rceil$) για να αριθμήσουμε τους κανόνες, 15-bits bit-vectors ((3+2)*$\lceil \log_2(7+1) \rceil$) για να απεικονίσουμε τους κανόνες, τελικά για την απεικόνιση των state, 8-bits bit-vectors (($\lceil \log_2(6) \rceil + \lceil \log_2(6) \rceil + \lceil \log_2(3+1) \rceil$)) χρειάζονται, 3bits προσδιορίζουν τον κανόνα, 3 bits προσδιορίζουν το set που state δημιουργήθηκε (S_f) και 2 bits προσδιορίζουν την θέση της τελείας. Η απεικόνιση των δεδομένων φαίνεται στον Πίνακα 2.

Σ' αυτό το παράδειγμα Predict (N) = {S→.E, E→.E+T, E→.T, T→.P, P→.a, P→.b}. Αυτό το σύνολο προστίθεται σε όλα sets προκειμένου να παραλειφθεί η λειτουργία Predictor. Η μνήμη αρχικοποιείται από τα bit-vectors που απεικονίζουν τους κανόνες, ακολουθούμενα n sets που περιέχουν αρχικά μόνο τα states που προήλθαν από την λειτουργία Predict (N) όπως φαίνεται στον Πίνακα 2..

Το σύνολο από dotted rules Predict(N), προστίθεται σε όλα τα επιμέρους σύνολα προκειμένου να παραλειφθεί η λειτουργία predictor.

Η μνήμη λοιπόν αρχικοποιείται από κάτω προς τα πάνω με w=6 διανύσματα μεγέθους 20 bit, που αντιπροσωπεύουν τους κανόνες. Τα υπόλοιπα σύνολα της μνήμης αρχικοποιούνται με τις καταστάσεις που προκύπτουν από την διαδικασία Predict(N). Η συνεχόμενη εφαρμογή των λειτουργιών (checker, scanner, completer) θα γεμίσουν τα επιμέρους σύνολα και θα μας οδηγήσουν αποδοχή ή απόρριψη της συμβολοσειράς εισόδου.

Ο παρακάτω πίνακας απεικονίζει την κωδικοποίηση της προηγούμενης γραμματικής.

Πίνακας 2. Απεικόνιση των δεδομένων στην μνήμη

Symbols		**No of rule**		**Rule representation**	**Memory**
S	000	000	S→E	000_001_111_111_111	
E	001	001	E→E+T	001_001_100_010_111	**Set n**
T	010	010	E→T	001_010_111_111_111	Predict (N)
P	011	011	T→P	010_011_111_111_111	⋮
+	100	100	P→a	011_101_111_111_111	**Set 1**
A	101	101	P→b	011_101_111_111_111	Predict (N)
B	110	**State representation**			**Set 0**
|	111	E→E+T. created in 3rd set S_2		001_010_011	Predict (N)
					Rules

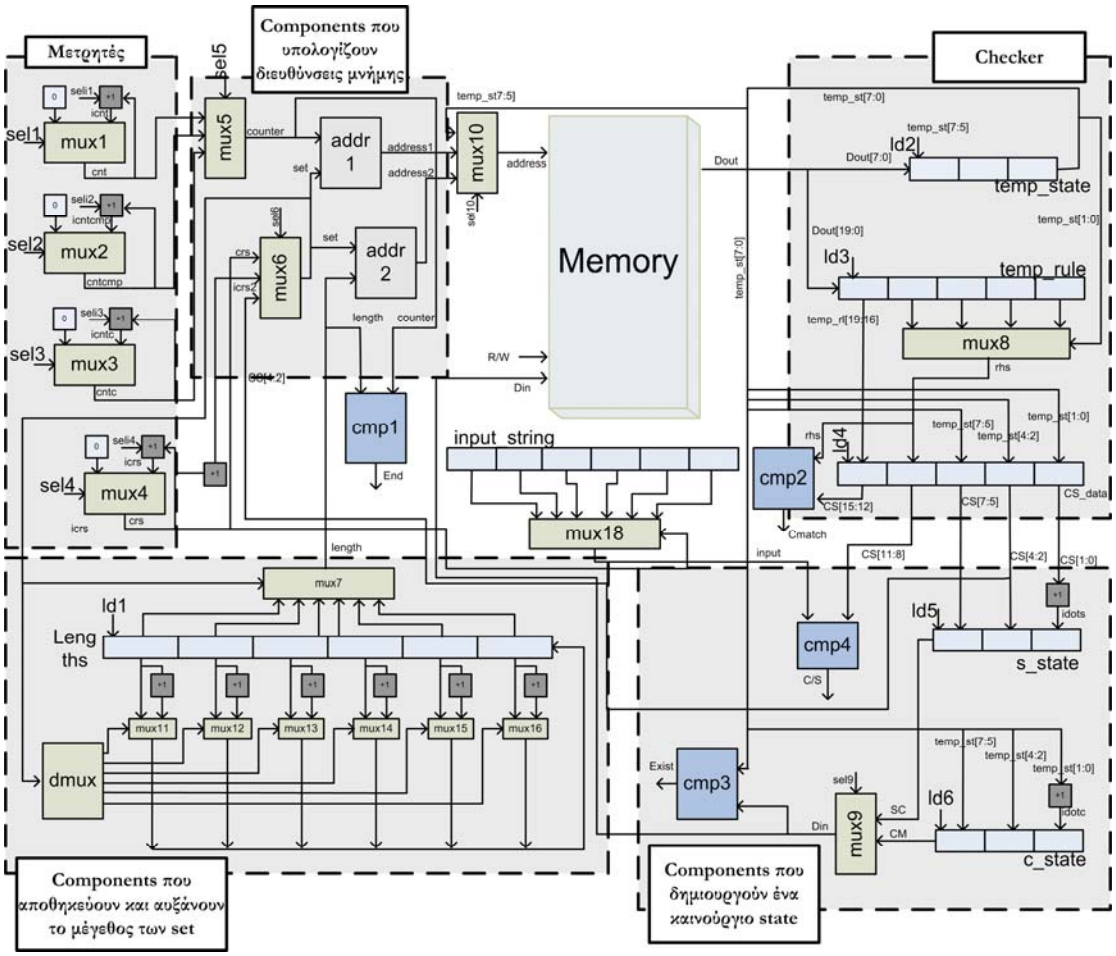
4.1.2 Τεχνικές λεπτομέρειες της υλοποίησης

Όπως έχει αναφερθεί παραπάνω, στην υλοποίηση μας [4.3] όλα τα σύνολα Si αποθηκεύονται σε τοπική μνήμη ενώ οι τρεις λειτουργίες (checker, scanner και completer) εκτελούνται από τον συντακτικό αναλυτή υλοποιημένο σε υλικό (hardware parser). Ο hardware parser διαβάζει διαδοχικά όλα τα state που είναι αποθηκευμένα στην τοπική μνήμη, τα εξετάζει, και δημιουργεί νέα state εάν αυτό επιβάλλεται από τις λειτουργίες scanner και completer.

Στην πραγματικότητα καλούμαστε να σχεδιάσουμε μια πρότυπη υπολογιστική μονάδα, έναν πρότυπο επεξεργαστή. Ο επεξεργαστής αυτός αποτελείται από μια μονάδα ελέγχου (Control Unit) και μια δίοδο δεδομένων (Datapath). Η δίο-

δος δεδομένων αναφέρεται σε κυκλώματα που επιτελούν συγκεκριμένες λειτουργίες όπως διευθυνσιοδότηση, ανάκληση περιεχομένων (καταστάσεων ή κανόνων) από την μνήμη, συγκρίσεις, αποθήκευση δεδομένων χρήσιμα για την εκτέλεση του αλγορίθμου που υλοποιούμε κτλ. Με την μονάδα ελέγχου (Control Unit) καταφέρνουμε να οδηγούμε τα επιμέρους κυκλώματα του Datapath.

Η δομή της διόδου δεδομένων (Datapath) για την σειριακή υλοποίηση του Weakened Earley αλγορίθμου, για την γραμματική G της προηγούμενης παραγράφου, δίνεται στο επόμενο σχήμα :



Σχήμα 8. Αρχιτεκτονική υλοποίησης του weakened αλγορίθμου Earley

Η λειτουργία checker χρησιμοποιεί δύο μετρητές (counters) για να γνωρίζει τη θέση του συνόλου που εφαρμόζεται {cnt} και της κατάστασης που ελέγχει {crs}.

Με την χρήση αυτών των μετρητών υπολογίζεται η διεύθυνση {addr1} της εκάστοτε κατάστασης, που βρίσκεται αποθηκευμένη στην τοπική μνήμη και η οποία προσκομίζεται από τη μνήμη στον καταχωρητή {temp_state}. Στη συνέχεια κάνοντας χρήση bit-διανυσμάτων που δείχνουν τον αύξοντα αριθμό του κανόνα στον οποίο αναφέρεται η κατάσταση του καταχωρητή {temp_state} επιτυγχάνουμε ανάκληση αυτού, από τις κατώτερες θέσεις της μνήμης. Ο κανόνας αυτός αποθηκεύεται στον καταχωρητή {temp_rule} από τον οποίο τελικά λαμβάνουμε το σύμβολο του αριστερού μέρους του κανόνα { lhs } καθώς και το σύμβολο δεξιά της τελείας { rhs }. Αυτά τα δυο σύμβολα μαζί με το διάνυσμα κατάστασης του {temp_state} φορτώνονται στον καταχωρητή {CS_data}, οποίος πλέον περιέχει τα απαραίτητα δεδομένα για αποφασιστεί ποια από τρεις λειτουργίες (checker, scanner, completer) θα ακολουθήσει. Η απόφαση αυτή εξαρτάται από την τιμή του σήματος {C_S}.

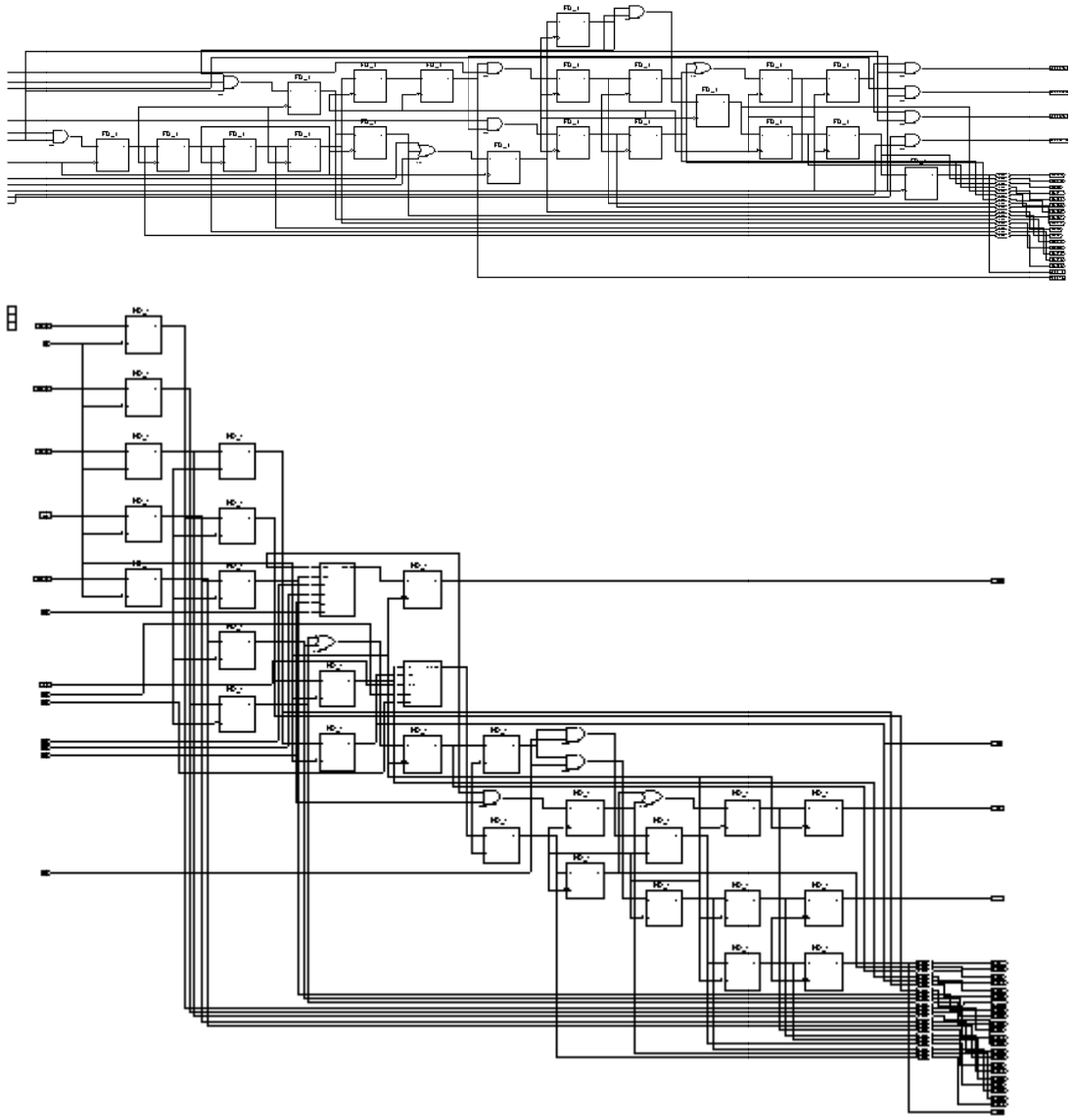
Αν {C_S = 11} τότε η επόμενη λειτουργία που εφαρμόζεται στην κατάσταση του {temp_state} είναι αυτή του scanner. Ο scanner δημιουργεί μια νέα κατάσταση αυξάνοντας κατά ένα το πεδίο που αναφέρεται στη θέση τη τέλειας. Η καινούργια αυτή κατάσταση αποθηκεύεται στον {s_state }. Στη συνέχεια με την χρήση του μετρητή {cntcmp} προσκομίζονται όλες τις καταστάσεις του συνόλου Si+1 και συγκρίνονται με το περιεχόμενο του {s_state }. Εάν η νέα κατάσταση δεν υπάρχει { Exist = 0 } στο σύνολο Si+1, θα γραφτεί στο σύνολο αυτό, αυξάνοντας και το μέγεθος του κατά ένα. Εάν { Exist = 1 } τότε το σύνολο Si+1 μένει ως έχει.

Αν {C_S = 01} τότε η επόμενη λειτουργία που εφαρμόζεται στην κατάσταση του {temp_state} είναι αυτή του completer. Εδώ η κατάσταση του {temp_state} χρησιμοποιείται για να διευκρινιστεί σε ποιόν κανόνα αναφερόμαστε καθώς και το { lhs} αυτού. Κατόπιν με την χρήση του bit-διανύσματος του {CS_data}, που αφορά το σύνολο όπου δημιουργήθηκε η κατάσταση, και ενός μετρητή {cntc} όλες οι καταστάσεις του συνόλου αυτού εξετάζονται εάν έχουν { rhs } ίσο με το { lhs} της εξεταζόμενης κατάστασης. Για κάθε κατάσταση που ικανοποιεί το προηγούμενο, {Cmatch = 1 }, μια νέα κατάσταση δημιουργείται αυξάνοντας κατά ένα το πεδίο που αναφέρεται στη θέση τη τέλειας. Η κατάσταση αυτή αποθηκεύεται στον

καταχωρητή { c_state }. Εάν η νέα κατάσταση δεν υπάρχει στο Si {Exist = 0 } τότε θα γραφεί σε αυτό.

Η μονάδα έλεγχου οδηγεί τα επιμέρους κυκλώματα του Datapath και ουσιαστικά εξασφαλίζει σωστή εκτέλεση του αλγορίθμου της συντακτικής αναγνώρισης.

Στη δομή Datapath υπάρχουν δεκαεννέα σήματα ελέγχου. Τα σήματα ελέγχου {seli} και {selij} κατευθύνουν τους πολυπλέκτες, ενώ τα σήματα {ldk}ελέγχουν τη διαδικασία φόρτωσης στους καταχωρητές. Το σήμα {R/W} αναφέρεται στην λειτουργία της μνήμης. Με R/W=0 διαβάζουμε το περιεχόμενο της μνήμης ενώ με R/W=1 γράφουμε καινούργια δεδομένα στην μνήμη.



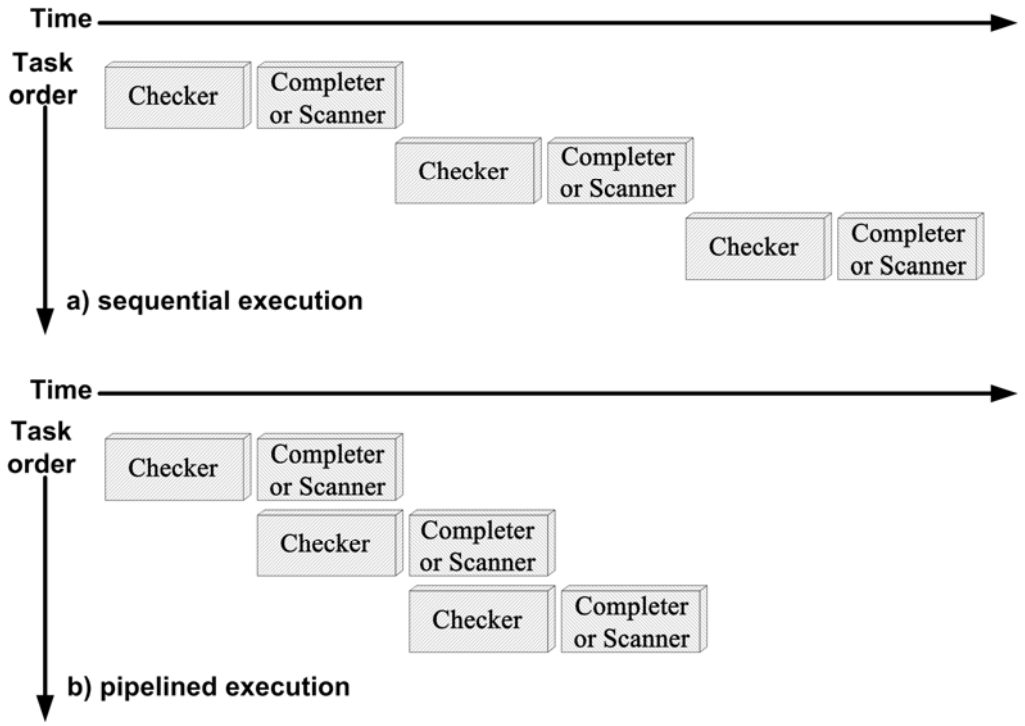
Σχήμα 9. Η σχηματική αναπαράσταση της μονάδας ελέγχου στην ακολουθιακή υλοποίηση

Η μονάδα ελέγχου είναι ουσιαστικά μια μηχανή πεπερασμένων καταστάσεων (Finite State Machine, FSM) που αποτελείται από είκοσι πέντε καταστάσεις και οδηγεί τα σήματα ελέγχου. Σε κάθε αρνητική ακμή του παλμού του ρολογιού το FSM εξετάζει κάποια από τα σήματα C/S, Cmatch, End, Exist που παράγονται από τη δομή Datapath και με βάση την κατάσταση που βρίσκεται εκείνη τη στιγμή οδηγούμαστε στην επόμενη κατάσταση.
Στο Σχήμα 9 απεικονίζεται η εσωτερική δομή του FSM της μονάδας ελέγχου για την περίπτωση της σειριακής υλοποίησης όπως αυτή προκύπτει από το Xilinx Synthesis Tool (XST) :

4.2 Υλοποίηση του αλγορίθμου του Earley με τεχνικές pipelining

Οι τρεις λειτουργίες (checker, scanner, completer) του αλγορίθμου συντακτικής αναγνώρισης μπορούν να εκτελεστούν παράλληλα χρησιμοποιώντας τεχνικές pipelining. Με τις τεχνικές pipelining βελτιώνουμε την απόδοση της υλοποίησης, όσον αφορά το χρόνο απόκρισης.

Στην αρχιτεκτονική που προτείνεται [4.3], τα ζεύγη λειτουργιών ***checker-scanner*** και ***checker-completer*** εκτελούνται παράλληλα και επικαλύπτονται κατά τη διάρκεια του *checker* (Σχήμα 10). Κατά τη διάρκεια της εκτέλεσης ενός *scanner* ή ενός *completer* η επόμενη κατάσταση μπορεί να ελέγχεται από λειτουργία *checker*.



Σχήμα 10. Γραφική απεικόνιση σειριακής& pipelined εκτέλεσης

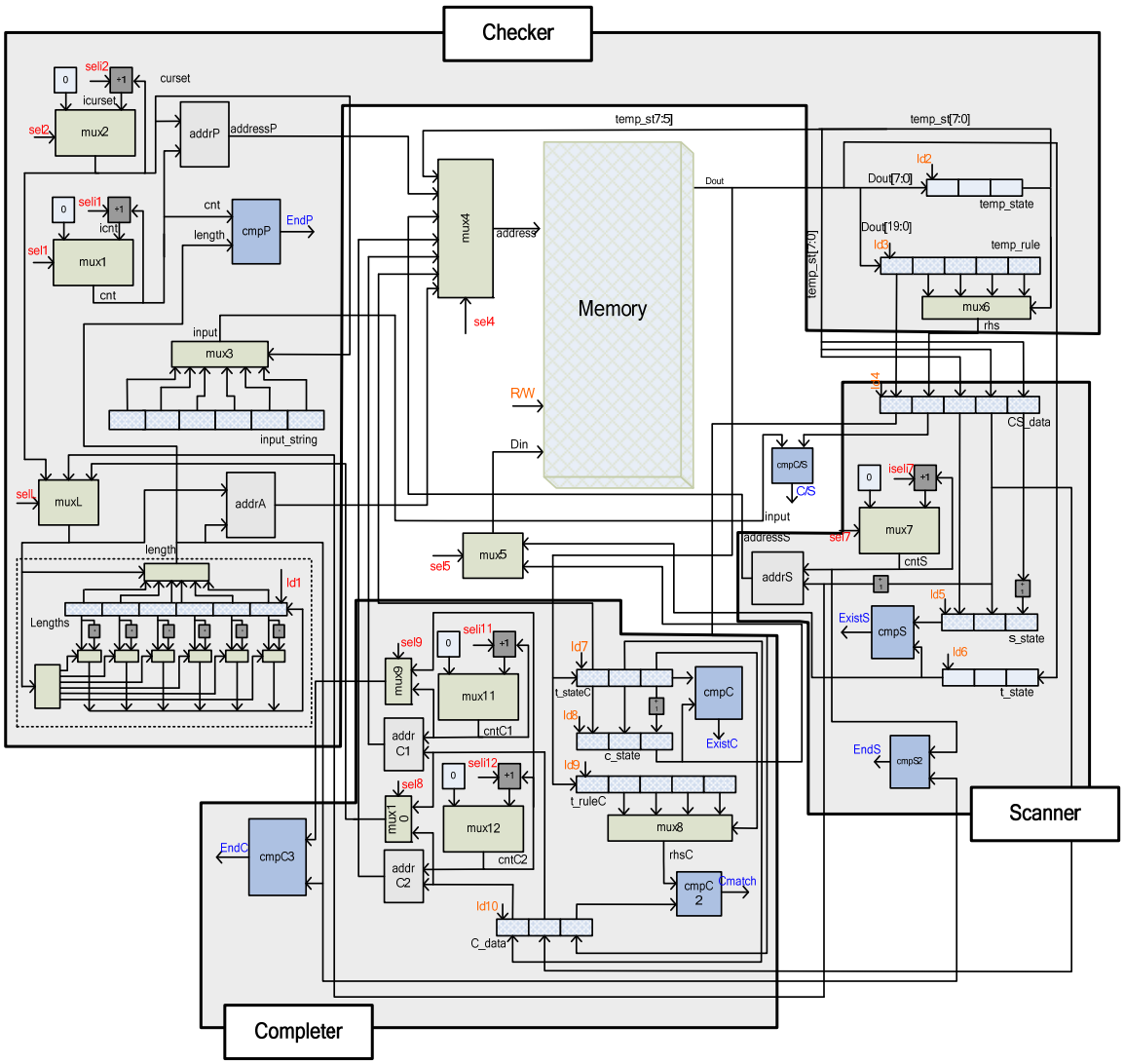
Διατηρώντας το *datapath* της σειριακής υλοποίησης, που παρουσιάστηκε στην προηγουμένη παράγραφο, θα πρέπει να αντιμετωπίσουμε τους "*κίνδυνους δεδομένων*" (*data hazards*) που προκύπτουν λόγω των κοινών πόρων που διατίθενται για παράλληλη εκτέλεση των εργασιών *checker-scanner* και *checker-completer*. Για να αποφευχθούν τέτοιου είδους κίνδυνοι αυξάνουμε τα επιμέρους *components* (*μετρητές, καταχωρητές, πολυπλέκτες, κ.λπ...*) της δομής του *datapath* ώστε οι λειτουργίες να μπορούν να εκτελούνται αυτόνομα, όσον αφορά τους πόρους που απασχολούν.

Θα πρέπει να σημειωθεί ότι η παράλληλη εκτέλεση του checker με κάποιον scanner ή κάποιον completer δεν προκαλεί λογικές ανακολουθίες κατά την εκτέλεση του αλγορίθμου της συντακτικής αναγνώρισης. Αυτό συμβαίνει διότι η σειρά που θα εφαρμοστούν σε κάποιο σύνολο οι scanner και completer δεν έχει σημασία, καθώς οι καταστάσεις του κάθε συνόλου αποτελούν την ένωση των καταστάσεων που υπάρχουν αρχικά και εκείνων των καταστάσεων που προστίθενται από την εφαρμογή των scanner και completer.

Η περίπτωση κάποια κατάσταση να προστεθεί σε κάποιο σύνολο και να μην ελεγχθεί από την λειτουργία checker, αποφεύγεται θέτοντας τον περιορισμό ο

checker να μην προχωρεί στον έλεγχο του επόμενου συνόλου αν πρώτα όλες οι διαδικασίες scanner και completer του προηγούμενου συνόλου δεν έχουν ολοκληρωθεί. Το γεγονός ότι η λειτουργία checker περιμένει το τέλος της εκτέλεσης όλων των scanner και completer μειώνει την επιτάχυνση της pipelined υλοποίησης, παρόλα αυτά ο χρόνος απόκρισης είναι παραμένει μικρότερος από αυτόν της σειριακής. Η νέα δομή *datapath* παρουσιάζεται στο Σχήμα 10. Όπως εύκολα φαίνεται, δεν υπάρχει κανένα component που χρησιμοποιείται σε περισσότερες από μια λειτουργίες. Το μόνο component που είναι μοναδικό και χρησιμοποιείται και από τις τρεις λειτουργίες είναι μνήμη.

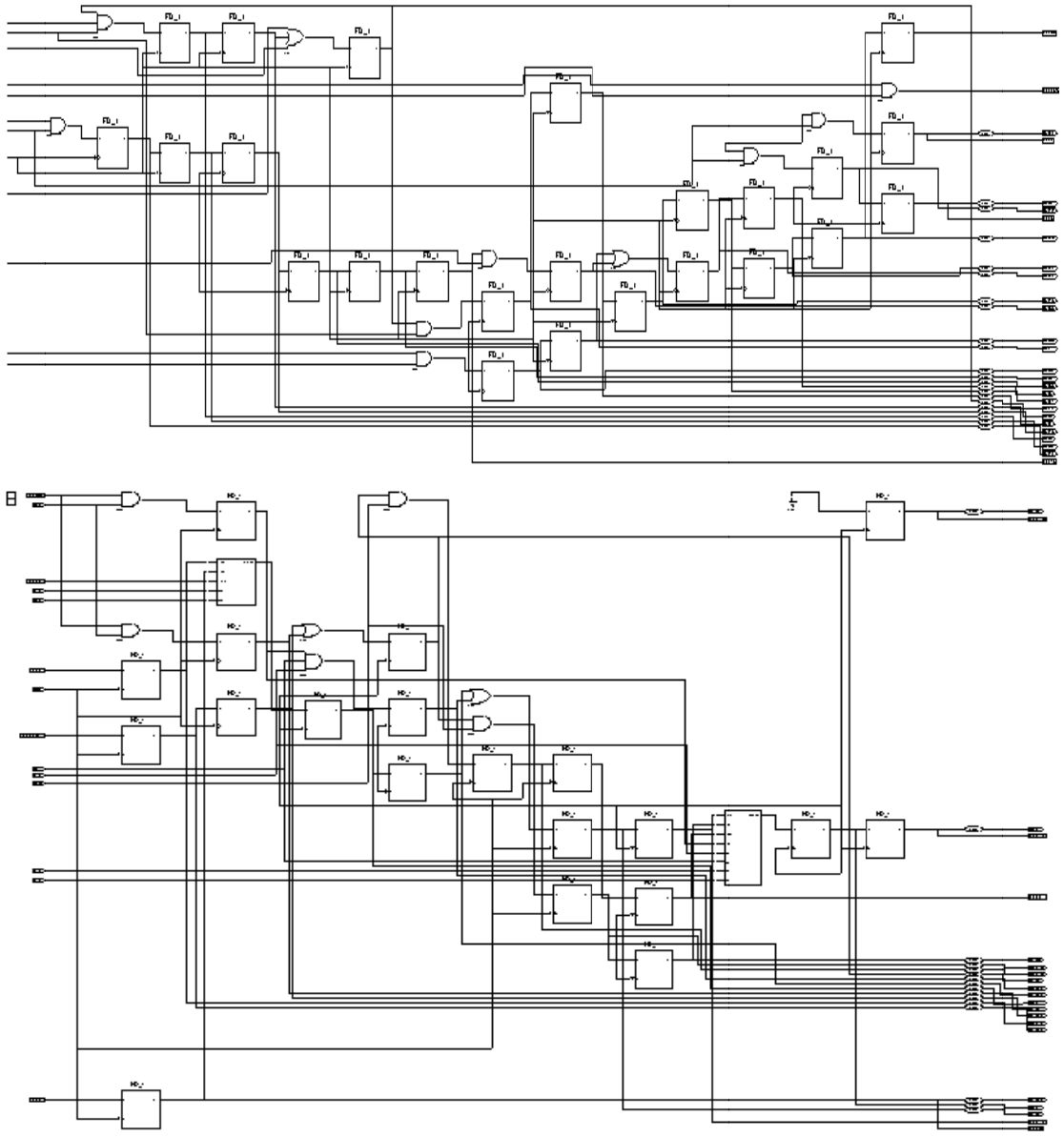
Η μονάδα ελέγχου της pipelined υλοποίησης είναι και αυτή με τη σειρά της μια μηχανή πεπερασμένων καταστάσεων (Finite State Machine, FSM) που αποτελείται από σαράντα-οχτώ καταστάσεις και οδηγεί τα σήματα ελέγχου. Σε κάθε αρνητική ακμή του παλμού του ρολογιού το FSM εξετάζει κάποια από τα σήματα C/S, Cmatch, EndP, EndS, EndC, ExistS, ExistC που παράγονται από τη δομή Datapath και με βάση την κατάσταση που βρίσκεται εκείνη τη στιγμή οδηγούμαστε στην επόμενη κατάσταση.



Σχήμα 11. Darapath pipelined υλοποίησης

Τα σήματα ελέγχου {seli} και {selij}, όπως και στη σειριακή υλοποίηση, κατευθύνουν τους πολυπλέκτες, ενώ τα σήματα {ldk} ελέγχουν τη διαδικασία φόρτωσης στους καταχωρητές. Τα σήματα επίτρεψης {seli} και φόρτωσης {ldk} είναι σε περισσότερα σε αριθμό από αυτά της σειριακής αρχιτεκτονικής, δεδομένου ότι τα components του Datapath έχουν αυξηθεί. Η αύξηση του πλήθους των σημάτων ελέγχου και η παραλληλία στο χρόνο εκτέλεσης που προκύπτει από την ανεξαρτησία των λειτουργιών checker-scanner και checker-completer, αυξάνει όπως είναι φυσικό και την πολυπλοκότητα του FSM που υλοποιεί την μονάδα ελέγχου για την pipelined αρχιτεκτονική.

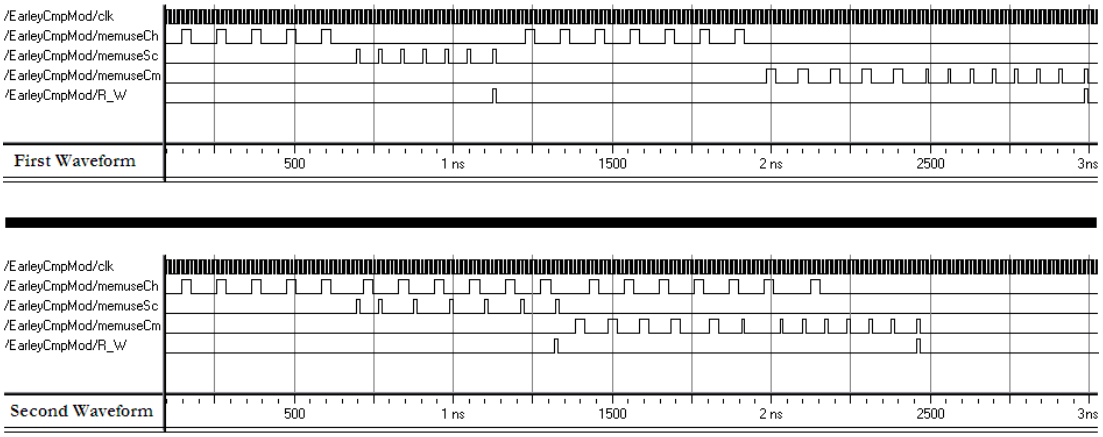
Παρακάτω απεικονίζεται η εσωτερική δομή του FSM της μονάδας ελέγχου για την περίπτωση της pipelined υλοποίησης όπως αυτή προκύπτει από το Xilinx Synthesis Tool (XST) :



Σχήμα 12. Η σχηματική αναπαράσταση της μονάδας ελέγχου στην pipelined υλοποίηση

Ακολουθεί ένα παράδειγμα προσομοίωσης όπου φαίνεται η διαφορετική προσέγγιση των δύο αρχιτεκτονικών σειριακής και pipelined για ότι αφορά τη χρήση

της μνήμης. Δεδομένου ότι η μνήμη είναι το μόνο component που χρησιμοποιείται και από τις τρεις λειτουργίες, ο τρόπος χειρισμού της είναι κρίσιμης σπουδαιότητας. Το σήμα {R/W} αναφέρεται στην λειτουργία της μνήμης. Με R/W=0 διαβάζουμε το περιεχόμενο της μνήμης ενώ με R/W=1 γράφουμε καινούργια δεδομένα στην μνήμη. Τα σήματα memuseCh, memuseSc, memuseCm δείχνουν εάν η μνήμη χρησιμοποιείται από τον checker, scanner, completer. Επιπλέον το σήμα clk (ρολόι) δείχνει τη σχέση χρονισμού των παραπάνω σημάτων. Στο ακόλουθο σχήμα φαίνονται προαναφερόμενες κυματομορφές.



Σχήμα 13. Κυματομορφές των υλοποιήσεων

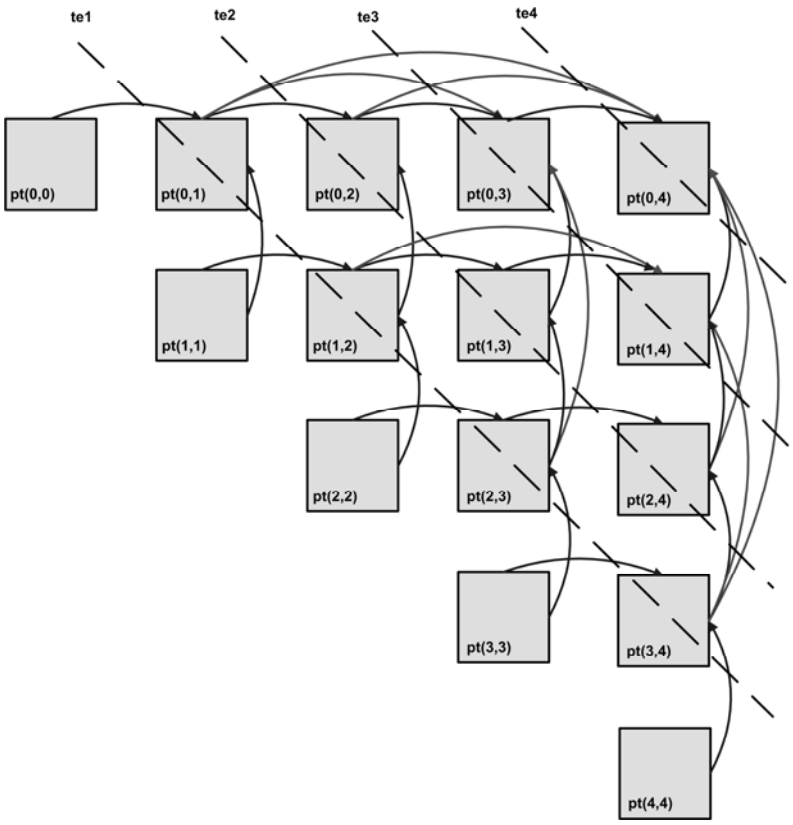
Στην πρώτη κυματομορφή είναι προφανές ότι οι τρεις λειτουργίες εκτελούνται σειριακά. Πρώτα, η λειτουργία του checker εκτελείται, αυτός διαβάζει συνεχώς καταστάσεις από τη μνήμη και ελέγχει εάν σε αυτές εφαρμοστεί η λειτουργία scanner ή completer. Εδώ αποφασίζετε να ακολουθήσει μια λειτουργία scanner. Επομένως ο scanner δημιουργεί τη νέα κατάσταση και αρχίζει η ανάγνωση καταστάσεων από τη μνήμη για να επιβεβαιωθεί ότι η νέα κατάσταση δεν υπάρχει ήδη στο σύνολο που πρόκειται να γραφεί. Στο παράδειγμα μας το σύνολο περιέχει ήδη έξι καταστάσεις οι οποίες ελέγχονται και διαπιστώνεται ότι καμία δεν είναι όμοια με αυτήν που πρόκειται να προστεθεί. Έτσι ο scanner αφού διαβάσει τις έξι ήδη υπάρχουσες καταστάσεις, γράφει τη νέα κατάσταση στη μνήμη (time:1,15ns). Στη συνέχεια μετά τη τέλος του scanner μια λειτουργία checker εκτελείται πάλι. Ανακαλεί από τη μνήμη μια καινούργια κατάσταση και αποφα-

σίζει (time:1,95ns) ότι η λειτουργία completer πρέπει να εφαρμοστεί σε αυτή. Ο completer χρησιμοποιεί τη μνήμη και γράφει τελικά μια νέα κατάσταση στο σύνολο((time: 3ns).

Η δεύτερη κυματομορφή προέρχεται από την pipelined υλοποίηση. Και εδώ συνολικά εκτελούνται δύο λειτουργίες checker, μια λειτουργία scanner και μια λειτουργία completer. Λόγω της παράλληλης εκτέλεσης των λειτουργιών, οι οποίες επικαλύπτονται μεταξύ τους, η όλη διαδικασία έχει τελειώνει 0,525ns νωρίτερα σε σχέση με την σειριακή υλοποίηση. Η λειτουργία checker επικαλύπτεται πάντα από μια λειτουργία scanner ή μια λειτουργία completer, εκτός από την πρώτη εκτέλεση του. Η επιτάχυνση στο συγκεκριμένο παράδειγμα για την pipelined υλοποίηση κυμαίνεται περίπου στο 1,3.

4.3 Υλοποίηση του αλγορίθμου των Chiang & Fu

Οι Chiang και Fu [4.5] παρουσίασαν τον αλγόριθμο του *Earley* με αποδυναμωμένο *predictor* (***weakened Earley algorithm***) και στη συνέχεια παρουσίασαν μια επέκταση αυτού ώστε να μπορεί να εκτελείτε παράλληλα (***Parallel Earley's Recognition Algorithm***). Στην προσέγγιση αυτή προτείνεται η χρήση ενός πίνακα PT (Parse Table) (βλέπε Σχ. 11)αντί των συνδεδεμένων λιστών του Earley . Το κάθε κελί pt(ι,j) του πίνακα PT περιέχει όλους τους dotted rules που ανήκουν στο σύνολο Sj και δημιουργήθηκαν αρχικά στο σύνολο Si , όπου Sj, Si αναφέρονται στα set του αλγορίθμου του Earley . Δεδομένου ότι το κάθε σύνολο μπορεί να περιέχει καταστάσεις που προέρχονται από προηγούμενα σύνολα ή το πολύ το ίδιο σύνολο , διαπιστώνουμε ότι υπάρχει ο εξής περιορισμός ι<j . Αυτό συμβαίνει διότι δεν είναι δυνατό μια κατάσταση να δημιουργηθεί προερχόμενη από σύνολο που δεν έχει εξεταστεί ακόμα . Επομένως διαπιστώνουμε ότι χρησιμοποιούνται μόνο τα κελιά pt(ι,j) με ι<j δηλαδή μόνο αυτά που βρίσκονται επάνω από τη διαγώνιο του πίνακα PT .



Σχήμα 14. Εκτέλεση του παράλληλου Earley αλγορίθμου

Οι λειτουργίες scanner και completer μπορούν να εκφραστούν με τη βοήθεια του τελεστή '⊗' που ορίζεται ως εξής.

Ορισμός 4.1

Αν Q, W σύνολο από dotted rules και R⊆V ορίζουμε :

- $Q \otimes R = \{A \rightarrow \alpha U \beta \bullet \gamma \mid A \rightarrow \alpha \bullet U \beta\gamma \in Q, \text{ and } \beta \xrightarrow{*} \lambda \text{ and } U \in R\}$

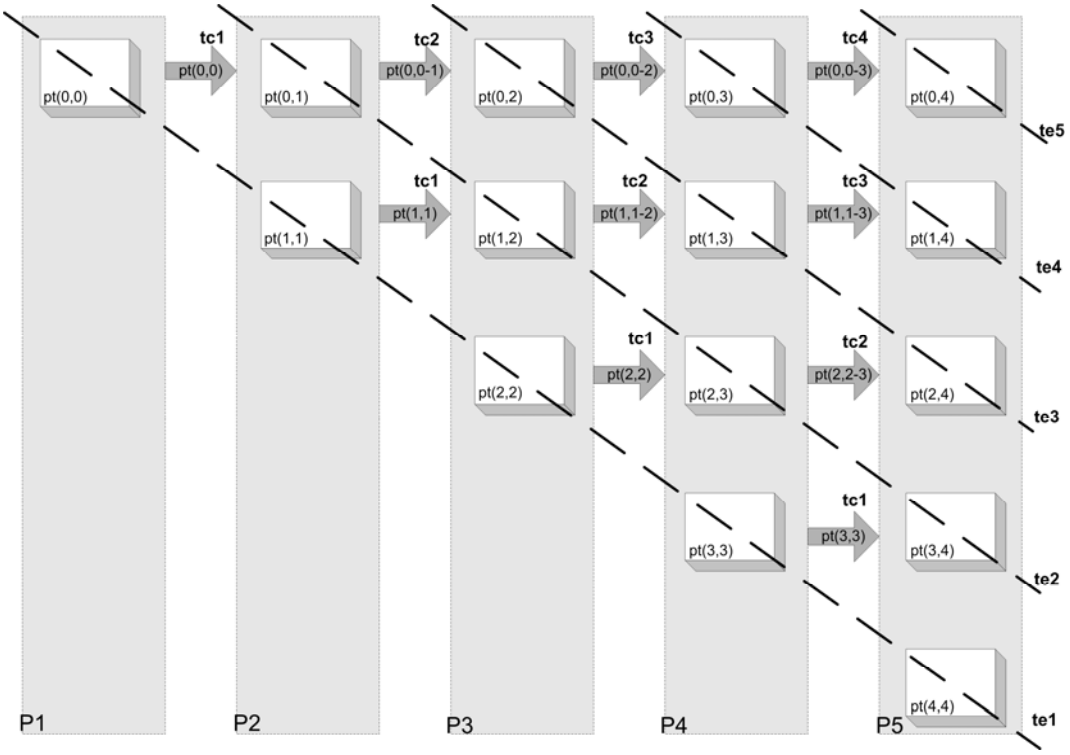
  $\text{and } \{B \rightarrow \delta C \xi \bullet \eta \mid \gamma = \lambda, B \rightarrow \delta \bullet C \xi\eta \in \Upsilon, \xi \xrightarrow{*} \lambda \text{ and } C \xrightarrow{*} A\}$
- $Q \otimes W = \{A \rightarrow \alpha U \beta \bullet \gamma \mid A \rightarrow \alpha \bullet U \beta\gamma \in Q, \beta \xrightarrow{*} \lambda \text{ and } U \rightarrow \delta \bullet \in W\}$

  $\text{and } \{B \rightarrow \delta C \xi \bullet \eta \mid \gamma = \lambda, B \rightarrow \delta \bullet C \xi\eta \in \Upsilon, \xi \xrightarrow{*} \lambda \text{ and } C \xrightarrow{*} A\}$

Το κάθε κελί PT(i,j) ,που στην πραγματικότητα είναι ένα σύνολο από dotted rules, υπολογίζεται σύμφωνα με την ακόλουθη εξίσωση :

$$pt(i,j)=\begin{cases} scanner: & pt(i,j-1)\otimes \mathrm{a}_j & \cup \\ completer: & pt(i,i+1)\otimes pt(i+1,j) & \cup \\ & pt(i,i+2)\otimes pt(i+2,j) & \cup \\ & \cdots & \cup \\ & pt(i,j-1)\otimes pt(j-1,j) & \end{cases}$$

Στηριζόμενοι σε αυτόν τον αλγόριθμο προτείναμε μια καινούργια αρχιτεκτονική που φαίνεται στο Σχήμα 15 και αυξάνει την επιτάχυνση κατά ένα παράγοντα n (το μέγεθος της συμβολοσειράς εισόδου) ενώ χρειάζεται μόνο n επεξεργαστικά στοιχεία και όχι n(n+1)/2 του Σχήματος 14



Σχήμα 15. Η προτεινόμενη παράλληλη αρχιτεκτονική

Ο αριθμός δηλαδή των απαιτούμενων επεξεργαστικών μονάδων μειώθηκε κατά ένα ποσοστό n/2. Κάθε μονάδα επεξεργασίας μπορεί να θεωρηθεί σαν μια στήλη του πίνακα pt και υπολογίζει κελιά (cell) της στήλης αυτής. Οι στήλες ουσιαστικά είναι τα σύνολα Earley [4.4]. Σε κάθε κύκλο εκτέλεσης (t_{e1}, t_{e2},…, t_{en}) η κάθε επεξεργαστής υπολογίζει μόνο ένα κελί, το οποίο μαζί με όλα τα προηγούμενα κελιά της ίδιας γραμμής τα μεταδίδει στην ακριβώς δεξιά του επεξεργαστική μονάδα κατά το κύκλο επικοινωνίας (t_{c1}, t_{c2},…, t_{cn}).

Ένα καινούργιο κελί pt[i,j] υπολογίζεται εφαρμόζοντας τις λειτουργίες scanner και completer στις καταστάσεις των κελιών που είτε έχουν υπολογιστεί κατά προγενέστερους κύκλους εκτέλεσης στην επεξεργαστική μονάδα j είτε έχουν ληφθεί από τον επεξεργαστή j-1. Για το λόγο αυτό για τον υπολογισμό του κάθε pt[i,j] μπορούμε να χρησιμοποιήσουμε την pipelined αρχιτεκτονική που παρουσιάστηκε σε προηγούμενη παράγραφο του κεφαλαίου και εκτελεί τις λειτουργίες checker, scanner και completer.

Η συσχέτιση μεταξύ του τελεστή '⊗' που πρότειναν οι Chiang-Fu[10] και των λειτουργιών checker, scanner και completer της προτεινόμενης αρχιτεκτονικής είναι η εξής:

- Η πράξη pt[i,j] ⊗ $\{a_{j+1}\}$, όπου a_{j+1} ανήκει στο σύνολο NT της γραμματικής, ισοδυναμεί με μια λειτουργία checker στο κελί pt[i,j] την οποία μπορεί να ακολουθήσει η λειτουργία scanner, αν βρεθεί κατάσταση στο pt[i,j] με rhs= a_{j+1}. Η καινούργια κατάσταση που θα προκύψει από τον scanner θα γραφεί στο κελί που έλαβε χώρα η πράξη pt[i,j] ⊗ $\{a_{j+1}\}$.
- Η πράξη pt[i,j] ⊗ pt[k,l], όπου pt[i,j] και pt[k,l] σύνολα από dottted rules, ισοδυναμεί με μια λειτουργία checker στο κελί pt[k,l] την οποία μπορεί να ακολουθήσει η λειτουργία scanner, αν βρεθεί κατάσταση στο pt[i,j] με rhs=|. Το κελί pt[i,j] είναι αυτό στο οποίο θα αναζητήσει ο completer να βρει dottted rules με rhs ίδιο με το lhs του κανόνα που προκάλεσε τον completer. Η καινούργια κατάσταση που θα προκύψει από τον scanner θα γραφεί στο κελί που έλαβε χώρα η πράξη pt[i,j] ⊗ pt[k,l].

Επιπροσθέτως με την προτεινόμενη παράλληλη αρχιτεκτονική, το σχήμα επικοινωνίας μεταξύ των επεξεργαστικών μονάδων βελτιώθηκε αισθητά καθώς δεν απαιτούνται πλέον κάθετοι διάδρομοι (data-bus) για την μετάδοση των δεδομένων, δεδομένου ότι τα κελιά που ανήκουν στην ίδια στήλη υπολογίζονται από τον ίδιον επεξεργαστή. Οι δεδομένα τώρα μεταφέρονται μόνο οριζόντια από τις αριστερότερες προς τα δεξιότερες επεξεργαστικές μονάδες. Ο πρώτος επεξεργαστής δεν λαμβάνει ποτέ καινούργιες καταστάσεις ενώ ο τελευταίος δεν στέλνει ποτέ.

Η αλληλουχία των κύκλων εκτέλεσης και των κύκλων επικοινωνίας φαίνεται στον Πίνακα 3. Η κάθε γραμμή αναφέρεται στην παράλληλη λειτουργία των κελιών του parse table.

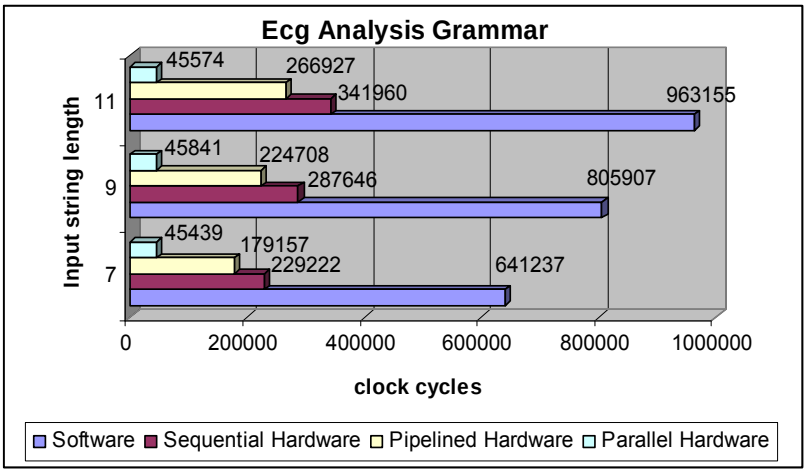
Πίνακας 3. Χρονοδιάγραμμα κύκλων εκτέλεσης και επικοινωνίας

Processor / Task	P_1	P_2	P_3	P_4
computation: 1	pt(0,1)	pt(1,2)	pt(2,3)	pt(3,4)
send: 1	pt(0,1)	pt(1,2)	pt(2,3)	-
receive: 1	-	pt(0,1)	pt(1,2)	pt(2,3)
computation: 2	-	pt(0,2)	pt(1,3)	pt(2,4)
send: 2	-	pt(0,1-2)	pt(1,2-3)	-
receive: 2	-	-	pt(0,1-2)	pt(1,2-3)
computation: 3	-	-	pt(0,3)	pt(1,4)
send: 3	-	-	pt(0,1-3)	-
receive: 3	-	-	-	pt(0,1-3)
computation: 4	-	-	-	pt(0,4)
send: 4	-	-	-	-
receive: 4	-	-	-	-

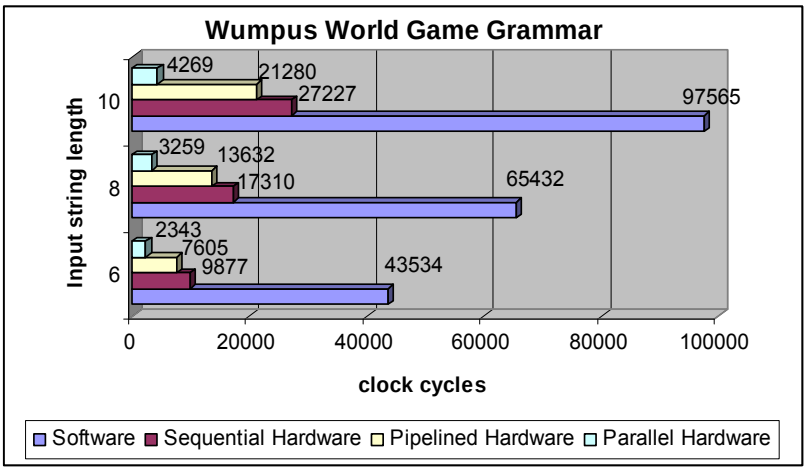
4.4 Πειραματικά Αποτελέσματα

Τα πειραματικά αποτελέσματα των παραπάνω υλοποιήσεων φαίνονται στα σχήματα που ακολουθούν. Οι μετρήσεις έγιναν για δύο γραμματικές, μια από το πεδίο της Ιατρικής και μια από το πεδίο της Τεχνητής Νοημοσύνης. Η πρώτη γραμματική είναι το συντακτικό κομμάτι της γραμματικής που αναγνωρίζει το ΗλεκτροΚαρδιοΓράφημα (ΗΚΓ). Η γραμματική αυτή έχει 64 συντακτικούς κανόνες με μέγιστο μέγεθος 8, τα μη τερματικά σύμβολα είναι 35 και τα τερματικά είναι 6. Συνεπώς s = 41 και w = 64 ενώ m =8. Η δεύτερη γραμματική είναι τα συντακτικό κομμάτι της Κατηγορηματικής Γραμματικής που είναι ισοδύναμη με το λογικό πρόβλημα του παιχνιδιού "Wumpus World". Η γραμματική αυτή έχει 11 συντακτικούς κανόνες με μέγιστο μέγεθος 4, τα μη τερματικά σύμβολα είναι 5 και τα τερματικά είναι 7. Συνεπώς s = 12 και w = 11 ενώ m =4.

Οι μετρήσεις έγιναν για διάφορα μεγέθη συμβολοσειράς εισόδου.



Σχήμα 16. Τα αποτελέσματα της γραμματικής αναγνώρισης του ΗΚΓ

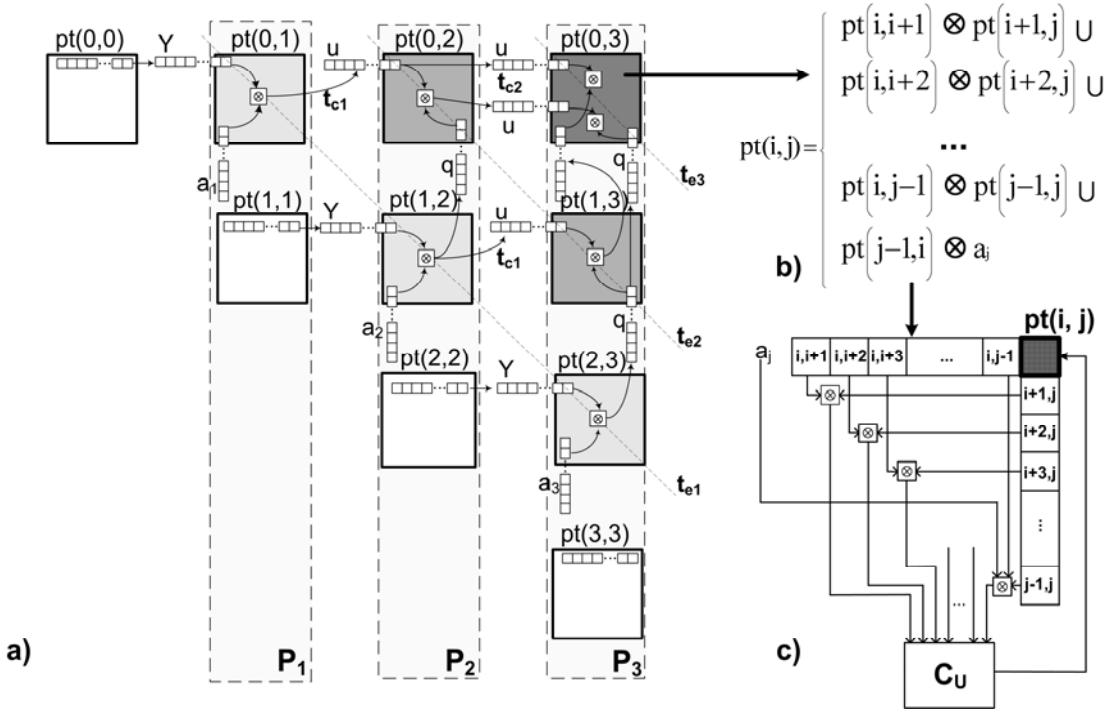


Σχήμα 17. Τα αποτελέσματα της γραμματικής για την εφαρμογή Τεχνητής Νοημοσύνης

Η ακολουθιακή υλοποίηση αυξάνει την απόδοση της συντακτικής αναγνώρισης κατά ένα παράγοντα 5 συγκριτικά με την software υλοποίηση, ενώ η pipelined υλοποίηση επιτυγχάνει μια επιπλέον επιτάχυνση σε ποσοστό σε ποσοστό 28%. Επιπρόσθετα, η παράλληλη αρχιτεκτονική βελτιώνει επιπλέον την απόδοση κατά ένα παράγοντα ανάλογο του μεγέθους της συμβολοσειράς εισόδου, έχοντας σταθερά απόδοση η οποία είναι καλύτερη κατά μια τάξη μεγέθους καλλίτερη από την software υλοποίηση

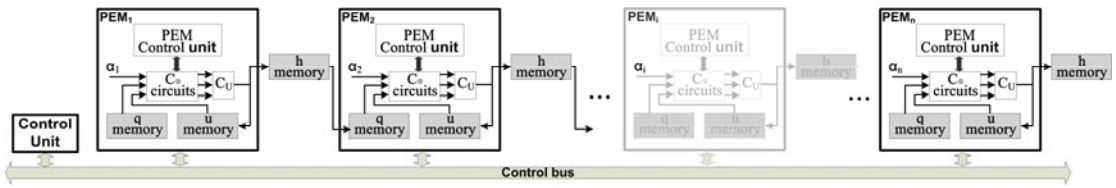
4.5 Υλοποίηση του αλγορίθμου των Chiang & Fu με συνδυαστικό κύκλωμα

Οι παραπάνω υλοποιήσεις επιτυγχάνουν να βελτιστοποιήσουν την απόδοση της Συντακτικής Αναγνώρισης κατά μια τάξη μεγέθους. Δεδομένης της υλοποίησης των αλγορίθμων σε υλικό η επιτάχυνση αυτή δεν είναι ικανοποιητική και αυτό οφείλεται στην πολυπλοκότητα του τελεστή ⊗. Στην προσπάθεια να βελτιστοποιηθεί περεταίρω η υλοποίηση, αποφασίστηκε να υλοποιηθεί ο τελεστής ⊗ με συνδυαστικό κύκλωμα $C_{\otimes}$ [4.1], [4.2] και να υπολογίζεται κάθε κελί pt(i,j) του πίνακα PT, παράλληλα με χρήση πολλών τέτοιων κυκλωμάτων όπως φαίνεται στο παρακάτω Σχήμα 18c. Η ένωση (C_U) των αποτελεσμάτων των κυκλωμάτων αυτών αποτελούν το περιεχόμενο του pt(i,j). Επιτυγχάνοντας έτσι δύο μορφές παραλληλισμού, μια σε επίπεδο αρχιτεκτονικής (υπολογισμός πίνακα) και μια σε επίπεδο υπολογισμού κελιού.



Σχήμα 18. Προτεινόμενη αρχιτεκτονική για την υλοποίηση του αλγορίθμου με χρήση συνδυαστικού κυκλώματος

Η προτεινόμενη εφαρμογή είναι κυρίως βασισμένη στην ύπαρξη ενός βασικού module, Processing Element Module (PEM), που περιγράφεται σε Verilog HDL. Κάθε στοιχείο επεξεργασίας (P_1, P_2, . . .P_n) που παρουσιάζεται στο σχήμα 18a είναι στιγμιότυπο του PEM. Κάθε PEM υπολογίζει σε κάθε βήμα εκτέλεσης (te1, te2, . . . ten) ένα κελί του PT όπως φαίνεται στο Σχήμα 18b. Έτσι ώστε το PEM για να υλοποιήσει την αρχιτεκτονική του Σχήματος 18, ένας απαραίτητος αριθμός C⊗ κυκλωμάτων απαιτούνται, όπως και δύο ουσιαστικές μονάδες μνήμης, q και u. Η πρώτη μονάδα, q, χρησιμοποιείται για την αποθήκευση των στοιχείων που έρχονται από το ίδιο στοιχείο επεξεργασίας, αλλά ανήκουν σε ένα χαμηλότερο κύτταρο τού PT. Η δεύτερη μονάδα, u, χρησιμοποιείται για την αποθήκευση των στοιχείων που προέρχονται από τα κελιά στα αριστερά του τρέχοντος PEM. Σαφώς μια μονάδα ελέγχου PEM (PEM Control Unit) απαιτείται για να καθοδηγήσει τις διαδικασίες που εκτελούνται από το PEM όπως η παραγωγή των κατάλληλων διευθύνσεων των μνημών, προσκομίζοντας στοιχεία κ.λπ.. Σε κάθε βήμα εκτέλεσης τα απαραίτητα στοιχεία, όποιοι προσκομίζονται από τις μνήμες q και u, υποβάλλεται σε επεξεργασία από $C_{\otimes}$ κυκλώματα και η ένωσή τους (CU) (βλέπε Σχήμα 1c) παράγει τα δεδομένα εξόδου. Τα δεδομένα εξόδου, που υπολογίζονται από ένα PEM, αποθηκεύεται σε μια μονάδα μνήμης, h, διασυνδέοντας δύο διαδοχικά PEMs, καθώς επίσης και στη μνήμη u του για τη μελλοντική χρήση από το ίδιο το PEM. Κάθε PEM εκτελεί διαδοχικά τρεις διαδικασίες: το βήμα εκτέλεσης, το βήμα αποστολής και το βήμα λήψης. Μια πρόσθετη εισαγωγή για κάθε μια Pi είναι το αντίστοιχο σύμβολο τ [i] από την συμβολοσειρά εισόδου. Η αρχιτεκτονική αυτή είναι παρουσιάζεται στο Σχήμα 19.



Σχήμα 19. Αρχιτεκτονική για την υλοποίηση του αλγορίθμου με χρήση συνδυαστικού κυκλώματος

Πριν προχωρήσουμε στην ανάλυση του συνδυαστικού κυκλώματος $C_{\otimes}$ κρίνεται αναγκαίο να δοθούν οι παρακάτω ορισμοί

Ορισμός 4.2

Predecessors(A) = {B∈N | $B \xrightarrow{*} A$}, δηλαδή όλα τα σύμβολα που παράγουν το A, συμπεριλαμβανομένου και του A.

Descendants(A) = {B∈N | $A \xrightarrow{*} B$}, δηλαδή όλα τα σύμβολα που παράγει το A, συμπεριλαμβανομένου και του A.

Ορισμός 4.3

Δεδομένου ενός συμβόλου a, και των συνόλων από dotted rules Q and U, τα σύνολα $H_1(a, Q)$ και $H_2(Q, U)$ ορίζονται ως εξής:

$H_1(a, Q) = \{A \rightarrow \alpha a \beta \bullet \gamma \mid A \rightarrow \alpha \bullet a \beta \gamma \in Q \text{ and } \beta \xrightarrow{*} \varepsilon\}$.

$H_2(Q, U) = \{A \rightarrow \alpha E \beta \bullet \gamma \mid A \rightarrow \alpha \bullet E \beta \gamma \in Q, \beta \xrightarrow{*} \varepsilon, \text{ and } E \rightarrow \zeta \bullet \in U\}$.

Ορισμός 4.4

Δεδομένου ενός συνόλου από dotted X το σύνολο $H_Y(X)$ ορίζεται ως εξής:

$H_Y(X) = \{B \rightarrow \delta C \xi \bullet \eta \mid D \rightarrow \kappa \bullet \in X, C \in$ Predecessors(D), $B \rightarrow \delta \bullet C \xi \eta \in$ Predict(N) and $\xi \xrightarrow{*} \varepsilon\}$.

Ορισμός 4.5

Με την χρήση των παραπάνω συνόλων μπορούμε να ορίσουμε τον τελεστή ⊗ ως εξής:

$Q \otimes a = H_1(a, Q) \cup H_Y(H_1(a, Q))$, and

$Q \otimes U = H_2(Q, U) \cup H_Y(H_2(Q, U))$.

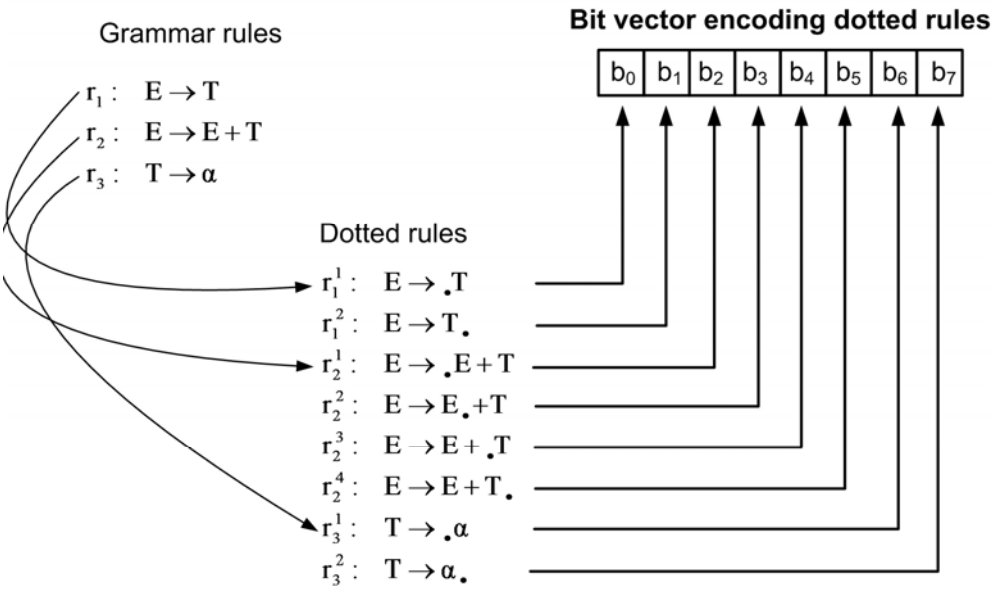
$Q \otimes U \otimes a = H_{12}(Q, U) \cup H_Y(H_{12}(Q, U))$ όπου $H_{12}() = H_1() \cup H_2()$.

Ορισμός 4.6

Κάθε κανόνας r που ανήκει στο σύνολο P συμβολίζεται με r_i, όπου i ($1 \le i \le |P|$) είναι η θέση του κανόνα στο σύνολο αυτό. Προκειμένου να συμβολίσουμε ένα dotted rule, ο ανωτέρω συμβολισμός επεκτείνεται στον r_i^j, όπου j είναι η θέση της τελείας στον κανόνα r_i ($1 \le j \le |r_i|$).

4.5.1 Απεικόνιση των δεδομένων στην μνήμη

Στην παρούσα υλοποίηση προτιμήθηκε μια διαφορετική απεικόνιση των δεδομένων. Κάθε set από dotted rules απεικονίζεται από ένα bit-vector τόσων bits όσοι και οι πιθανοί dotted rules όπως φαίνεται στο Σχήμα 20.



Σχήμα 20. Απεικόνιση δεδομένων για το συνδυαστικό κύκλωμα

4.5.2 Τεχνικές λεπτομέρειες της υλοποίησης

Δεδομένου ότι η υλοποίηση αυτή ακολουθεί τον ορισμό 4.5 έπρεπε να οριστούν οι συναρτήσεις $h_{12}()$ και $h_Y()$ που υπολογίζουν τα διανύσματα $\mathbf{h}_{12}$ και $\mathbf{h}_Y$ που σύμφω-

να με την παραπάνω απεικόνιση αποθηκεύουν τα δεδομένα των συνόλων H_{12}, H_Y του ορισμού 4.5.

Αρχικά, οι δυαδικές εξισώσεις της συνάρτησης $h_{12}()$ θα εξεταστούν. Για κάθε dotted rule $\mathbf{r}_i^j$, που δεν είναι completed, το σύμβολο στα δεξιά της τελείας μπορεί να είναι είτε terminal είτε nonterminal. Εξετάζουμε αυτές τις δύο περιπτώσεις χωριστά:

- Για κάθε dotted rule $\mathbf{r}_i^j$ της μορφής $A \rightarrow \alpha \bullet a\beta$, που ανήκει στο q στη θέση s−1, ο dotted rule $\mathbf{r}_i^{j+1}$ $A \rightarrow \alpha a \bullet \beta$ θα προστεθεί στο $\mathbf{h_{12}}$ στη θέση s, εάν το terminal σύμβολο a είναι ίσος με το τ (z), όπου τ (z) είναι το σύμβολο από το input string για να αναγνωριστεί από το P_z Processing Element. Συμπερασματικά, η συνάρτηση $h_{12}()$ ορίζεται ως:

  $$h_{12}(s) = q(s - 1) \cdot \tau(z) \quad (1)$$

  όπου " · " είναι το λογικό AND. Αυτό σημαίνει ότι εάν το bit q(s − 1) είναι 1 (δήλωση της ύπαρξης $A \rightarrow \alpha \bullet a\beta$) και το bit τ (z) που αντιστοιχεί στο a είναι 1, τότε ο dotted rule $A \rightarrow \alpha a \bullet \beta$ πρέπει να προστεθείτε στο h_{12} (το bit $h_{12}(s) = 1$).

- Για κάθε dotted rule $\mathbf{r}_i^j$ της μορφής $B \rightarrow \gamma \bullet C\delta$, που ανήκει στο q στη θέση s−1, εξετάζουμε τους κανόνες που έχουν το C ως lhss. Έστω αυτοί οι κανόνες είναι οι (rc1) : $C \rightarrow \zeta$, (rc2) : $C \rightarrow \eta$, . . . , (rcl) : $C \rightarrow \rho$. Εάν οι completed dotted rules $C \rightarrow \zeta \bullet$, $C \rightarrow \eta \bullet$, . . ., $C \rightarrow \rho \bullet$ που αντιστοιχούν στην θέση $v_1, v_2, \ldots, v_l$, τότε ο κανόνας $\mathbf{r}_i^{j+1}$ της μορφής $B \rightarrow \gamma C \bullet \delta$ θα πρέπει να προστεθεί στο h_{12} στη θέση s εάν τουλάχιστον ένα από τα bits u(v1), . . . ,u(vl) είναι 1. Συμπερασματικά, η συνάρτηση $h_{12}()$ ορίζεται ως:

  $$h_{12}(s) = q(s - 1) \cdot (u(v1) + \ldots + u(vl)) \quad (2)$$

  όπου "+"είναι το λογικό OR. Αυτό σημαίνει ότι εάν το q(s − 1) είναι 1 (δήλωση της ύπαρξης $B \rightarrow \gamma \bullet C\delta$) και τουλάχιστον ένα από bits u(v1), . . . ,u(vl) (δήλωση της ύπαρξης τουλάχιστον ενός από $C \rightarrow \zeta \bullet$, $C \rightarrow \eta \bullet$ . . ., $C \rightarrow \rho \bullet$) είναι 1 τότε ο dotted rule $B \rightarrow \gamma C \bullet \delta$ πρέπει να προστεθείτε στο $\mathbf{h_{12}}$ (bit h12(s) είναι 1).

Επιπλέον εάν το C παράγει το ε-string, το $h_{12}(s)$ πρέπει να τεθεί στην τιμή του $h_{12}(s-1)$.

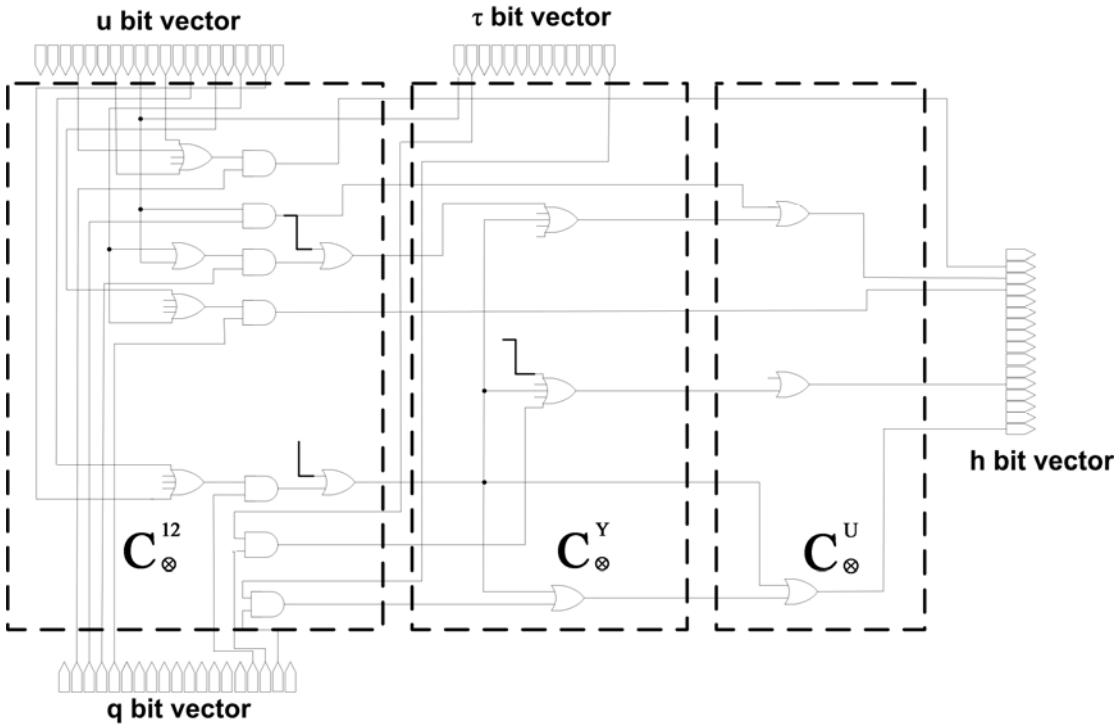
Έπειτα θα εξεταστεί η συνάρτηση $h_Y()$. Ένα dotted rule $\mathbf{r}_i^j$ της μορφής B→δC•ξη προστίθεται στη θέση s του $\mathbf{h}_Y$, εάν οι ακόλουθοι όροι ικανοποιούνται:

(1) Ο dotted rule $\mathbf{r}_i^{j+1}$ της μορφής B→δ•Cξη είναι 1 στο σύνολο p (Predict(N)), δηλαδή $p(s-1) = 1$

(2) Για τουλάχιστον ένα από τα σύμβολα που ανήκουν στο σύνολο Descendant(C) = {D_1,. . . , D_d} υπάρχει ένας completed dotted rule με αυτό το σύμβολο σαν lhss στο $\mathbf{h}_{12}$, δηλαδή, τουλάχιστον ένας Di→κ• ανήκει στο $\mathbf{h}_{12}$, όπου $1 \le i \le d$. Ας υποθέσουμε ότι τα σύμβολα D_1, . . .,D_d είναι lhss symbols σε π completed dotted rules, αντιστοιχία στα bits w_1, . . ., w_π. Τότε $h_Y()$ δίνεται από την ακόλουθη εξίσωση:

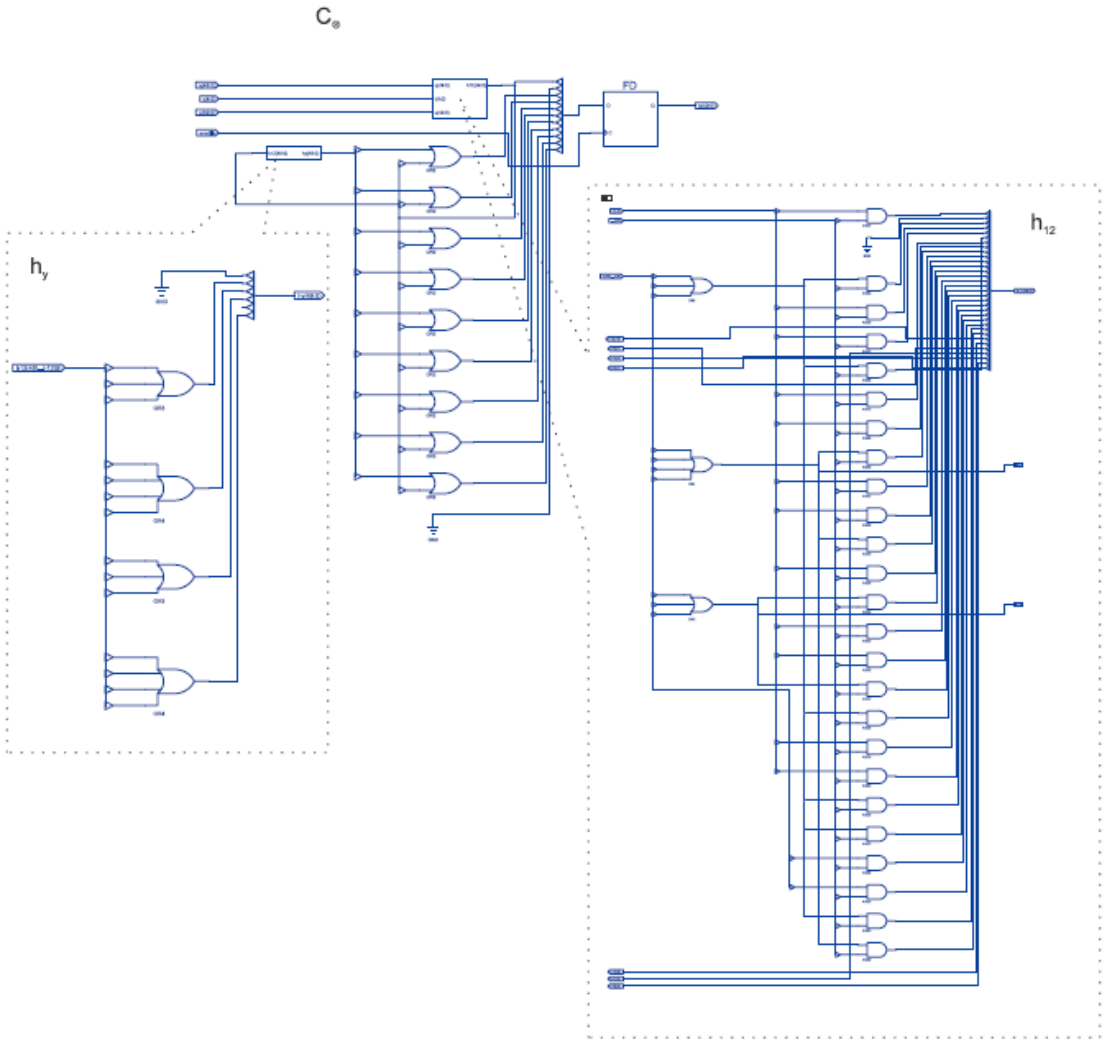
$$hY(s) = p(s-1) \cdot (h12(w1) + \ldots + h12(w\pi)) \quad (3)$$

Αυτό σημαίνει ότι εάν bit $p(s-1)$ είναι 1 (ύπαρξη B→δ•Cξη στο Predict(N)) και τουλάχιστον ένα από τα bits $h_{12}(w_1)$, . . . ,$h_{12}(w_\pi)$ (ύπαρξη του Di→κ•) είναι 1 τότε ο dotted rule B→γC•δ πρέπει να προστεθεί στο $\mathbf{h}_Y$ (bit $h_Y(s)$ γίνεται 1). Επιπλέον, εάν το C παράγει το ε-string, το $h_Y(s)$ πρέπει να τεθεί 1 εάν $hY(s-1) = 1$



Σχήμα 21. Περιληπτική σχηματική απεικόνιση του $C_\otimes$

Μια περιληπτική σχηματική απεικόνιση του $C_{\otimes}$ φαίνεται στο Σχήμα 21, όπου παρουσιάζονται τα τρία βασικά του τμήματα και οι συνδέσεις τους ενώ ένα πλήρες σχηματικό για την Γραμματική αναγνώρισης του ΗλεκτροΚαρδιοΓρφήματος φαίνεται στο σχήμα 22.



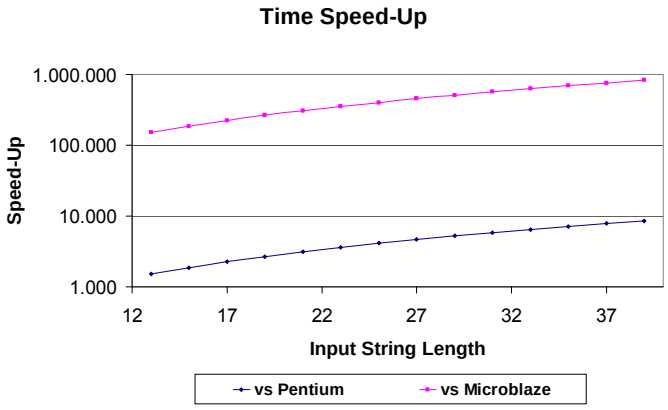
Σχήμα 22. Σχηματικό του $C_{\otimes}$ για την Γραμματική αναγνώρισης του ΗλεκτροΚαρδιοΓρφήματος

4.5.3 Πειραματικά αποτελέσματα

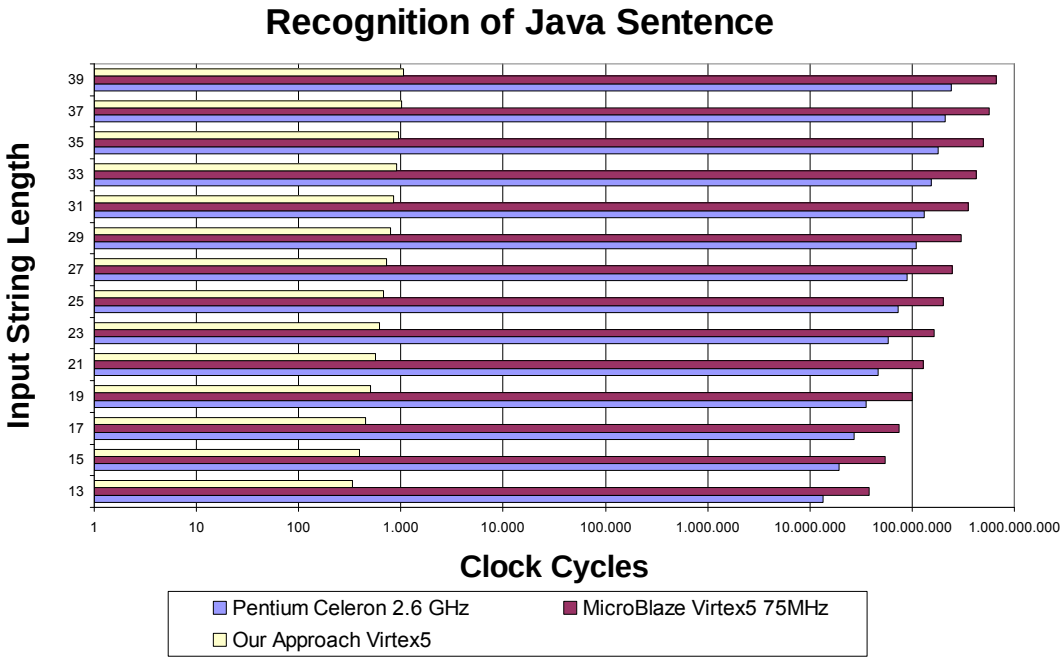
Τα πειραματικά αποτελέσματα της υλοποίησης αυτής είναι πραγματικά εντυπωσιακά. Έγιναν μετρήσεις συγκριτικά με software υλοποιήσεις τόσο σε συμβατικό επεξεργαστή όσο και στον soft-core μικροεπεξεργαστή Microblaze και η επιτάχυνση κυμάνθηκε στις 5 με 6 τάξεις μεγέθους όταν το μέτρο σύγκρισης ήταν οι κύκλοι μηχανής και 3-4 όταν το μέτρο σύγκρισης ήταν ο χρόνος. Τα αποτελέσματα φαίνονται στον πίνακα 4 και στα σχήματα 23 και 24.

Πίνακας 4. Πειραματικά αποτελέσματα υλοποίησης του αλγορίθμου με χρήση συνδυαστικού κυκλώματος

Input String Length	**Software 2,66 Ghz celeron**		**Hardware 100Mhz**		**MicroBlaze 75 Mhz**	
	cyles	**time**	**cycles**	**time**	**cycles**	**time**
13	13.525.073	5,085E-03	334	3,342E-06	38.433.054	5,124E-01
15	19.444.973	7,310E-03	391	3,905E-06	54.860.287	7,315E-01
17	26.824.933	1,008E-02	447	4,469E-06	75.162.510	1,002E+00
19	35.736.302	1,343E-02	503	5,032E-06	99.717.261	1,330E+00
21	46.458.946	1,747E-02	560	5,595E-06	128.903.662	1,719E+00
23	59.010.655	2,218E-02	616	6,159E-06	163.102.419	2,175E+00
25	73.520.863	2,764E-02	672	6,722E-06	202.695.822	2,703E+00
27	90.203.243	3,391E-02	729	7,285E-06	248.067.745	3,308E+00
29	109.161.208	4,104E-02	785	7,849E-06	299.603.646	3,995E+00
31	130.649.765	4,912E-02	841	8,412E-06	357.690.567	4,769E+00
33	154.695.761	5,816E-02	898	8,975E-06	422.717.134	5,636E+00
35	181.439.394	6,821E-02	954	9,539E-06	495.073.557	6,601E+00
37	211.321.186	7,944E-02	1010	1,010E-05	575.151.630	7,669E+00
39	243.870.849	9,168E-02	1067	1,067E-05	663.344.731	8,845E+00



Σχήμα 23. Γραφική απεικόνιση της επιτάχυνσης



Σχήμα 24. Γραφική απεικόνιση των απαιτούμενων κύκλων εκτέλεσης

4.6 Αναφορές

[4.1] "Optimal Reconfigurable Embedded Parsers", C. Pavlatos, Alexandros Dimopoulos, Andrew Koulouris, Theodore Andronikos, Ioannis Panagopoulos and George Papakonstantinou, Computer Languages, Systems & Structures, DOI link: http://dx.doi.org/10.1016/j.cl.2007.08.001

[4.2] "Efficient Signal Processing Using Syntactic Pattern Recognition Methods", Andrew Koulouris, Theodore Andronikos, Christos Pavlatos, Alexandros Dimopoulos, Ioannis Panagopoulos and George Papakonstantinou, SIP 2007, Honolulu USA,Aug. 2007

[4.3] "A programmable Pipelined Coprocessor for Parsing Applications" C. Pavlatos, I. Panagopoulos, G, Papakonstantinou, Workshop on Application Specific Processors (WASP), Stockholm, Sept. 2004

[4.4] "An efficient context–free parsing algorithm", J. Earley, Communications of ACM, vol.13, pp. 94-102, 1970.

[4.5] "Parallel parsing algorithms and VLSI implementation for syntactic pattern recognition", Y. Chiang and K. Fu, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-6, May 1984.

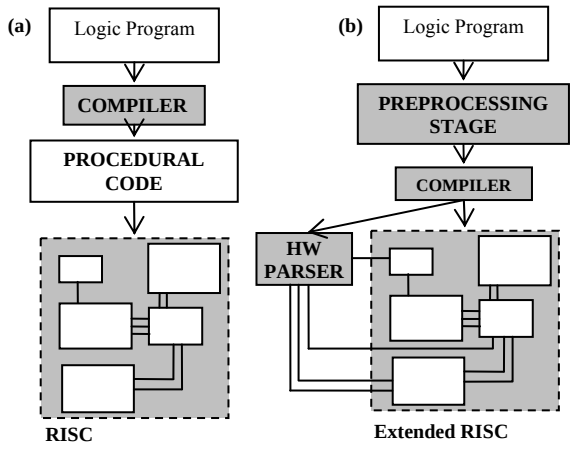
5. Υλοποιήσεις Αλγορίθμων Αποτίμησης Κατηγορηματικών Γραμματικών

Ο δεύτερος στόχος της διατριβής αυτής ήταν η ανάπτυξη αλγορίθμου αποτίμησης των κατηγορημάτων (σημασιολογική ανάλυση) που θα εκτελείται παράλληλα με την συντακτική ανάλυση σε μικροεπεξεργαστή που θα επικοινωνεί με το hardware module του συντακτικού αναλυτή. Συνεπώς οι προτεινόμενοι αλγόριθμοι συντακτικής ανάλυσης πρέπει να βελτιωθούν και επεκταθούν για να καλύψουν και τη σημασιολογία της γραμματικής

5.1 Υλοποίηση του αλγορίθμου του Floyd

Αρχικά μελετήθηκε ο αλγόριθμος του Floyd [5.1] αφού η δομή του επέτρεπε την εύκολη επέκταση του σε Αποτιμητή Κατηγορημάτων γεγονός που μας επέτρεπε να τον χρησιμοποιήσουμε σαν αφετηρία για την υλοποίηση της όλης διατριβής.
Συμβατικές προσεγγίσεις στην υλοποίηση αποτιμητών κατηγορηματικών γραμματικών σε λογισμικό ακολουθούν τα στάδια επεξεργασίας που παρουσιάζονται στο Σχήμα 25(α). Μια εξειδικευμένη γλώσσα προγραμματισμού χρησιμοποιείται, η οποία μπορεί να αποδώσει εκφραστικά όλες τις δομές που απαντώνται στις κατηγορηματικών γραμματικές (συντακτικοί κανόνες, κανόνες αποτίμησης κατηγορημάτων). Κατόπιν, χρησιμοποιείται ένας μεταγλωττιστής ο οποίος εκτελεί τη μετάβαση από τον αρχικό δηλωτικό αλγόριθμο στον αντίστοιχο διαδικαστικό. Ο τελικός διαδικαστικός αλγόριθμος απεικονίζεται σε κάποιον επεξεργαστή. Η προτεινόμενη μεθοδολογία [5.2] [5.3] (Σχήμα 25(β)) ξεκινάει από την έκφραση του αποτιμητή σε μία γλώσσα προγραμματισμού που αποτελεί προέκταση της C και την οποία ονομάζουμε C-AG. Παρουσιάζουμε επίσης, μία προέκταση του μι-

κροεπεξεργαστή RISC η οποία αποτελείται από έναν προγραμματιζόμενο συντακτικό αναλυτή για τον ορισμό των συντακτικών κανόνων, ο οποίος αναλαμβάνει τη δημιουργία του συντακτικού δέντρου, και έναν επεξεργαστή RISC ο οποίος αναλαμβάνει την αποτίμηση των κατηγορημάτων.

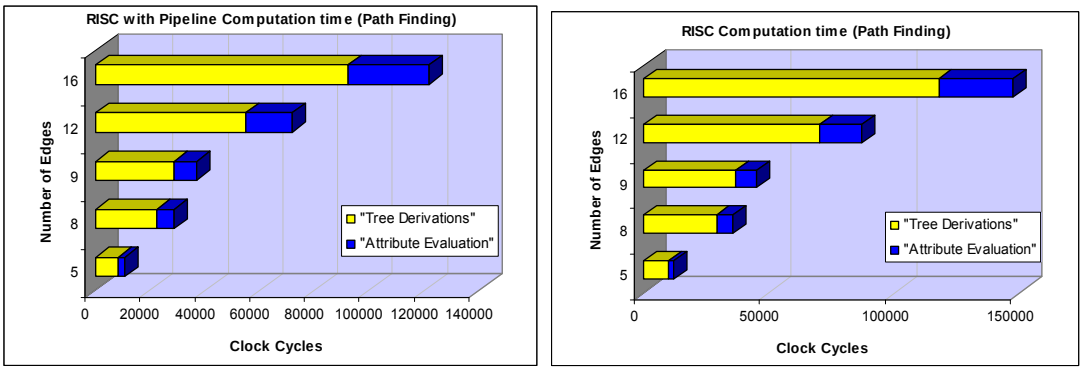


Σχήμα 25. (α) Συμβατικές προσεγγίσεις στην δημιουργία αποτιμητών κατηγορηματικών γραμματικών (β) Προτεινόμενη μεθοδολογία

Η γενική ιδέα που διέπει την προτεινόμενη σχεδίαση βασίζεται στον εμπλουτισμό του αρχικού επεξεργαστή με δύο τρόπους λειτουργίας: έναν διαδικαστικό και έναν δηλωτικό. Στον διαδικαστικό τρόπο, ο επεξεργαστής λειτουργεί κατά το συμβατικό τρόπο (η προέκταση του συντακτικού αναλυτή είναι ανενεργή) και η εκτέλεση του προγράμματος είναι ακολουθιακή μέσω διαδοχικών αυξήσεων του μετρητή προγράμματος ή αλμάτων από εντολές jump. Στο δηλωτικό τρόπο, ο συντακτικός αναλυτής είναι ενεργός και αναλαμβάνει την παραγωγή του συντακτικού δέντρου. Οι κόμβοι που δημιουργούνται αποθηκεύονται σε μία εξειδικευμένη λίστα στην υλικό του αναλυτή. Υπάρχει μία «ένα προς ένα» αντιστοιχία μεταξύ των κόμβων του συντακτικού δέντρου και των γραμμών της στοίβας στην οποία είναι αποθηκευμένοι. Συνεπώς, η θέση του κόμβου στη στοίβα αποτελεί έναν διακριτικό αριθμό για αυτόν (NID-Node IDentification number). Σε κάθε γραμμή της στοίβας αποθηκεύεται επίσης μία κωδικοποίηση του μη τερματικού συμβόλου που αντιστοιχεί σε αυτόν τον κόμβο, μαζί με πληροφορίες για τις εξαρτήσεις αυτού του κόμβου με άλλους κόμβους του δέντρου (που αναφέρο-

νται ως θέσεις στη στοίβα). Σε κάθε δημιουργία/προσπέλαση ενός κόμβου, πρέπει να εκτελεστούν οι κανόνες αποτίμησης των κατηγορημάτων για τον υπολογισμό της τιμής των κατηγορημάτων και την απόφαση για τον αν η σημασιολογική ανάλυση είναι επιτυχής. Οι κανόνες αυτοί είναι οργανωμένοι σε μπλοκ μνήμης στον επεξεργαστή. Κάθε μπλοκ συσχετίζεται με ένα αντίστοιχο μη τερματικό σύμβολο. Ο συντακτικός αναλυτής χρησιμοποιεί το αντίστοιχο μη τερματικό σύμβολο που αντιστοιχεί στον τρέχοντα κόμβο και είναι αποθηκευμένο στην αντίστοιχη γραμμή της στοίβας ώστε να αποφασίσει ποιο μπλοκ κώδικα πρέπει να εκτελεστεί (ελέγχοντας την τιμή του μετρητή προγράμματος). Ένας χώρος δεσμεύεται επίσης στην μνήμη του επεξεργαστή για την αποθήκευση των τιμών των κατηγορημάτων. Τα τελευταία οργανώνονται στη μνήμη δυναμικά σε μπλοκ και συσχετίζονται με κάθε μία από τις γραμμές της εξειδικευμένης στοίβας. Κατόπιν μπορούν να προσπελαστούν βάσει του NID του κάθε κόμβου. Αν απαιτείται η παρουσία συμβολοσειράς εισόδου, και αυτή αποθηκεύεται στη μνήμη του επεξεργαστή, διότι όπως προαναφέρθηκε, η συντακτική ανάλυση πραγματοποιείται με συγκρίσεις των συμβόλων της συμβολοσειράς, με τις τιμές των κατηγορημάτων στα φύλλα του δέντρου. (κατηγορηματικές γραμματικές οριστικών προτάσεων). Η επιλογή αποθήκευσης της συμβολοσειράς εισόδου και των συμβόλων της συντακτικής ανάλυσης σε κατηγορήματα στη μνήμη έγινε εκούσια, από τη στιγμή που τα σύμβολα αυτά συνήθως αναφέρονται σε ολόκληρες κατηγορίες δεδομένων (όπως ακέραιοι, πραγματικοί αριθμοί κ.α.) και μπορούν να είναι συμβολοσειρές οποιουδήποτε μεγέθους. Η αποθήκευση των συμβόλων αυτών στο υλικό του συντακτικού αναλυτή θα οδηγούσε σε δραματική αύξηση του μεγέθους του και θα δυσχέραινε τον επαναπρογραμματισμό της όλης σχεδίασης. Χρησιμοποιώντας την παρούσα μεθοδολογία, πέραν του γεγονότος ότι ο συγκερασμός του δηλωτικού-ακολουθιακού προγραμματισμού μειώνει σημαντικά την πολυπλοκότητα προγραμματισμού εφαρμογών, επηρεάζει επίσης θετικά την απόδοση του τελικού συστήματος.(περίπου κατά 1000%). Αυτό οφείλεται κυρίως στο γεγονός ότι η υλοποίηση σε υλικό του συντακτικού αναλυτή απελευθερώνει τον επεξεργαστή από την ανάγκη παραγωγής του συντακτικού δέντρου. Στο Σχήμα 26, γίνεται φανερό ότι ο υπολογιστικός χρόνος που α-

παιτείται για την δημιουργία των συντακτικών δέντρων από έναν επεξεργαστή RISC και από έναν RISC με pipeline αποτελεί περίπου το 80% του συνολικού χρόνου που απαιτείται για την αποτίμηση των κατηγορημάτων (σε μέσο φόρτο απαιτούμενων πράξεων για την αποτίμησή τους). Συνεπώς, η απεικόνιση του αλγορίθμου παραγωγής του συντακτικού δέντρου σε υλικό βελτιώνει σημαντικά τη συνολική απόδοση του Ε.Σ.



Σχήμα 26. Ο υπολογιστικός χρόνος που απαιτείται για την εκτέλεση εφαρμογής αποτιμητή κατηγορηματικών γραμματικών σε επεξεργαστή RISC χωρίς ή με pipeline

5.2 Ακολουθιακός Αλγόριθμος

Αφού υλοποιήθηκαν σε υλικό και συγκρίθηκαν οι πιο γνωστοί αλγόριθμοι συντακτικής ανάλυσης, δεύτερος στόχος της διατριβής αυτής είναι η ανάπτυξη αλγορίθμου αποτίμησης των κατηγορημάτων (σημασιολογική ανάλυση) που θα εκτελείται παράλληλα με την συντακτική ανάλυση στον μικροεπεξεργαστή που θα επικοινωνεί με το hardware module του parser.

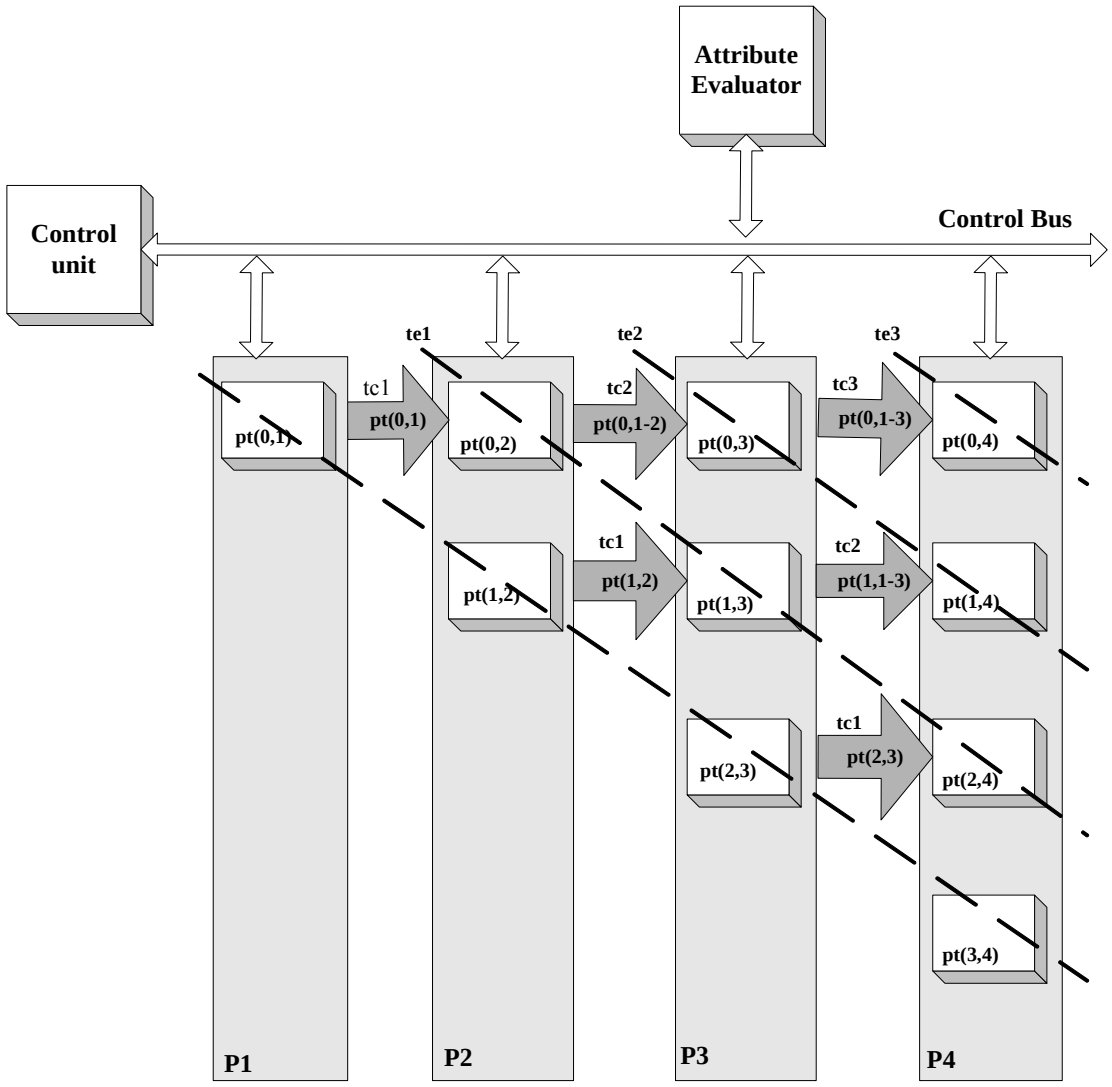
Η πρώτη προσέγγιση [5.6] σε αυτή την προσπάθεια έγινε τροποποιώντας τον σειριακό αλγόριθμο του Earley [5.4]. Πολύ απλά, κάθε φορά που εκτελούταν τελεστής Predictor υπολογίζονταν τα κληρονομούμενα κατηγορήματα, ενώ κάθε φορά που εκτελούταν τελεστής Completer υπολογίζονταν τα συντιθέμενα κατηγορήματα, προτείνοντας έτσι τον παρακάτω αλγόριθμο. Ο αλγόριθμος αυτός μπορεί να εφαρμοστεί σε γραμματικές χωρίς αριστερή αναδρομή. Στα λογικά προγράμματα που στον τελεστή conjunction "$\wedge$" ισχύει η αντιμεταθετική ικανότητα μπορούμε να τροποποιήσουμε κατάλληλα το πρόγραμμα, έτσι ώστε στην

παραγόμενη Κατηγορηματική Γραμματική να μην υπάρχει αριστερή αναδρομή και να μπορεί να εφαρμοστεί ο αλγόριθμος.

```
add state <0,0,0> to S0
for (;;)
{
Predictor{Earley's predictor
              evaluate inherited Attributes   }
Completer{Earley's completer
              evaluate synthesized Attributes }
Scanner  {if(there is correct terminal symbol to the right
           of the dot)
          then execute the respective semantic rules
          if (flag =1)
          add this dotted rule to the next set after
              moving the dot over the dummy terminal symbol}
 if state <0,2,0> is created success
 if new states are not created return failure
```

5.3 Παράλληλος Αλγόριθμος

Στηριζόμενοι στην παραπάνω ιδέα, επεκτάθηκε [5.7] η αρχιτεκτονική του Σχήματος 15 προσθέτοντας ένα μικροεπεξεργαστή που επικοινωνεί με το hardware module που έχει αναλάβει την συντακτική ανάλυση όπως φαίνεται στο Σχήμα 27. Στον μικροεπεξεργαστή εκτελείται ένας αλγόριθμος (Attribute evaluator) που σταδιακά λαμβάνει μια τις στήλες του πίνακα και υπολογίζει τα κατηγορήματα. Μετά από κάθε βήμα εκτέλεσης του παράλληλου αλγορίθμου συντακτικής ανάλυσης, όποιο Processing Element έχει ολοκληρώσει τον υπολογισμό της στήλης του στέλνει στον μικροεπεξεργαστή τα δεδομένα αυτά. Ο μικροεπεξεργαστής στο επόμενο βήμα εκτέλεσης θα υπολογίσει τα κατηγορήματα για τα state που έχει λάβει, ενώ η διαδικασία συντακτικής ανάλυσης συνεχίζεται. Η χρονοδρομολόγηση αυτή φαίνεται στον πίνακα 5, ενώ τα διαγράμματα ροής, τόσο για τα Processing Elements, όσο και για τον Attribute Evaluator φαίνονται στο Σχήμα 29.

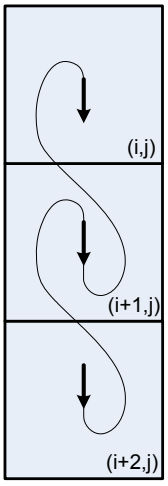


Σχήμα 27. Η αρχιτεκτονικής αποτίμησης κατηγορημάτων με τον παράλληλο συντακτικό αναλυτή

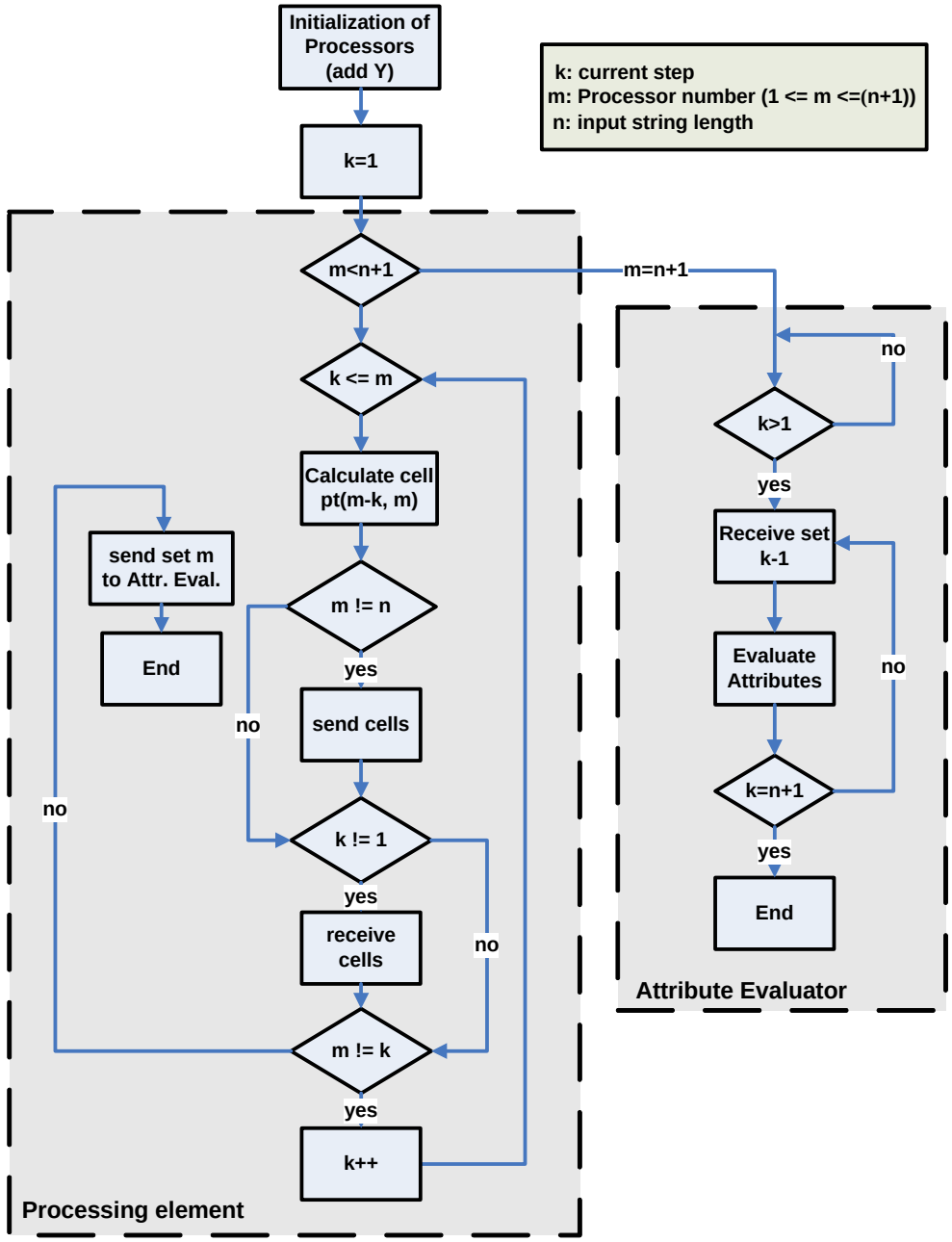
Δεδομένου ότι στον Attribute Evaluator τα δεδομένα φτάνουν κατά στήλες έχει αποδειχτεί ότι η σωστή σειρά να αποτιμηθούν τα κατηγορήματα είναι αυτή του Σχήματος 28. Δηλαδή από επάνω προς τα κάτω για να διατρέξουμε το περιεχόμενο κάθε κελιού του πίνακα αλλά από κάτω προς τα επάνω για να διατρέξουμε τα κελία μιας στήλης του πίνακα .

Πίνακας 5. Πίνακας χρονοδρομολόγησης εργασιών

Processor / Task	P_1	P_2	P_3	P_4	Attribute evaluator
computation: k = 1	pt(0,1)	pt(1,2)	pt(2,3)	pt(3,4)	-
send states: k = 1	pt(0,1)	pt(1,2)	pt(2,3)	-	-
receive states: k = 1	-	pt(0,1)	pt(1,2)	pt(2,3)	-
attribute eval.: k =1	-	-	-	-	-
computation: k = 2	-	pt(0,2)	pt(1,3)	pt(2,4)	-
send states: k = 2	pt(0,1)	pt(0,1-2)	pt(1,2-3)	-	-
receive states: k = 2	-	-	pt(0,1-2)	pt(1,2-3)	pt(0,1)
attribute eval. k = 2					pt(0,1)
computation: k = 3	-	-	pt(0,3)	pt(1,4)	-
send states: k = 3	-	pt(0-1, 2)	pt(0,1-3)	-	-
receive states: k = 3	-	-	-	pt(0,1-3)	pt(0-1,2)
attribute eval. k = 3					pt(0-1,2)
computation: k = 4	-	-	-	pt(0,4)	-
send states: k = 4	-	-	pt(0-2,3)	-	-
receive states: k = 4	-	-	-	-	pt(0-2,3)
attribute eval. k = 4					pt(0-2,3)
computation: k = 5	-	-	-	-	-
send states: k = 5	-	-	-	pt(0,4)	-
receive states: k = 5	-	-	-	-	pt(0,4)
attribute eval. k =5	-	-	-	-	pt(0,4)



Σχήμα 28. Διαδρομή διάσχισης της στήλης του πίνακα συντακτικής ανάλυσης

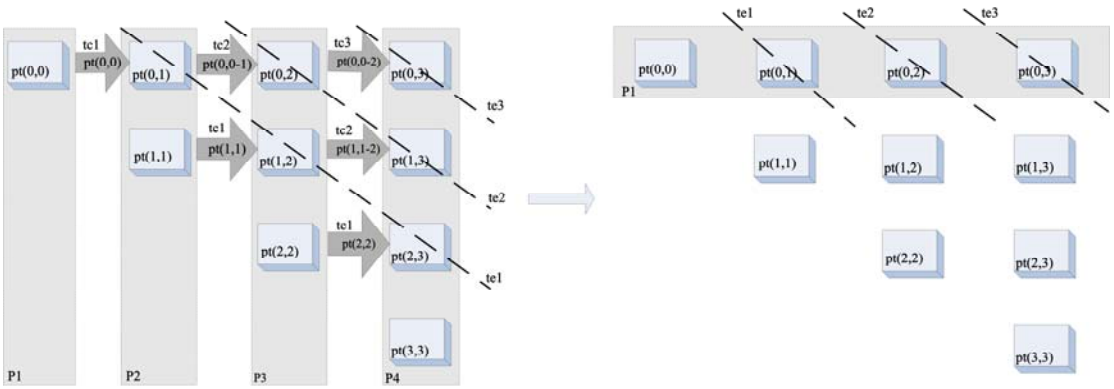


Σχήμα 29. Διάγραμμα ροής για Processing Element και Attribute Evaluator

5.4 Αλγόριθμος για Λογικά Προγράμματα

Η γενική ιδέα της χρησιμοποίησης μιας AG για αναπαράσταση γνώσης (knowledge representation) είναι να χρήση μόνο ενός terminal symbol, του NULL. Κατά συνέπεια, η γραμματική αναγνωρίζει μόνο κενά strings από χαρακτήρες. Προκειμένου να καταστεί η γραμματική συμβατή με τον επιλεγμένο συντακτικό αναλυτή, εισάγουμε την χρήση ενός dummy terminal symbol "d". Συνεπώς, ο συντακτικός αναλυτής αναγνωρίζει inputs strings της μορφής "dd…d|.".

Αφού a_i=d για 1≤i≤n, τα κύτταρα που εκτελούνται κατά τη διάρκεια του βήματος εκτέλεσης t_{e1}, είναι ίσα με $pt(i,j) = pt(i,j-1) \otimes d$. Εντούτοις, τα κύτταρα που ανήκουν στην κύρια διαγώνιο είναι τα ίδια syntax-wise. Επομένως, όλα τα κύτταρα που εκτελούνται κατά τη διάρκεια του βήματος εκτέλεσης t_{e1} δηλαδή τα $pt(i,j),\ 1 \le i<n$, j=i+1, είναι τα ίδια. Επαγωγικά, βασιζόμενοι σε αυτό το κρίσιμο σχόλιο και λόγω της μορφής του τελεστή $\otimes$, μπορεί εύκολα να αποδειχθεί ότι όλα τα κελιά pt(i,j) που ανήκουν στην ίδια διαγώνιο περιέχουν τα ίδια state. Πρέπει να διευκρινιστεί ότι αν και τα κελιά μπορούν να έχουν τα ίδια states, οι τιμές των κατηγορημάτων είναι σαφώς διαφορετικές, αφού τα κατηγορήματα συνδέονται αυστηρά με τη θέση τους στο parse table και στις τιμές των κατηγορημάτων των predecessor και successor συμβόλων. Κατά συνέπεια, η συντακτική ανάλυση task μπορέστε να ολοκληρωθείτε από τη χρήση ενός επεξεργαστικού στοιχείου, αντί για n, που υπολογίζει μόνο τα κελιά της πρώτης σειράς του PT, όπως φαίνεται στο Σχήμα 30 [5.8]. Μόλις υπολογιστεί ένα κελί, αντιγράφεται στα υπόλοιπα της ίδιας διαγώνιου έτσι ώστε να γεμίσει ό πίνακας και να προχωράει η αποτίμηση των κατηγορημάτων. Παραδείγματος χάριν το pt(0,1) θα αντιγραφεί στα pt(1,2) και pt(2,3). Το κόστος για αυτήν την μεταφορά είναι αμελητέο αναλογικά με το κόστος της όλης διαδικασίας. Η συντακτική ανάλυση γίνεται με την αρχιτεκτονική που παρουσιάστηκε στην ενότητα 4.3, ενώ η σημασιολογική με τον αλγόριθμο της προηγούμενης ενότητας.



Σχήμα 30. Βελτιστοποίηση του συντακτικού αναλυτή σε εφαρμογές λογικών προγραμμάτων

5.5 Αναφορές

[5.1] "The Syntax of Programming Languages-A Survey", R.W. Floyd, IEEE Transactions on Electr. Comp., Vol EC 13, No 4, August 1964

[5.2] "An Embedded System for Intelligent Control", I. Panagopoulos, C. Pavlatos and G. Papakonstantinou, Journal of Intelligent and Robotics Systems,vol.42,p.179-211,2005

[5.3] "An Embedded System for Artificial Intelligence", I. Panagopoulos, C. Pavlatos, G. Papakonstantinou, International Journal of Computational Intelligence, vol.1 no1-4, 2004

[5.4] "An efficient context–free parsing algorithm", J. Earley, Communications of ACM, vol.13, pp. 94-102, 1970.

[5.5] "Parallel parsing algorithms and VLSI implementation for syntactic pattern recognition", Y. Chiang and K. Fu, IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-6, May 1984.

[5.6] "Knowledge representation using a Modified Earley's Algorithm", C. Pavlatos, I. Panagopoulos, G, Papakonstantinou, Lectures of Notes in Artificial Intelligence, Springer Verlag, pp.321-330, 2004

[5.7] "An Intelligent Embedded System for Control Applications", C. Pavlatos, A. Dimopoulos, G. Papakonstantinou, Workshop on Modeling and Control of Complex Systems, Cyprus, July 2005

[5.8] "An Efficient Hardware Implementation for AI applications", A. Dimopoulos, C. Pavlatos, I. Panagopoulos, G. Papakonstantinou,SETN'06, Heraklion, Crete, May 2006

6. Εφαρμογές

Οι αρχιτεκτονικές που υλοποιήθηκαν εφαρμόστηκαν κυρίως σε εφαρμογές Συντακτικής Αναγνώρισης Προτύπων και Τεχνητής Νοημοσύνης. Υλοποιώντας κατά περίπτωση κατάλληλο συντακτικό και σημασιολογικό αναλυτή, επιτυγχάνοντας βελτίωση στην απόδοση των εφαρμογών αυτών όπως αναλύθηκε σε προηγούμενα κεφάλαια.

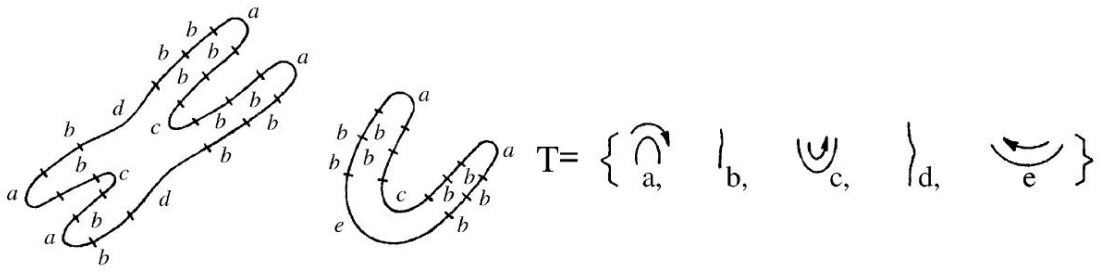
6.1 Εφαρμογές Συντακτικής Αναγνώρισης Προτύπων

Στο πεδίο της Συντακτικής Αναγνώρισης Προτύπων υλοποιήθηκε η αναγνώριση του ΗλεκρτοΚαρδιοΓραφήματος (ΗΚΓ) τόσο με τον παράλληλο συντακτικό αναλυτή των Chiang & Fu [6.8] όσο και με το παράλληλο συνδιαστικό κύκλωμα [6.1]. Στη συντακτική αναγνώριση προτύπων, το κομμάτι της αναγνώρισης ουσιαστικά περιορίζεται στην ανάλυση μιας γλωσσολογικής αναπαράστασης των προτύπων προς αναγνώριση, με την χρήση ενός συντακτικού αναλυτή που χρησιμοποιεί μια συγκεκριμένη γραμματική που καλείται “pattern grammar”. Η γραμματική αυτή περιγράφει τα πρότυπα προς αναγνώριση, και η διατύπωση και συντακτική ανάλυση της είναι πάντα τα κρίσιμα υποπροβλήματα σε εφαρμογές αναγνώριση προτύπων που πρόκειται να αντιμετωπιστούν με συντακτικές μεθόδους. Στην περίπτωση του ΗΚΓ, όπου έχουμε έναν μεγάλο αριθμό διαφορετικών μορφολογιών, όπου επιπρόσθετες μορφολογίες μπορούν να δημιουργηθούν εξαιτίας του θορύβου, και όπου οι μετρήσεις διάφορων παραμέτρων πρέπει να εκτελεσθούν, χρειάζονται ισχυρές γραμματικές ικανές να περιγράψουν τόσο το συντακτικό όσο και το σημασιολογικό κομμάτι. Λόγω της εκφραστικής τους δύναμής οι κατηγορηματικές γραμματικές επιλέχτηκαν και χρησι-

μοποιήθηκαν σαν μοντέλο για την διατύπωση των pattern grammar για ΗΚΓ. Στην εργασία [6.9] το αλφάβητο των τερματικών συμβόλων Τ ={Κ+, Κ-, Ε, Π} έχει υιοθετηθεί για την κωδικοποίηση του ΗΚΓ, όπου Κ+ δηλώνει τις θετικές αιχμές, Κ- αρνητικές αιχμές, Ε τμήματα ευθειών γραμμών και Π παραβολικά τμήματα. Κατά συνέπεια η κυματομορφή του ΗΚΓ αναπαρίσταται γλωσσολογικά ως μια συμβολοσειρά από το αλφάβητο Τ. Επομένως μια συμβολοσειρά της ακόλουθης μορφής μπορεί να είναι η γλωσσολογική αναπαράσταση της κυματομορφή ενός ΗΚΓ : Π Κ- Κ+ Π Κ- Ε Κ- Κ+ Π Κ- Κ+ Ε Κ- Ε Κ- Κ+ Ε

Με την χρήση των αρχιτεκτονικών που παρουσιάστηκαν σε προηγούμενα κεφάλαια και κατάλληλης γραμματικής υλοποιήθηκε συντακτικός και σημασιολογικός αναλυτής, επιτυγχάνοντας βελτίωση στην απόδοση της αναγνώρισης του ΗΚΓ.

Μία άλλη εφαρμογή που υλοποιήθηκε ήταν αυτή της αναγνώρισης των χρωμοσωμάτων [6.1], [6.5]. Η γλωσσολογική αναπαράσταση των χρωμοσωμάτων φαίνεται στο παρακάτω σχήμα (Σχ. 31). Τα χρωμοσώματα εκφρασμένα σε συμβολοσειρές της μορφής *babcbbabbcbba* μπορούν να αναγνωριστούν από τον συντακτικό αναλυτή της γραμματικής Gchrom.



Σχήμα 31. Η γλωσσολογική αναπαράσταση των χρωμοσωμάτων

Gchrom = {N, T, P, S} where N : {submedian chromosome, telocentric chromosome, arm pair, left part, right part, arm, side, bottom}, T : {a, b, c, d, e}, P : {submedian chromosome → arm_pair arm_pair, telocentric chromosome → bottom arm_pair, arm_pair → side arm_pair, arm_pair → arm_pair side, arm_pair → arm right_part,

arm_pair → left_part arm, left_part → arm c, right part → c arm, bottom → bottom b, bottom → b bottom, bottom → e, side → side b, side → b side, side → b, side → d, arm → arm b, arm → b arm, arm → a},
S : {submedian chromosome, telocentric chromosome}
Με την χρήση των αρχιτεκτονικών που παρουσιάστηκαν σε προηγούμενα κεφάλαια και της γραμματικής Gchrom υλοποιήθηκε συντακτικός αναλυτής, επιτυγχάνοντας βελτίωση στην απόδοση της αναγνώρισης του ΗΚΓ.

Τέλος, έχουν επίσης υλοποιηθεί συστήματα αναγνώρισης συμβολοσειρών της γλώσσας προγραμματισμού Java [6.1] και προτάσεων φυσικής γλώσσας [6.4].

6.2 Εφαρμογές Τεχνητής Νοημοσύνης

Ενώ, στο πεδίο της Τεχνητής Νοημοσύνης υαλοποιήθηκε το παιχνίδι "Wumpus World" [6.2], [6.3], [6.7] και το πρόβλημα εύρεσης μονοπατιού διασύνδεσης κόμβων σε μη κυκλικό γράφο [6.6].

Στην ενότητα αυτή εστιάζουμε στην παρουσίαση χρήσης της προτεινόμενης υλοποίησης σε ένα σύστημα ευφυούς ελέγχου. Ένα Ε.Σ., εφοδιασμένο με τον προτεινόμενο επεξεργαστή, ελέγχει την κίνηση μίας αναπηρικής καρέκλας σε δωμάτιο. Οι βηματικοί κινητήρες της καρέκλας επιτρέπουν μόνο κατακόρυφες και οριζόντιες κινήσεις (δεν επιτρέπεται η κίνηση
διαγωνίως). Επιβάλλουμε τον περιορισμό αυτόν για να δείξουμε καθαρότερα τα χαρακτηριστικά της υλοποίησης. Σε πραγματικές συνθήκες, είναι δυνατό να υποστηριχτεί και κίνηση διαγωνίως με ανάλογη επέκταση των λογικών κανόνων και επιλογών. Τα αποτελέσματα θα είναι τα ίδια με αυτά που θα παρουσιαστούν εδώ. Θεωρούμε επίσης ότι η κίνηση στις τέσσερις κατευθύνσεις γίνεται με διακριτά βήματα, δηλαδή, κάθε σήμα που στέλνεται στους βηματικούς κινητήρες προκαλεί την ενεργοποίησή τους για κάποιο χρονικό διάστημα και έτσι την κίνηση της αναπηρικής καρέκλας για συγκεκριμένη απόσταση (d) προς την επιθυμητή κατεύθυνση (Βορράς, Νότος, Ανατολή, Δύση). Η λειτουργία αυτή μας

επιτρέπει να θεωρήσουμε ότι το δωμάτιο χωρίζεται σε ένα πλέγμα κελιών μεγέθους dxd. Κάθε κελί του πλέγματος μπορεί να καλύπτεται από κάποιο σταθερό εμπόδιο (έπιπλα, συσκευές), ή ένα κινούμενο εμπόδιο (άνθρωποι, ζώα) ή και την ίδια την καρέκλα. Μία κάμερα στο κέντρο της οροφής του δωματίου δίνει την απαραίτητη πληροφορία για κάθε σημείο του χώρου.

Για τον πειραματισμό μας με την εφαρμογή, επιλέξαμε την υλοποίηση ενός ανάλογου παραδείγματος που βασίζεται στο πολύ γνωστό "Wumpus World Game". Συγκεκριμένα εφαρμόσαμε μερικές τροποποιήσεις στο εν λόγω παιχνίδι και στις ιδιότητες του κόσμου για να είναι παρόμοιο με την ζητούμενη εφαρμογή. Σε αυτήν την τροποποίηση, ο ήρωας μπορεί να κινηθεί κάθετα η οριζόντια σε ένα πλέγμα προσπαθώντας να βρει το κελί που περιέχει το χρυσό. Ο Wumpus κινείται τυχαία στο χώρο προσπαθώντας να φάει τον ήρωα. Ο ήρωας μέσα σε μία χρονική στιγμή, μπορεί είτε να κινηθεί προς μία από τις τέσσερις κατευθύνσεις, είτε να αντιληφθεί το χώρο γύρω του χάνοντας χρόνο, δηλαδή, να δει τι βρίσκεται στα γειτονικά τετράγωνα. (να βρει δηλαδή αν υπάρχει χαντάκι (Pit), ο Wumpus η ο χρυσός (Gold) σε κάποιο γειτονικό κελί). Το μόνο που μπορεί να αντιληφθεί ο ήρωας χωρίς την ενέργεια «perceive» είναι το γεγονός ότι ο Wumpus βρίσκεται κάπου γύρω του.

Το παιχνίδι αυτό είναι ανάλογο με την εφαρμογή κίνησης της αναπηρικής καρέκλας. Θεωρούμε ότι ο ήρωας αντιπροσωπεύει την καρέκλα, ο Wumpus τα κινούμενα αντικείμενα, τα Pits τα σταθερά αντικείμενα και ο χρυσός τον επιθυμητό προορισμό. Έτσι για το υπόλοιπο της ενότητας θα ασχοληθούμε με το παιχνίδι "Wumpus world" για την παρουσίαση των πλεονεκτημάτων της υλοποίησής μας.

Αρχικά ορίζουμε την βάση γνώσης του προβλήματος που αποτελείται από τα παρακάτω γεγονότα

- Pit (x,y) Αληθής αν υπάρχει χαντάκι στο x,y
- Gold(x,y) Αληθής αν υπάρχει ο χρυσός στο x,y
- Wumpus(x,y) Αληθής αν ο Wumpus είναι στο x,y
- Empty(x,y) Αληθής αν το x,y είναι άδειο
- DontKnow(x,y) Αληθής αν ο ήρωας δε γνωρίζει τη υπάρχει στο x,y

- BeenAt(x,y,l) Ο ήρωας επισκέφτηκε το x,y l φορές
- At(x,y) Ο ήρωας είναι στο x,y

Οι ενέργειες που μπορεί να εκτελέσει ο ήρωας και τα αποτελέσματά τους είναι τα εξής.

- Go North Αν στο x,y πήγαινε στο x,y-1
- Go South Αν στο x,y πήγαινε στο x,y+1
- Go East Αν στο x,y πήγαινε στο x+1,y
- Go West Αν στο x,y πήγαινε στο x-1,y
- Perceive Αν στο x,y τότε μάθε τι έχουν τα x-1,y , x+1,y, x,y+1 και d x,y-1

Τέλος, οι συνέπειες που πρέπει να ελέγχονται από το σύστημα αξιολόγησης είναι οι εξής

- GoldFound Ο ήρωας βρήκε το χρυσό
- WumpusEats Ο Wumpus έφαγε τον ήρωα
- PitFall Ο ήρωας έπεσε σε χαντάκι
- Nothing Δεν συνέβη τίποτα

Κατασκευάστηκε περιβάλλον αξιολόγησης το οποίο αποτελείται από δύο δυσδιάστατους πίνακες, όπου ο ένας κρατάει τα χαρακτηριστικά του κόσμου (τι περιέχεται σε κάθε κελί) (world[i][j]) ενώ ο άλλος αποτελεί τη γνώση του ήρωα (ποια κελιά από τα προηγούμενα είναι γνωστά στον ήρωα) (Knowledge[i][j]). Ο Wumpus ελέγχεται από το περιβάλλον αξιολόγησης
και η κίνησή του είναι τελείως τυχαία (καμία μορφή νοημοσύνης). Ο ήρωας βασισμένος στην τρέχουσα γνώση που αποκτά για τον κόσμο επιλέγει την ενέργεια που θεωρεί καταλληλότερη και την εκτελεί. Ο μηχανισμός ευφυΐας του εφαρμόζει βάρη σε καθεμία από τις πιθανές κινήσεις στο χώρου που κυμαίνονται από 0 μέχρι 1. Η κίνηση με το μεγαλύτερο βάρος, επιλέγεται. Αν καμία κίνηση δεν ξεπερνάει ένα συγκεκριμένο κατώφλι THR τότε ο ήρωας προτιμά να αντιληφθεί το χώρο, μέσω της ενέργειας «Perceive» στην τρέχουσα χρονική μονάδα.
Για να προσθέσουμε κάποιας μορφής τυχαιότητα στα βάρη κίνησης (στην περίπτωση που δύο ή περισσότερα βάρη έχουν την ίδια τιμή) εφαρμόζουμε επίσης τη μεταβλητή «random» που κυμαίνεται από (-0,1 έως 0,1) και επηρεάζει τα βάρη διαφοροποιώντας τα, στην περίπτωση που είναι ίδια. Τέλος, επιλέξαμε επίσης να

ενσωματώσουμε μία μεταβλητή «courage» που κυμαίνεται από (0 έως 0.2) και αντικατοπτρίζει το θάρρος του ήρωα, δηλαδή, πόσο πιθανό είναι ο ήρωας να κινηθεί σε κελί του οποίου το περιεχόμενο δε γνωρίζει. Τα βάρη αυτά υπολογίζονται βάσει των δεδομένων που έχει κάθε στιγμή ο ήρωας για τον κόσμο στη βάση γνώσης του. Έτσι, επεκτείναμε τα προαναφερθέντα γεγονότα με παραμέτρους που επιτρέπουν τον υπολογισμό αυτών των βαρών. Με την εκτέλεση οποιασδήποτε ενέργειας, το σύστημα αξιολόγησης ελέγχει για οποιαδήποτε συνέπεια που αυτή μπορεί να είχε. Αν υπάρχει κάποια σοβαρή συνέπεια (θάνατος ήρωα, εύρεση χρυσού) τότε η προσομοίωση τελειώνει. Σημειώνεται ότι η επιτυχία/αποτυχία του ήρωα να βρει το χρυσό εξαρτάται από το χρόνο που απαιτείται για να αποφασίσει την ενέργεια που θα επιλέξει αφού μια γρηγορότερη μηχανής λογικής επιτρέπει γρηγορότερους συλλογισμούς και καθιστά πιο απίθανο ο Wumpus να κινηθεί σε κελί που βρίσκεται ο ήρωας. Βάσει των παραπάνω σχηματίστηκε το λογικό πρόγραμμα το οποίο, με την μεθοδολογία που έχει περιγραφεί σε προηγούμενο κεφάλαιο, ματασχηματίστηκε στη αντίστοιχη Κατηγορηματική Γραμματική. Με την χρήση των αρχιτεκτονικών που παρουσιάστηκαν σε προηγούμενα κεφάλαια και της γραμματικής αυτής υλοποιήθηκε συντακτικός αναλυτής, επιτυγχάνοντας βελτίωση στην απόδοση της εφαρμογής.

6.3 Αναφορές

[6.1] "Optimal Reconfigurable Embedded Parsers", C. Pavlatos, Alexandros Dimopoulos, Andrew Koulouris, Theodore Andronikos, Ioannis Panagopoulos and George Papakonstantinou, Computer Languages, Systems & Structures, DOI link: http://dx.doi.org/10.1016/j.cl.2007.08.001

[6.2] "An Embedded System for Intelligent Control", I. Panagopoulos, C. Pavlatos and G. Papakonstantinou, Journal of Intelligent and Robotics Systems,vol.42,p.179-211,2005

[6.3] "An Embedded System for Artificial Intelligence", I. Panagopoulos, C. Pavlatos, G. Papakonstantinou, International Journal of Computational Intelligence, vol.1 no1-4, 2004

[6.4] "Hardware Natural Language Interface", Christos Pavlatos, Alexandros Dimopoulos and George Papakonstantinou, AIAI 2007, Athens 2007

[6.5] "Efficient Signal Processing Using Syntactic Pattern Recognition Methods", Andrew Koulouris, Theodore Andronikos, Christos Pavlatos, Alexandros Dimopoulos, Ioannis Panagopoulos and George Papakonstantinou, SIP 2007, Honolulu USA,Aug. 2007

[6.6] "An Efficient Hardware Implementation for AI applications", A. Dimopoulos, C. Pavlatos, I. Panagopoulos, G. Papakonstantinou,SETN'06, Heraklion, Crete, May 2006

[6.7] "An Intelligent Embedded System for Control Applications", C. Pavlatos, A. Dimopoulos, G. Papakonstantinou, Workshop on Modeling and Control of Complex Systems, Cyprus, July 2005

[6.8] "An embedded system for the electrocardiogram recognition", C. Pavlatos, A. Dimopoulos, G. Papakonstantinou, EMBEC'05, Prague, Czech Republic, November 2005

[6.9] , "Syntactic Pattern Recognition of the ECG", P. Trahanias, E. Skordalakis, IEEE Transactions on PAMI, vol. 12, no. 7, July 1990

7. Επίλογος - Μελλοντικές Επεκτάσεις

Η παρούσα διδακτορική διατριβή παρουσίασε ένα σύνολο από αυτοματοποιημένες και μη τεχνικές συσχεδίασης Υλικού/Λογισμικού ενσωματωμένων συστημάτων. Οι προτεινόμενες σχεδιάσεις κινήθηκαν με βασικό γνώμονα την απλότητα, επαναχρησιμοποίηση υλικού, αύξηση της τελικής απόδοσης, εύκολο προγραμματισμό και γρήγορη εξερεύνηση του χώρου σχεδίασης. Έτσι, οι μεθοδολογίες που προέκυψαν, συμβάλλουν στην αποδοτικότερη σχεδίαση ενσωματωμένων συστημάτων, παρουσιάζοντας λύσεις σχεδίασης, οι οποίες καλύπτουν τις τρέχουσες ανάγκες σχεδίασης ενσωματωμένων συστημάτων.

Πιο συγκεκριμένα η διδακτορική διατριβή επικεντρώθηκε :

- Στην ανάπτυξη ενός νέου εργαλείου που συνδυάζει τόσο το δηλωτικό όσο και το διαδικαστικό μοντέλο προγραμματισμού υλοποιώντας σε υλικό έναν ακολουθιακό συντακτικό αναλυτή.
- Τη βελτίωση του καλλίτερου στην βιβλιογραφία , παράλληλου συντακτικού αναλυτή και υλοποίηση του σε υλικό.
- Την επέκταση και βελτίωση του παραπάνω παράλληλου συντακτικού αναλυτή για να καλύψει και τη σημασιολογία της γραμματικής.
- Την ανάπτυξη ενός νέου εργαλείου, με τεχνικές Συσχεδίασης Υλικού/Λογισμικού και ορίζοντας κατάλληλο interface μεταξύ των δύο, που συνδυάζει τόσο το δηλωτικό όσο και το διαδικαστικό μοντέλο προγραμματισμού βασισμένο στον παραπάνω βελτιωμένο παράλληλο συντακτικό αναλυτή.

- Την ανάπτυξη μιας πλατφόρμας αυτόματης παραγωγής Ενσωματωμένων Συστημάτων για τέτοιου είδους υβριδικές εφαρμογές, βασισμένης στο παραπάνω εργαλείο .
- Στην εφαρμογή των παραπάνω εργαλείων σε πρακτικές εφαρμογές.

Ως μελλοντικές επεκτάσεις της παρούσας διατριβής αναφέρονται τα παρακάτω:

- Η επέκταση του παράλληλου αλγορίθμου συντακτικής ανάλυσης που υλοποιείται με συνδιαστικό κύκλωμα με σκοπό να καλύπτει την σημασιολογία της γραμματικής
- Η πλήρης απεικόνιση του αποτιμητή κατηγορηματικών γραμματικών σε υλικό χωρίς να είναι αναγκαία η χρήση μικροεπεξεργαστή.
- Η επέκταση του αυτόματου εργαλείου σε εφαρμογές Fuzzy Logic

Δημοσιεύσεις Υποψηφίου Διδάκτορα

Περιοδικά

[1] "Optimal Reconfigurable Embedded Parsers", C. Pavlatos, Alexandros Dimopoulos, Andrew Koulouris, Theodore Andronikos, Ioannis Panagopoulos and George Papakonstantinou, Computer Languages, Systems & Structures, DOI link: http://dx.doi.org/10.1016/j.cl.2007.08.001

[2] "An Embedded System for Intelligent Control", I. Panagopoulos, C. Pavlatos and G. Papakonstantinou, Journal of Intelligent and Robotics Systems,vol.42,p.179-211,2005

[3] "An Embedded System for Artificial Intelligence", I. Panagopoulos, C. Pavlatos, G. Papakonstantinou, International Journal of Computational Intelligence, vol.1 no1-4, 2004

Συνέδρια

[4] "Hardware Natural Language Interface", Christos Pavlatos, Alexandros Dimopoulos and George Papakonstantinou, AIAI 2007, Athens 2007

[5] "Hardware Solution of a First-Order Diophantine Equation", Ioannis Panagopoulos, Christos Pavlatos, Alexandros Dimopoulos and George Papakonstantinou, HERCMA 2007, Athens 2007

[6] "A Flexible General-Purpose Parallelizing Architecture for Nested Loops in Reconfigurable Platforms", Ioannis Panagopoulos, Christos Pavlatos, George Manis and George Papakonstantinou, Patmos 2007, Sweden, Sept. 2007

[7] "Efficient Signal Processing Using Syntactic Pattern Recognition Methods", Andrew Koulouris, Theodore Andronikos, Christos Pavlatos, Alexandros Dimopoulos, Ioannis Panagopoulos and George Papakonstantinou, SIP 2007, Honolulu USA,Aug. 2007

[8] "An Efficient Hardware Implementation for AI applications", A. Dimopoulos, C. Pavlatos, I. Panagopoulos, G. Papakonstantinou,SETN'06, Heraklion, Crete, May 2006

[9] "An Intelligent Embedded System for Control Applications", C. Pavlatos, A. Dimopoulos, G. Papakonstantinou, Workshop on Modeling and Control of Complex Systems, Cyprus, July 2005

[10] "Hardware implementation of PAN & TOMPKINS QRS detection algorithm", C. Pavlatos, A. Dimopoulos, G. Manis, G. Papakonstantinou, EMBEC'05, Prague, Czech Republic, November 2005

[11] "An embedded system for the electrocardiogram recognition", C. Pavlatos, A. Dimopoulos, G. Papakonstantinou, EMBEC'05, Prague, Czech Republic, November 2005

[12] "Knowledge representation using a Modified Earley's Algorithm", C. Pavlatos, I. Panagopoulos, G, Papakonstantinou, Lectures of Notes in Artificial Intelligence, Springer Verlag, pp.321-330, 2004

[13] "A programmable Pipelined Coprocessor for Parsing Applications" C. Pavlatos, I. Panagopoulos, G, Papakonstantinou, Workshop on Application Specific Processors (WASP), Stockholm, Sept. 2004

[14] "An extended RISC microprocessor for Attribute Grammar Evaluation", I. Panagopoulos, C. Pavlatos, G. Papakonstantinou, SAC2004, ACM Symposium on Applied Computers, Nicosia, Cyprus, 2004

[15] "Hardware Implementation of Syntactic Pattern Recognition Algorithms C.Pavlatos, I. Panagopoulos, G, Papakonstantinou," The International Association of Science and Technology for Development Conference (SPPRA 2003), Rhodes, Greece, 2003

Πίνακας Ελληνο-Αγγλικών Όρων

AG - Attribute Grammar	Κατηγορηματική Γραμματική
AI – Artificial Intelligence	Τεχνητή Νοημοσύνη
Ambiguous Grammar	Διφορούμενη Γραμματική
Attribute	Κατηγόρημα
Bit Vector	Διάνυσμα Δυαδικών Ψηφίων
CFG – Context Free Grammar	Γραμματική χωρίς Συμφραζόμενα
Compiler	Μεταγλωττιστής
Inference Engine	Μηχανή Εξαγωγής Συμπερασμάτων
Inference Rule	Κανόνας Εξαγωγής Συμπερασμάτων
Inherited Attribute	Κληρονομούμενο Κατηγόρημα
lhss – left hand side symbol	Αριστερό μέλος του κανόνα παραγωγής
LUT – Look up Table	Πίνακας Αναζήτησης
Module	Λειτουργικό Τμήμα
Parse Table	Συντακτικός Πίνακας
Parse Tree	Συντακτικό Δέντρο
Parser	Συντακτικός Αναλυτής
Pattern Recognition	Συντακτική Αναγνώριση
Pattern Grammar	Γραμματική προτύπων
rhss – right hand side symbol	Δεξιό μέλος του κανόνα παραγωγής
Synthesized Attribute	Συντιθέμενο Κατηγόρημα
Soc – System on a chip	Σύστημα στην ίδια πλακέτα
Unification Process	Διαδικασία Ενοποίησης

Κατάλογος Σχημάτων

Κατάλογος Πινάκων