Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων Και Μηχανικων Υπολογιστων

Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Υπολογιστικών Συστημάτων

Ευφυή Ενσωματωμένα Συστήματα

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Αλέξανδρος Κ. Δημόπουλος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών
Εθνικού Μετσόβιου Πολυτεχνείου (2004)

Αθήνα, Απρίλιος 2009

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Εργαστήριο Υπολογιστικών Συστημάτων

Ευφυή Ενσωματωμένα Συστήματα

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Αλέξανδρος Κ. Δημόπουλος

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών
Εθνικού Μετσόβιου Πολυτεχνείου (2004)

Συμβουλευτική	Γεώργιος Παπακωνσταντίνου
Επιτροπή:	Παναγιώτης Τσανάκας
	Νεκτάριος Κοζύρης

Εγκρίθηκε από την επταμελή εξεταστική επιτροπή την $13^{η}$ Απριλίου 2009

.....	.....	.....
Γεώργιος Παπακωνσταντίνου	Παναγιώτης Τσανάκας	Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Αν. Καθηγητής Ε.Μ.Π.
.....	.....	.....
Κωνσταντίνα Νικήτα	Ανδρέας - Γεώργιος	Στέφανος Κόλλιας
Καθηγήτρια Ε.Μ.Π.	Σταφυλοπάτης	Καθηγητής Ε.Μ.Π.
	Καθηγητής Ε.Μ.Π.	

.....
Μαρία Νικολαΐδου
Αν. Καθηγήτρια
Χαροκοπείου Πανεπιστημίου

Αθήνα, Απρίλιος 2009

.....

Αλέξανδρος Κ. Δημόπουλος

Διδάκτωρ Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π

©Αλέξανδρος Κ. Δημόπουλος, 2009
Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

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

ignoramus et ignorabimus
Emil du Bois-Reymond

Περιεχόμενα

Περίληψη

Τα τελευταία χρόνια, λόγω της αυξημένης ζήτησης σε υπολογιστική ισχύ σε συνδυασμό με την απαίτηση για μεταφερσιμότητα (portability) παρατηρείται μια μεγάλη αύξηση του πεδίου που βρίσκουν εφαρμογή τα Ενσωματωμένα Συστήματα. Ως Ενσωματωμένο Σύστημα (ΕΣ) μπορεί να οριστεί ένα σύστημα - περιορισμένου φυσικού μεγέθους - ειδικού σκοπού το οποίο επιτελεί μία συγκεκριμένη και προκαθορισμένη εργασία. Μάλιστα, αναλόγως με τους χρονικούς περιορισμούς της εφαρμογής χωρίζονται σε δύο μεγάλες κατηγορίες, σε αυτά που πρέπει να αντεπεξέλθουν αυστηρά σε εφαρμογές πραγματικού χρόνου (hard real time embedded) και σε εκείνα τα οποία είναι πιο χαλαρά σε σχέση με τους χρονικούς περιορισμούς (soft real time embedded). Για να εκτελέσει την εργασία του, το ΕΣ είναι συνήθως εφοδιασμένο με έναν μικροελεγκτή ή ακόμη και μικροεπεξεργαστή και έχει όσο το δυνατόν μικρότερες απαιτήσεις ενέργειας και σχετικά χαμηλό κόστος. Συνεπώς, αντί για την επιλογή ενός γενικής χρήσης συστήματος, προτιμάται για συγκεκριμένες εφαρμογές, η χρήση ενός συστήματος που μπορεί να εκτελέσει αποκλειστικά και μόνο μια συγκεκριμένη εργασία αλλά είναι μικρό σε μέγεθος, έχει χαμηλή κατανάλωση ισχύος, αυξημένες χρονικές απαιτήσεις και το κόστος του είναι χαμηλό. Με την ολοένα αυξανόμενη χρήση των ΕΣ, υπάρχει μια μετατόπιση των εφαρμογών από το πεδίο του καθαρού λογισμικού που εκτελείται σε υλικό γενικής χρήσης (pure software on general purpose hardware) σε αυτό της συνύπαρξης εξειδικευμένου υλικού/λογισμικού (dedicated hardware). Στη μετατόπιση αυτή έχει βοηθήσει σημαντικά και η εξέλιξη του προγραμματιζομένου υλικού και πιο συγκεκριμένα των FPGA (Field Programmable Gate Arrays), που αποτελούν μονάδες υλικού που επιτρέπουν τον εσωτερικό προγραμματισμό του υλικού τους, ώστε να επιτελείται μια συγκεκριμένη εργασία. Έτσι, ο προγραμματισμός περνάει σε ένα νέο επίπεδο, στο οποίο ο προγραμματιστής πια δεν καλείται να γράψει κώδικα που θα εκτελεστεί σε ένα συγκεκριμένο επεξεργαστή αλλά ο κώδικας περιγράφει το ίδιο το εξειδικευμένο υλικό. Η περιγραφή αυτή γίνεται με ειδικές γλώσσες περιγραφής υλικού (Hardware Description Languages - HDL), με κυριότερες τις Verilog και VHDL.

Η συγκεκριμένη διατριβή έχει ως αντικείμενο την υλοποίηση σε υλικό (FPGA) κατάλληλων αλγορίθμων που προσθέτουν "ευφυΐα" σε ένα ΕΣ, μέσω της αναγνώρισης προτάσεων που ανήκουν σε γραμματικές χωρίς συμφραζόμενα καθώς και την αναγνώριση και υπολογισμό των αντίστοιχων κατηγορημάτων για προτάσεις που ανήκουν σε κατηγορικές γραμματικές. Ως γραμματική ορίζεται ένα σύστημα από κανόνες παραγωγής συμβολοσειρών. Οι κατηγορίες των γραμματικών, όπως έχουν οριστεί από τον Chomsky, ποικίλλουν αλλά στη συγκεκριμένη εργασία θα χρησιμοποιηθούν οι γραμματικές χωρίς συμφραζόμενα και μια επέκταση αυτών, οι κατηγορικές γραμματικές. Για τη συντακτική αναγνώριση μιας πρότασης, υπάρχουν κατάλληλοι αλγόριθμοι που ελέγχουν εάν η πρόταση ανήκει σε μια δοσμένη γραμματική ή όχι. Οι αλγόριθμοι αυτοί, που απλώς απαντούν δυαδικά με ένα ναι ή ένα όχι για κάθε πρόταση, ονομάζονται αναγνωριστές (recognizer). Στην περίπτωση που κατά τη διάρκεια της αναγνώρισης,

παράγεται και το συντακτικό δέντρο αναγνώρισης (parse tree) για τη συγκεκριμένη πρόταση, τότε ο αλγόριθμος ονομάζεται συντακτικός αναλυτής (parser). Επιπλέον, προτείνεται ένα σύστημα για την αυτόματη παραγωγή ευφυών ΕΣ. Αντί να περιγράφεται το ΕΣ με τις κλασικές γλώσσες Verilog και VHDL περιγράφεται σε υψηλότερο δηλωτικό επίπεδο με τη χρήση του συμβολισμού των κατηγορικών γραμματικών.
Η υλοποίηση των κατηγορικών γραμματικών βασίζεται στην παράλληλη υλοποίηση του αλγορίθμου του Earley και αποτελεί επέκταση του τελευταίου, επιτρέποντας την υλοποίησή του με μικρότερες απαιτήσεις χώρου αλλά και χρονικές απαιτήσεις. Επιπλέον, έχει επεκταθεί κατάλληλα ο αλγόριθμος προκειμένου να μπορεί να χειρισθεί και κατηγορικές γραμματικές και να υπολογίζει τα αντίστοιχα κατηγορήματα, που καθορίζουν τη σημασιολογία της γραμματικής. Για τον υπολογισμό των κατηγορημάτων, είναι απαραίτητη η ύπαρξη κάποιας μονάδας που να εκτελεί τις αναγκαίες πράξεις. Κατά τη διάρκεια της συγκεκριμένης εργασίας, έγινε χρήση διαφορετικών αρχιτεκτονικών για τη συντακτική ανάλυση και τον υπολογισμό των κατηγορημάτων. Συγκεκριμένα, για τον υπολογισμό των κατηγορημάτων - και ανάλογα με τις απαιτήσεις σε υπολογιστική ισχύ και μέγεθος - επιλέγεται ενίοτε η χρήση μικροελεγκτή, η χρήση εξωτερικού μικροεπεξεργαστή γενικής χρήσης, η χρήση εσωτερικού μικροεπεξεργαστή γενικής χρήσης και τελικά η χρήση επεξεργαστή εξειδικευμένης χρήσης ειδικά σχεδιασμένου για τις ανάγκες κάθε εφαρμογής.

Abstract

In recent years, due to the increased demand for computing power, combined with the requirement for portability, a large increase in the application field of embedded systems is observed. An Embedded System (ES) may be defined as a special purpose system with limited physical size, which performs a specific and predetermined task. Indeed, depending on the time constraints of the application, ES can be divided into two broad categories: those where real-time applications must be handled strictly (hard real time embedded) and those where time restrictions do not matter that much (soft real time embedded). In order to carry out the desired task, an ES is usually equipped with a microcontroller or even a microprocessor and has the lowest possible power requirements and relatively low cost. Therefore, instead of selecting a general purpose system, the usage of a system that can perform only a particular task but is small in size, has low power consumption, increased time requirements and low cost is preferred for specific applications. With the growing use of ES, there is a shift in applications from the field of pure software running on general purpose hardware to the field of specialized hardware/software coexistence (dedicated hardware). This shift has been significantly influenced by the progress in programmable hardware and more specifically of FPGA (Field Programmable Gate Arrays), which are hardware units that can be hardware programmed in order to execute a specific task. Thus, the programming goes to a new level, where the programmer is no longer required to write source code that will run on a specific processor but the code describes the hardware itself. The description is made with special Hardware Description Languages (HDL), typical HDL are Verilog and VHDL.

The main purpose of this thesis is the implementation in hardware (FPGA) of appropriate algorithms that add "intelligence" to an ES, through the recognition of sentences that belong in a context free grammar (CFG) and through the evaluation of the corresponding attributes for sentences belonging to an attribute grammar (AG). A grammar is defined as a system of production rules that produce strings. The grammar categories, as defined by Chomsky, vary but in the specific thesis only CFG and an extension of them, AG, will be used. For the syntactic recognition of a sentence, suitable algorithms are used to determine whether or not a sentence belongs to a given grammar. These binary algorithms, which only respond with a yes or a no for each sentence, are called recognizers. In the case that during the recognition, a parse tree is also constructed for the given sentence, the algorithm is called parser. Furthermore, a platform for the automatic production of intelligent ES is proposed. Rather than describing the ES in classical Verilog or VHDL, it is described at a higher level using the symbolism of attribute grammars.

The implementation of attribute grammars is based on the parallel implementation of Earley's algorithm and is an extension of the latter, allowing its implementation with smaller space and time requirements. Moreover, it has been properly

extended in order to be capable to also handle attribute grammars and evaluate the corresponding attributes, defining the semantics of the grammar. To evaluate these attributes, a unit to perform the necessary actions is essential. During this work, different architectures for the syntactic analysis and the attribute evaluation have been used. More specifically, to evaluate the attributes depending on the requirements in computing power and size, the followng has been chosen; the usage of a microcontroller, an external general purpose microprocessor, an internal general purpose microprocessor and finally the usage of special purpose processor specially designed for the needs of each application.

Πρόλογος

Η παρούσα διδακτορική διατριβή εκπονήθηκε κατά το διάστημα 2004 – 2009 στο Εργαστήριο Υπολογιστικών Συστημάτων (CSLab), της σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου, υπό την επίβλεψη του καθηγητή Γιώργου Παπακωνσταντίνου. Ξεκινώντας, θα ήθελα να κάνω μια αναφορά σε αυτά τα πέντε χρόνια, διάστημα κατά το οποίο είχα την χαρά να βρίσκομαι σε έναν εργαστηριακό χώρο πολύ παραγωγικό, από ερευνητικής σκοπιάς αλλά και πολύ ανθρώπινο. Ξεκινώντας από το διευθυντή του εργαστηρίου τον καθηγητή Γιώργο Παπακωνσταντίνου, θέλω να τον ευχαριστήσω για το κλίμα που έχει δημιουργήσει στο εργαστήριο, που διευκολύνει αφάνταστα την πολύωρη καθημερινή εργασία στο χώρο. Ακόμη περισσότερο όμως θέλω να τον ευχαριστήσω για το δεύτερο και καθοριστικό ρόλο του, εκείνον του επιβλέποντα καθηγητή που καθημερινά ήταν παρών σε όλα τα ευχάριστα και δυσάρεστα γεγονότα, ερευνητικά και μη. Με τον δικό του μοναδικό και ήπιο τρόπο δίνει τις σωστές κατευθύνσεις στις ανησυχίες και στα καθημερινά προβλήματα, αντιμετωπίζοντας όλα τα παιδιά του εργαστηρίου με πατρική στοργή, κερδίζοντας το σεβασμό αλλά και την εκτίμηση όλων μας. Θα ήθελα επίσης να ευχαριστήσω και τα άλλα δύο μέλη ΔΕΠ του εργαστηρίου, καθώς και μέλη της τριμελούς συμβουλευτικής επιτροπής μου, τον καθηγητή Παναγιώτη Τσανάκα και τον αναπληρωτή καθηγητή Νεκτάριο Κοζύρη που συμβάλουν στην εύρυθμη λειτουργία του εργαστηρίου.

Θα ήθελα να αναφερθώ και να ευχαριστήσω όλο το υπόλοιπο έμψυχο δυναμικό του εργαστηρίου, με τους οποίους η καθημερινή συνεργασία και συνύπαρξη ήταν άψογη. Ιδιαιτέρως πρέπει να ευχαριστήσω τους φίλους και συνεργάτες Χρήστο Παυλάτο, Μαρίνο Σαμψών, Γιάννη Ρυακιωτάκη και Βαγγέλη Κούκη για την ερευνητική τους βοήθεια στους χώρους του εργαστηρίου αλλά και για τις ώρες που περάσαμε διασκεδάζοντας εκτός (ή και εντός) εργαστηρίου.

Νιώθω ακόμη την ανάγκη να ευχαριστήσω την οικογένειά μου για την υποστήριξή τους όλα αυτά τα χρόνια, ψυχολογική και υλική, χωρίς την οποία δε θα είχα τη δυνατότητα να βρίσκομαι σήμερα σε αυτή τη θέση. Τέλος, ένα μεγάλο ευχαριστώ στους κοντινούς μου ανθρώπους, που βοήθησαν ο καθένας με το δικό του τρόπο.

Κλείνοντας, θα ήταν παράληψη να μην ευχαριστήσω το Κοινωφελές Ίδρυμα Αλέξανδρος Σ. Ωνάσης για την οικονομική στήριξή του κατά τη διάρκεια της εκπόνησης της διατριβής.

Αλέξανδρος Κ. Δημόπουλος
Αθήνα, Απρίλιος 2009

Κεφάλαιο 1

Εισαγωγή

Σκοπό της συγκεκριμένης διατριβής αποτελεί η υλοποίηση σε επαναπρογραμματιζόμενο υλικό (FPGA) κατάλληλων αλγορίθμων που προσθέτουν "ευφυΐα" σε ένα ενσωματωμένο σύστημα καθώς και η δημιουργία συστήματος για την αυτόματη παραγωγή τέτοιων συστημάτων. Θεωρητικό υπόβαθρο της προσπάθειας αυτής, όπως θα αναλυθεί και ακολούθως, αποτελούν οι Γραμματικές Χωρίς Συμφραζόμενα και οι Κατηγορικές Γραμματικές. Στο πλαίσιο αυτό, έχουν αναπτυχθεί και υλοποιηθεί ειδικοί αλγόριθμοι για την συντακτική αναγνώριση προτάσεων, δηλαδή κατά πόσο μια πρόταση ανήκει σε μια συγκεκριμένη γραμματική χωρίς συμφραζόμενα, αλλά και για την αποτίμηση των κατηγορημάτων των αντίστοιχων προτάσεων για την περίπτωση των κατηγορικών γραμματικών. Βάσει των τιμών που προκύπτουν από την αποτίμηση αυτή, μπορούν να ληφθούν αποφάσεις από το ενσωματωμένο σύστημα και επομένως, ορίζοντας κατάλληλους κανόνες και ενέργειες ανάλογα με τις τιμές των κατηγορημάτων, το σύστημα αποκτά την ψευδαίσθηση της ευφυΐας.

Οι Γραμματικές Χωρίς Συμφραζόμενα (Context-Free Grammar - CFG), συνδυάζουν εκφραστική δύναμη και απλότητα. Είναι αρκετά ισχυρές για να περιγράψουν τη σύνταξη γλωσσών προγραμματισμού [1] (σχεδόν όλες οι γλώσσες προγραμματισμού ορίζονται με χρήση γραμματικών χωρίς συμφραζόμενα) αλλά και απλές ώστε να επιτρέπουν τη δημιουργία αποτελεσματικών αλγορίθμων συντακτικής ανάλυσης. Επιπλέον, μέθοδοι που βασίζονται στις συγκεκριμένες γραμματικές έχουν βρει μέχρι σήμερα εφαρμογή σε διάφορα πεδία, όπως η αναγνώριση φωνής, η επεξεργασία πραγματικών γλωσσών ή γλωσσών προγραμματισμού, η ανάλυση εικόνων καθώς και η τεχνητή νοημοσύνη [2], [3], [4], [5], [6], [7], [8]. Το μεγάλο πλήθος δεδομένων προς επεξεργασία καθώς και οι σημερινές απαιτήσεις για αυστηρούς χρονικούς περιορισμούς καθιστούν αναγκαία τη δημιουργία βελτιστοποιημένων συντακτικών αναλυτών για τέτοιες γραμματικές.

Το 1968 ο Knuth [9] επεξέτεινε τις γραμματικές χωρίς συμφραζόμενα επιτρέποντας την προσθήκη ιδιοτήτων σχετικών με τα συμφραζόμενα, παρουσιάζοντας τις κατη-

γορικές γραμματικές. Η προσθήκη σημασιολογικών κανόνων στους ήδη υπάρχοντες συντακτικούς κανόνες των γραμματικών χωρίς συμφραζόμενα, αύξησε τις εκφραστικές τους ικανότητες και προσέφερε πλεονεκτήματα που οδήγησαν στη χρησιμοποίησή τους στους τομείς του προσδιορισμού, της ανάλυσης και της μετάφρασης γλωσσών. Οι κατηγορικές γραμματικές βρήκαν εφαρμογή πρωτίστως στον τομέα των γλωσσών προγραμματισμού [1] αλλά επιπλέον ήταν βολικές για πεδία όπως η τεχνητή νοημοσύνη [10], [11], η αναγνώριση προτύπων [12] ή ακόμη και η βιοϊατρική [13].
Ανεξαρτήτως του πεδίου εφαρμογής, με τις κατηγορικές γραμματικές η γνώση αναπαρίσταται με συντακτικούς και σημασιολογικούς συμβολισμούς. Για το πρώτο μέρος, το συντακτικό, ένας συντακτικός αναλυτής αναλαμβάνει τη συντακτική αναγνώριση και την κατασκευή του συντακτικού δέντρου. Για το δεύτερο μέρος, εκείνο της διάσχισης του συντακτικού δέντρου και της αποτίμησης των κατηγορημάτων, χρειάζεται ένας αποτιμητής κατηγορημάτων. Συνεπώς, η συνολική διαδικασία της αποτίμησης μιας κατηγορικής γραμματικής μπορεί να χωρισθεί σε δύο υποτμήματα, το συντακτικό και το σημασιολογικό.
Σχετικά με το συντακτικό κομμάτι, η απόδοση ενός συντακτικού αναλυτή επηρεάζεται από δύο βασικούς παράγοντες:

1. Τον αλγόριθμο που ακολουθείται.
2. Την αρχιτεκτονική υλοποίησης, δηλαδή ακολουθιακή ή παράλληλη, σε υλικό ή σε λογισμικό.

Αναφορικά με τον πρώτο παράγοντα, πολλές τροποποιήσεις [14], [15] και βελτιώσεις μέσω παραλληλοποίησης [16] έχουν προταθεί για τους κλασικούς πλέον αλγορίθμους των Cocke-Younger-Kasami (CYK) [17], [18] και του Earley [19]. Μια πιο κοντινή ματιά στους παραπάνω αλγορίθμους, αποδεικνύει ότι έχουν ισχυρές ομοιότητες [14], καθώς πρόκειται για δυναμικές διαδικασίες με χρονική πολυπλοκότητα $O(n^3|G|)$, όπου n είναι το μήκος της πρότασης εισόδου και $|G|$ το μέγεθος της γραμματικής. Βέβαια, ο αλγόριθμος CYK απαιτεί η γραμματική να είναι σε μορφή CNF (Chomsky Normal Form). Όμως, κάθε γραμματική χωρίς συμφραζόμενα μπορεί να μετατραπεί σε CNF μορφή, αλλά όχι χωρίς τίμημα: για μια γραμματική μεγέθους $|G|$, θα χρειαστεί χρόνος $O(|G|^2)$ [20], [21], [22]. Βέβαια, ο μετασχηματισμός αυτός γίνεται μια μόνο φορά, οπότε φαινομενικά δεν επηρεάζει την απόδοση του αναλυτή. Παρόλα αυτά, εάν η νέα γραμματική G' μετά το μετασχηματισμό είναι μεγαλύτερη από την G (στην χειρότερη περίπτωση $|G'| = |G|^2$), η απόδοση του αναλυτή θα επηρεαστεί αφού εξαρτάται από το μέγεθος της γραμματικής ($|G'|$).
Αναφορικά με το δεύτερο παράγοντα, οι Chiang & Fu [16] και Cheng & Fu [23] παρουσίασαν υλοποιήσεις που έκαναν χρήση συστοιχειών VLSI για την υλοποίηση των παραπάνω αλγορίθμων, ενώ οι Ibbara et al. [24] και Ra et al. [25] παρουσίασαν παράλληλες υλοποιήσεις λογισμικού που εκτελούνται σε παράλληλα μηχανήματα. Ο προσα-

νατολισμός σε προσεγγίσεις υλικού αναζωογονήθηκε με την παρουσίαση σε FPGA του αλγορίθμου των CYK (Ciressan [26] και Bordim et al. [27]) και του αλγορίθμου του Earley (Pavlatos et al [28]). Ωστόσο, οι μέχρι τώρα προτεινόμενες αρχιτεκτονικές είτε αποτυγχάνουν να εκμεταλλευτούν στο έπακρο τη διαθέσιμη παραλληλοποίηση των αλγορίθμων ή απαιτούν υπερβολικούς αποθηκευτικούς πόρους. Η υλοποίηση σε λογισμικό, εκτελεί μέρη του αλγορίθμου συντακτικής ανάλυσης ακολουθιακά και έτσι αποτυγχάνει να πετύχει τη μέγιστη δυνατή επιτάχυνση. Στην υλοποίηση σε υλικό, οι υπάρχουσες μεθοδολογίες έρχονται αντιμέτωπες με την πολυπλοκότητα που επιβάλλεται από τους χρησιμοποιούμενους τελεστές των αλγορίθμων, κάτι που οδηγεί στην αυξημένη ανάγκη για χώρο. Προκειμένου να ελαττωθεί η πολυπλοκότητα, οι περισσότερες υλοποιήσεις σε υλικό επιλέγουν τον αλγόριθμο CYK, του οποίου οι βασικές πράξεις είναι πολύ πιο απλές από αυτές του Earley.

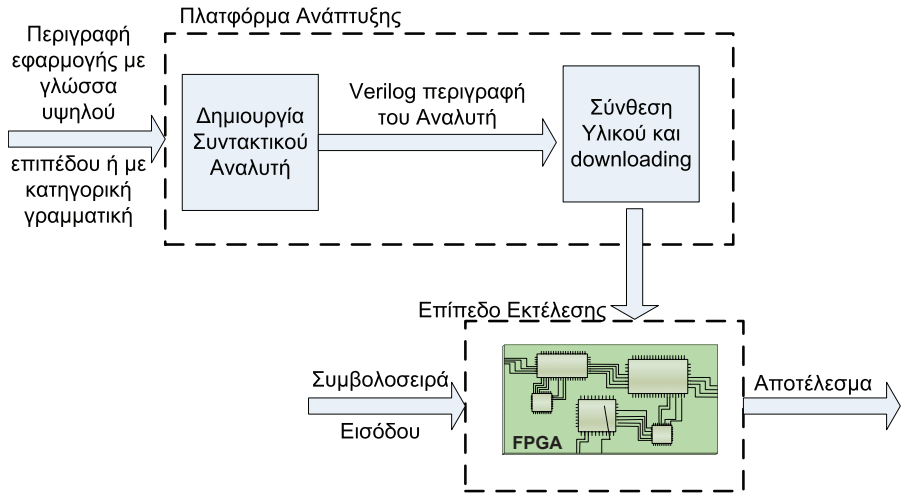
Σχετικά με το σημασιολογικό μέρος, υπάρχουν διάφορες προσεγγίσεις σε λογισμικό, που πέραν της συντακτικής ανάλυσης προχωρούν και στην αποτίμηση των κατηγορημάτων. Μια δημοφιλής προσέγγιση είναι το Yacc [29], που παράγει έναν συντακτικό αναλυτή βασιζόμενο σε μια περιγραφή της γραμματικής σε μορφή παρόμοια με BNF. Όπως θα εξηγηθεί όμως στα επόμενα κεφάλαια, οι παραγόμενοι αναλυτές παρουσιάζουν κάποιους περιορισμούς ως προς τις αποδεκτές γραμματικές (δεν επιτρέπουν διφορούμενες γραμματικές) αλλά και ως προς τα υποστηριζόμενα κατηγορήματα, αφού ακολουθείται η περιορισμένη περίπτωση των S-attributed κατηγορικών γραμματικών [7]. Άλλη μια παρεμφερής και πιο πρόσφατη προσέγγιση είναι το Bison [30], ένα εργαλείο που παράγει συντακτικούς αναλυτές γενικής χρήσης, ουσιαστικά μετατρέποντας την περιγραφή μιας γραμματικής σε ένα πρόγραμμα σε C που εκτελεί τη συντακτική ανάλυση. Το Bison είναι βασισμένο στην L-attributed προσέγγιση των κατηγορικών γραμματικών προσφέροντας αναλυτές με περισσότερες δυνατότητες από τους αντίστοιχους του Yacc. Άλλη μια προσέγγιση είναι το Eli [31], ένας συνδυασμός εργαλείων [32] που υλοποιούν διάφορες στρατηγικές κατασκευής μεταγλωττιστών σε ένα συγκεκριμένο προγραμματιστικό περιβάλλον. Για την αποτίμηση των κατηγορημάτων, το Eli κάνει χρήση της LIDO [33], μιας γλώσσας για τον προσδιορισμό των υπολογισμών σε συντακτικά δέντρα, που υποστηρίζει κάθε μορφής κατηγορήματα. Τέλος, στο πανεπιστήμιο της Ουτρέχτης έχει υλοποιηθεί το εργαλείο UUAG [34] το οποίο μπορεί να αποτιμήσει τα κατηγορήματα σε ένα δοσμένο συντακτικό δέντρο χωρίς όμως να έχει τη δυνατότητα της συντακτικής ανάλυσης.

Πέραν του λογισμικού, έχουν παρουσιαστεί προσεγγίσεις και σε υλικό, παλαιότερα σε ASIC ενώ πιο πρόσφατα σε FPGA. Στην εργασία [35] παρουσιάστηκε μια υλοποίηση για γραμματικές προτύπων (pattern grammars) σε VLSI, στην οποία όμως τα κατηγορήματα ήταν σταθερά και μόνο για τη συγκεκριμένη γραμματική ενώ λόγω της φύσης του VLSI η υλοποίηση ήταν για συγκεκριμένο μέγεθος συμβολοσειράς εισόδου. Το 2005 προτάθηκε από τους Panagopoulos et al [36] ένας μικροεπεξεργαστής

RISC για αποτίμηση κατηγορημάτων με τη βοήθεια FPGA. Ήταν η πρώτη προσπάθεια για σχεδίαση ενός εξειδικευμένου επεξεργαστή για την αποτίμηση κατηγορημάτων, προκειμένου να αξιοποιηθούν τα πλεονεκτήματά του στην ενσωμάτωση της γνώσης σε υπολογιστικά συστήματα. Ο συντακτικός αναλυτής μπορούσε να καθοδηγηθεί από τη σημασιολογία, αυξάνοντας την απόδοση και αποτρέποντας την υπερβολική κατανάλωση μνήμης για την αποθήκευση όλων των πιθανών συντακτικών δέντρων, ενώ παράλληλα παρέμενε μη ντετερμινιστικός, υπολογίζοντας όλες τις πιθανές λύσεις. Όμως, η συγκεκριμένη υλοποίηση βασιζόταν στον αλγόριθμο του σειριακού αναλυτή του Floyd [37], ο οποίος δεν είναι εξίσου αποδοτικός με τους αλγορίθμους των Earley [19] και CYK [18]. Επιπλέον, ουσιαστικά η αποτίμηση των κατηγορημάτων γινόταν σε έναν επεκταμένο επεξεργαστή RISC υλοποιημένο σε ένα FPGA, δηλαδή έμοιαζε με υλοποίηση σε λογισμικό η οποία εκτελείτο σε έναν επεξεργαστή με πολύ χαμηλότερη συχνότητα.

1.1 Συμβολή

Τελικό στόχο της προσπάθειας αποτελεί η κατασκευή ενός συστήματος στο ίδιο τσιπ (System on a Chip - SoC) που θα είναι ικανό να εκτελέσει τόσο την συντακτική όσο και την σημασιολογική ανάλυση. Μάλιστα, το σύστημα θα είναι τέτοιας μορφής που να δημιουργείται αυτομάτως, με τη βοήθεια κατάλληλης πλατφόρμας ανάπτυξης (βλέπε σχήμα 1.1), με διαδικασία περιγραφής που θα διαφέρει από την κλασική μέχρι στιγμής. Πιο συγκεκριμένα, αντί να περιγράφεται το ενσωματωμένο σύστημα με τις κλασικές γλώσσες περιγραφής υλικού, θα περιγράφεται σε υψηλότερο δηλωτικό επίπεδο με τη χρήση του συμβολισμού των κατηγορικών γραμματικών.
Η υλοποίηση των κατηγορικών γραμματικών, βασίστηκε στην παράλληλη υλοποίηση του αλγορίθμου του Earley, του ταχύτερου αλγορίθμου συντακτικής αναγνώρισης στη βιβλιογραφία. Θα δοκιμαστούν μάλιστα δύο διαφορετικές υλοποιήσεις, η πρώτη εκ των οποίων, η πιο κλασική, αποτελεί μεταφορά του παράλληλου αλγορίθμου από το λογισμικό σε υλικό, ενώ η δεύτερη κάνει χρήση αποκλειστικά συνδυαστικής λογικής, έχοντας μικρότερες απαιτήσεις χώρου αλλά και χρονικές απαιτήσεις. Όσον αφορά στο σημασιολογικό μέρος των κατηγορικών γραμματικών, είναι απαραίτητη η ύπαρξη κάποιας μονάδας που να εκτελεί τις αναγκαίες πράξεις. Κατά τη διάρκεια της συγκεκριμένης εργασίας, δοκιμάσθηκαν διαφορετικές αρχιτεκτονικές για τη συντακτική ανάλυση και τον υπολογισμό των κατηγορημάτων. Προφανώς η αρχιτεκτονική θα πρέπει να παραμείνει παραμετροποιήσιμη και να επιτρέπεται η αυτόματη παραγωγή, βάσει της κατηγορικής γραμματικής, τόσο των μονάδων που χρειάζονται για τη συντακτική αναγνώριση, όσο και των μονάδων της σημασιολογικής αποτίμησης. Σε κάθε στάδιο θα πρέπει να αναζητηθούν εφαρμογές στις οποίες η επιτάχυνση της υλοποίησης θα φανεί πραγματικά χρήσιμη. Υποψήφιες εφαρμογές μπορούν να βρεθούν



Σχήμα 1.1: Προτεινόμενο σχήμα Ενσωματωμένου Συστήματος.

στον τομέα των γλωσσών προγραμματισμού όπως η γλώσσα XML, στην οποία η γρήγορη μεταγλώττιση και αποτίμηση κατηγορημάτων θα βοηθήσει σημαντικά το έργο των εξυπηρετητών που έχουν να αποκριθούν ταυτόχρονα σε αιτήματα πολυάριθμων χρηστών, αλλά και η γλώσσα Java. Επιπλέον, μπορούν να αναζητηθούν και βιοϊατρικές εφαρμογές όπως η αναγνώριση του ηλεκτροκαρδιογραφήματος, αλλά και του γονιδιώματος RNA. Τέλος, δεν πρέπει να αποκλειστούν τα προγράμματα λογικού προγραμματισμού αφού όπως θα παρουσιαστεί αργότερα, υπάρχει πλήρης αναλογία μεταξύ των λογικών προγραμμάτων και των κατηγορικών γραμματικών.
Συνοπτικά, η πορεία που ακολουθήθηκε μπορεί να σκιαγραφηθεί στα ακόλουθα:

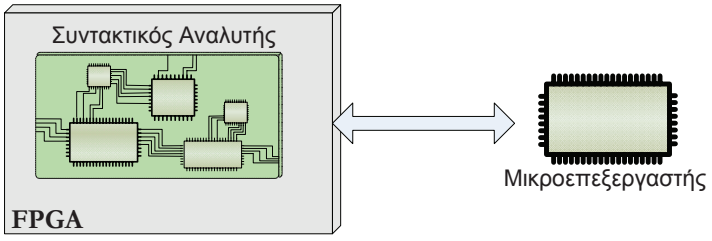
- Αρχικά, έγινε μια πρώτη προσπάθεια για ενσωμάτωση κατηγορημάτων σε έναν συντακτικό αναλυτή. Ο αναλυτής που χρησιμοποιήθηκε βασίσθηκε στην υλοποίηση των Pavlatos et al [28], αλλά επεκτάθηκε προκειμένου να είναι σε θέση να χειριστεί και το σημασιολογικό κομμάτι των γραμματικών. Επιπλέον, για την αποτίμηση των κατηγορημάτων έγινε χρήση εξωτερικού επεξεργαστή. Αποτέλεσμα της συγκεκριμένης εργασίας ήταν οι ανακοινώσεις [C1] και [C4].
- Στη συνέχεια έγινε μια προσπάθεια για την αντικατάσταση του εξωτερικού επεξεργαστή.
  - Σε πρώτη φάση δοκιμάστηκε η λύση εσωτερικού μικροελεγκτή (PicoBlaze). Το συγκεκριμένο σύστημα έδωσε την ανακοίνωση [C5].
  - Ακολούθως δοκιμάστηκε η χρήση εσωτερικού soft core επεξεργαστή (MicroBlaze).
  - Σε επόμενο βήμα, έγινε μια προσπάθεια για απαλοιφή της ανάγκης ύπαρξης επεξεργαστή/μικροελεγκτή για την αποτίμηση και αντικατάστασής τους από εξειδικευμένα λογικά κυκλώματα, με την ανάπτυξη αρχιτεκτονικών

βασισμένων σε στοίβες (stacks). Μάλιστα, η χρήση εξειδικευμένου υλικού για την αποτίμηση των κατηγορημάτων έδωσε την ανακοίνωση [C11].

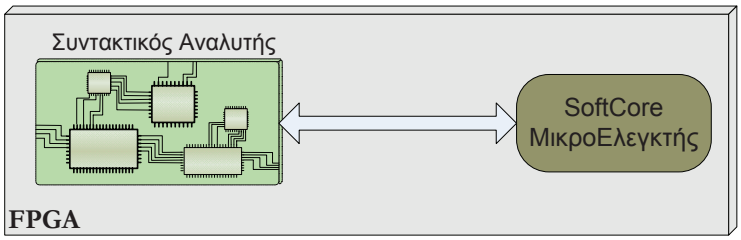
- Στην προσπάθεια να επιταχυνθεί περαιτέρω η διαδικασία συντακτικής αναγνώρισης, μια νέα υλοποίηση του παράλληλου αλγορίθμου δοκιμάστηκε, η οποία στηρίζεται αποκλειστικά σε συνδυαστικά κυκλώματα. Από τον αναλυτή προέκυψαν η δημοσίευση [J1] καθώς και η ανακοίνωση [C6].

- Τέλος, αναζητήθηκε επέκταση του προηγούμενου συνδυαστικού αναλυτή, ώστε να γίνει δυνατός ο συγκερασμός του με την αρχιτεκτονική των στοιβών παράγοντας ένα ενιαίο σύστημα. Η νέα αυτή αρχιτεκτονική έδωσε τις ανακοινώσεις [C9], [C10], [C12], [C13], την υπό κρίση δημοσίευση [UR.J2], την υπό κρίση ανακοίνωση [UR.C14] και τις υπό προετοιμασία [UP.J3, UP.J4].

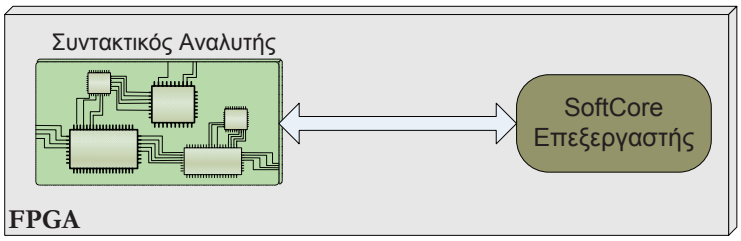
Οι παραπάνω διαδοχικές προσεγγίσεις φαίνονται παραστατικά στο σχήμα 1.2.
Πρέπει να τονιστεί ότι σε κάθε στάδιο της έρευνας, επιζητείται η δυνατότητα αυτοματοποίησης του παραγόμενου συστήματος. Ο χρήστης παρέχει τη γραμματική και με αυτόματο τρόπο παράγεται ο αναλυτής ή και ο αποτιμητής για την περίπτωση των κατηγορικών γραμματικών, ο οποίος προγραμματίζει κατάλληλα το FPGA. Ακόμη πιο γενικά, ο χρήστης μπορεί να περιγράψει την εφαρμογή σε μια γλώσσα υψηλού επιπέδου, όπως η PROLOG και να γίνει αυτόματα η μετατροπή σε κατηγορική γραμματική. Συνεπώς και για το λόγο αυτό, όλες οι αρχιτεκτονικές που υλοποιούνται ακολουθούν μια γενική παραμετροποιήσιμη λογική ανεξάρτητη από συγκεκριμένα χαρακτηριστικά γραμματικής, δίνοντας τη δυνατότητα στο χρήστη να παράγει γρήγορα και εύκολα τον κώδικα που περιγράφει τα αναγκαία εργαλεία για την εφαρμογή του. Όλα τα παραπάνω, παρουσιάζονται σχηματικά στο σχήμα 1.1 ενώ στο παράρτημα Β δίνονται πληροφορίες για το εργαλείο της αυτοματοποίησης αυτό καθεαυτό.



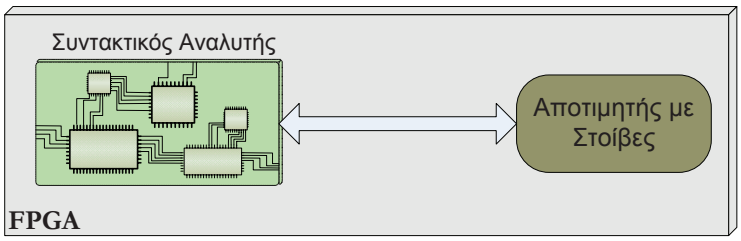
α) Συντακτικός Αναλυτής σε FPGA και αποτιμητής κατηγορημάτων σε εξωτερικό επεξεργαστή



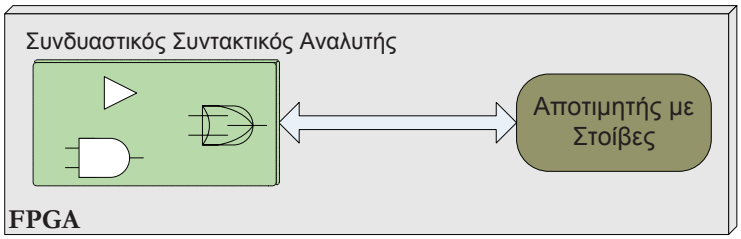
β) Συντακτικός Αναλυτής και αποτιμητής κατηγορημάτων σε εσωτερικό μικροελεγκτή στο ίδιο FPGA



γ) Συντακτικός Αναλυτής και αποτιμητής κατηγορημάτων σε εσωτερικό επεξεργαστή στο ίδιο FPGA



δ) Συντακτικός Αναλυτής και αποτιμητής σε ξεχωριστό Λειτουργικό Τμήμα με αρχιτεκτονική στοιβών στο ίδιο FPGA



ε) Συντακτικός Αναλυτής με συνδυαστική λογική και αποτιμητής σε ξεχωριστό Λειτουργικό Τμήμα με αρχιτεκτονική στοιβών στο ίδιο FPGA

Σχήμα 1.2: Διαδοχικές Προσεγγίσεις Υλοποιήσεων.

1.2 Δημοσιεύσεις

Οι δημοσιεύσεις που προέκυψαν κατά τη διάρκεια της διατριβής είναι οι ακόλουθες:

Σε Περιοδικό

J1. C. Pavlatos, A. Dimopoulos, A. Koulouris, T. Andronikos, I. Panagopoulos and G. Papakonstantinou, "Efficient Reconfigurable Embedded Parsers", *Computer Languages, Systems & Structures*, 35 (2) (2009), 196 - 215, doi= http://dx.doi.org/10.1016/j.cl.2007.08.001

Υπό κρίση Σε Περιοδικό

UR.J2 A. Dimopoulos, C. Pavlatos and G. Papakonstantinou, "AttrAutos: A platform for the automatic generation of attribute evaluation hardware systems", *Computer Languages, Systems & Structures*

Υπό Προετοιμασία

UP.J3 C. Pavlatos, A. Dimopoulos and G. Papakonstantinou, "Hardware Stohastic Context Free Grammars"

UP.J4 A. Dimopoulos, C. Pavlatos and G. Papakonstantinou, "An Embedded System for the ECG Recognition and Analysis"

Σε Συνέδρια

C1. C. Pavlatos, A. Dimopoulos and G. Papakonstantinou, "An Intelligent Embedded System for Control Applications", *Workshop on Modeling and Control of Complex Systems*, Cyprus, July 2005

C2. J. Stoitsis, S. Golemati, A. C. Dimopoulos and K. S. Nikita, "Analysis and quantification of arterial wall motion from B-mode ultrasound images - comparison of block-matching and optical flow", *The 27th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Shanghai, China, September 2005

C3. C. Pavlatos, A. Dimopoulos, G. Manis and G. Papakonstantinou, "Hardware implementation of PAN & TOMPKINS QRS detection algorithm", *Proceedings of the 3rd European Medical & Biological Engineering Conference - EMBEC'05, IFMBE Proceedings*, Prague, Czech Republic, November 2005

C4. C. Pavlatos, A. Dimopoulos, and G. Papakonstantinou, "An embedded system for the electrocardiogram recognition", *Proceedings of the 3rd European Medical*

& Biological Engineering Conference - EMBEC'05, IFMBE Proceedings , Prague, Czech Republic, November 2005

C5. A, Dimopoulos, C, Pavlatos, I. Panagopoulos and G. Papakonstantinou, "An Efficient Hardware Implementation for AI Applications", *Lecture Notes in Computer Science*, vol.3955, pp. 35-45, April 2006

C6. A. Koulouris, T. Andronikos, C. Pavlatos, A. Dimopoulos,I. Panagopoulos and G. Papakonstantinou, "Efficient Signal Processing using Syntactic Pattern Recognition Methods", *Proceedings of the Eighth IASTED International Conference on SIGNAL AND IMAGE PROCESSING*, Honolulu, Hawaii, USA, August 2006

C7. I. Panagopoulos, A. Dimopoulos, G. Manis and G. Papakonstantinou, "AMPLE: Automatic MaPping of aLgorithms for Embedded systems", *11th Panhellenic Conference on Informatics (PCI 2007)*, Patras, Greece, May 2007

C8. I. Panagopoulos, C. Pavlatos, A. Dimopoulos and George Papakonstantinou, "Hardware Solution of a First-Order Diophantine Equation", *The 8th Hellenic European Research on Computer Mathematics & its Applications Conference (HERCMA 07)*, Athens, Greece, September 2007

C9. C. Pavlatos, A. Dimopoulos and G. Papakonstantinou, "Hardware Natural Language Interface", *4th IFIP Conference on Artificial Intelligence Applications & Innovations (AIAI)*, Athens, Greece, September 2007

C10. A. C Dimopoulos, C. Pavlatos and G. Papakonstantinou, "Hardware Embedded System on a Chip for the Normal ECG Recognition", *14th Nordic Baltic Conference on Biomedical Engineering and Medical Physics (NBC-2008)*, Riga, Latvia, June 2008

C11. A.C. Dimopoulos, C.Pavlatos, P.Karanasou and George Papakonstantinou, "A generic platform for the SoC implementation of grammar-based applications", *The* 16^{th} *IFIP/ IEEE International Conference on Very Large Scale Integration (VLSI-SoC 2008)*, Rhodes, Greece, October 2008

C12. A. C. Dimopoulos, C.Pavlatos and George Papakonstantinou, "TELIOS: A Tool for the Automatic Generation of Logic Programming Machines", *The* 5^{th} *IFIP Conference on Artificial Intelligence Applications & Innovations (AIAI 2009)*, Thessaloniki, Greece, April, 2009

C13. C. Pavlatos, A. C. Dimopoulos and G. Papakonstantinou, "A Formal Method for Rapid SoC Prototyping", 20^{th} *IEEE/IFIP International Symposium on Rapid System Prototyping (RSP 2009)*, Paris, France, June, 2009

Υπό κρίση Σε Συνέδρια

UR.C14. C. Pavlatos, A. C. Dimopoulos and G. Papakonstantinou, "A hardware multi-pass attribute grammar evaluator", *The 9th Hellenic European Research on Computer Mathematics & its Applications Conference (HERCMA 09)*, Athens, Greece, September 2009

Κεφάλαιο 2

Βασικές Έννοιες

Όπως αναφέρθηκε και στο προηγούμενο κεφάλαιο, θεωρητικό υπόβαθρο της συγκεκριμένης προσπάθειας αποτελούν οι κατηγορικές γραμματικές. Στα ακόλουθα, θα οριστούν οι βασικές έννοιες σχετικά με γραμματικές και μεθόδους συντακτικής και σημασιολογικής ανάλυσης αυτών, καθώς και όλες οι βασικές έννοιες που διέπουν την συγκεκριμένη εργασία.

2.1 Γραμματικές

Προκειμένου να επικοινωνήσουν οι άνθρωποι, από τα πιο απλά καθημερινά θέματα μέχρι τα πιο σύνθετα επιστημονικά, έχουν ανάγκη από ένα μέσο, μια γλώσσα. Εφόσον, δύο άτομα μιλούν κοινή γλώσσα μπορούν να επικοινωνήσουν και να συνεννοηθούν. Πώς ορίζεται όμως μια γλώσσα; Πώς μπορεί κανείς να διαπιστώσει εάν μια πρόταση ανήκει σε μια γλώσσα ή ακόμη πιο γενικά, ποιες είναι οι προτάσεις που αποτελούν μια γλώσσα; Το 1956 ο Chomsky [38] έδωσε επιστημονικούς ορισμούς για την έννοια της γραμματικής που αποτελεί κάθε γλώσσα. Συγκεκριμένα, όρισε ότι κάθε γραμματική αποτελείται από την εξής τετράδα:

- Ένα πεπερασμένο σύνολο τερματικών συμβόλων, έστω T.
- Ένα πεπερασμένο σύνολο μη τερματικών συμβόλων, έστω N.
- Ένα πεπερασμένο σύνολο κανόνων παραγωγής, που εμπεριέχουν ακολουθίες τερματικών ή/και μη τερματικών στοιχείων έστω R. Οι κανόνες αυτοί ακολουθούν τη μορφή $\alpha \rightarrow \beta$, που δείχνει ότι το σύμβολο β μπορεί να παραχθεί από το σύμβολο α.
- Ένα αρχικό σύμβολο, έστω S.

Τα τερματικά σύμβολα αποτελούν το αλφάβητο της γραμματικής, μέσω του οποίου μπορεί να δημιουργηθεί κάθε συμβολοσειρά που ανήκει στη γραμματική. Για να παραχθεί μια συμβολοσειρά, ξεκινά η παραγωγή από το αρχικό σύμβολο και με τη βοήθεια των κανόνων παραγωγής, αντικαθίστανται μη τερματικά σύμβολα με άλλα σύμβολα,

όπως ορίζεται από τους κανόνες. Το σύνολο όλων των παραγόμενων συμβολοσειρών, καλείται γλώσσα επί του συγκεκριμένου αλφαβήτου.
Οι γραμματικές και κατ' επέκταση οι γλώσσες που παράγονται από αυτές, έχουν κατηγοριοποιηθεί από τον Chomsky στην ακόλουθη ιεραρχία (ιεραρχία Chomsky):

- Γραμματικές Τύπου 0. Εδώ ανήκουν όλες οι γραμματικές.
- Γραμματικές Τύπου 1 ή Γραμματικές Με Συμφραζόμενα (Context-Sensitive Grammars). Οι γραμματικές που ανήκουν σε αυτή την κατηγορία, έχουν κανόνες της μορφής $\alpha A\beta \rightarrow \alpha\gamma\beta$, όπου το A είναι μη τερματικό σύμβολο και τα α, β, γ είναι συμβολοσειρές τερματικών και μη τερματικών συμβόλων. Τα α και β μπορεί να είναι κενά αλλά το γ όχι. Ο κανόνας $A \rightarrow \epsilon$ επιτρέπεται, εφόσον το S δεν εμφανίζεται στο δεξιό μέρος κανενός κανόνα, όπου ϵ είναι η κενή συμβολοσειρά.
- Γραμματικές Τύπου 2 ή Γραμματικές Χωρίς Συμφραζόμενα (Context-Free Grammars). Εδώ, οι κανόνες παραγωγής είναι της μορφής $A \rightarrow \gamma$, όπου A είναι μη τερματικό σύμβολο και γ μια συμβολοσειρά από τερματικά και μη τερματικά σύμβολα. Και εδώ επιτρέπεται ο κανόνας $A \rightarrow \epsilon$, αν και αποδεικνύεται ότι κάθε γραμματική χωρίς συμφραζόμενα με κανόνα τέτοιας μορφής μπορεί να μετασχηματιστεί σε μια ισοδύναμη γραμματική χωρίς συμφραζόμενα, στην οποία να μην υπάρχουν καθόλου κανόνες με το κενό στο δεξιό μέλος τους.
- Γραμματικές Τύπου 3 ή Κανονικές Γραμματικές (Regular Grammars). Στις γραμματικές αυτής της κατηγορίας, στο αριστερό μέλος του κανόνα μπορεί να υπάρχει μόνο ένα σύμβολο, μη τερματικό, και στο δεξί μέρος ένα τερματικό σύμβολο και πιθανώς ένα μη τερματικό σύμβολο. Και εδώ, επιτρέπονται κανόνες της μορφής $A \rightarrow \epsilon$.

Σημειώνεται, ότι όπως προκύπτει και από τα παραπάνω, κάθε κανονική γραμματική είναι και γραμματική χωρίς συμφραζόμενα και αντίστοιχα κάθε γραμματική χωρίς συμφραζόμενα είναι και γραμματική με συμφραζόμενα. Όπως έχει αναφερθεί και πρωτύτερα, οι γραμματικές που θα μας απασχολήσουν είναι εκείνες του τύπου 2, οι γραμματικές χωρίς συμφραζόμενα.

2.1.1 Γραμματικές Χωρίς Συμφραζόμενα

Οι περισσότερες γλώσσες προγραμματισμού, ορίζονται με τη βοήθεια των γραμματικών χωρίς συμφραζόμενα. Σύμφωνα με τον ορισμό του Chomsky, τα αριστερά μέλη των κανόνων παραγωγής, αποτελούνται από ένα μόνο μη τερματικό σύμβολο. Σε κάθε βήμα μιας παραγωγής, αντικαθίσταται ένα μη τερματικό σύμβολο με τη συμβολοσειρά που υπάρχει στο δεξιό μέλος κάποιου κανόνα παραγωγής. Στην περίπτωση, όμως, που υπάρχουν περισσότερα του ενός μη τερματικά σύμβολα, τότε ανάλογα με

τη σειρά που θα ακολουθηθεί κατά την αντικατάσταση, μπορούν να παραχθούν διαφορετικές συμβολοσειρές. Για το λόγο αυτό, στο υπόλοιπο μέρος της εργασίας, σε κάθε βήμα παραγωγής, θα αντικαθίσταται το αριστερότερο μη τερματικό σύμβολο του δεξιού μέλους. Ακολουθεί ένα παράδειγμα μιας απλής γραμματικής χωρίς συμφραζόμενα:

Παράδειγμα 1 *Έστω η γραμματική* $G_1 = \{T, N, P, S\}$*, όπου:*

- $T = \{\alpha,\ *,\ +\}$
- $N = \{E,\ T,\ P\}$
- $P = \{S \to E,$
  $E \to T + E,$
  $E \to T,$
  $T \to P * T,$
  $T \to P,$
  $P \to \alpha\}$

Η συγκεκριμένη γραμματική μπορεί να παράγει συμβολοσειρές της μορφής $\alpha + \alpha$*,* $\alpha * \alpha$*,* $\alpha * \alpha + \alpha$*, ...*
Τα βήματα για την παραγωγή της συμβολοσειράς $\alpha * \alpha + \alpha$ *είναι:*
$S \Longrightarrow E \Longrightarrow T + E \Longrightarrow P * T + E \Longrightarrow \alpha * T + E \Longrightarrow \alpha * P + E \Longrightarrow \alpha * \alpha + E \Longrightarrow$
$\alpha * \alpha + T \Longrightarrow \alpha * \alpha + P \Longrightarrow \alpha * \alpha + \alpha$

2.1.2 Κατηγορικές Γραμματικές

Οι γραμματικές χωρίς συμφραζόμενα κατέχουν εξέχουσα θέση στην διαδικασία περιγραφής γλωσσών προγραμματισμού. Οι συγκεκριμένες γραμματικές επιτρέπουν τον ορισμό της σύνταξης των γλωσσών προγραμματισμού. Ωστόσο, η περιγραφή μιας γλώσσας προγραμματισμού δεν περιορίζεται μόνο στον ορισμό της σύνταξής της. Για παράδειγμα, ο περιορισμός ότι σε κάθε πρόγραμμα θα χρησιμοποιούνται μόνο μεταβλητές που έχουν ήδη οριστεί, δεν μπορεί να εκφραστεί με γραμματικές χωρίς συμφραζόμενα. Μια λύση θα ήταν η χρήση γραμματικών με συμφραζόμενα αλλά αυτό θα δυσκόλευε την διαδικασία της μεταγλώττισης. Για το λόγο αυτό, επεκτείνονται οι γραμματικές χωρίς συμφραζόμενα με την προσθήκη κατηγορημάτων για τα σύμβολα της γραμματικής. Έτσι, προκύπτουν οι κατηγορικές γραμματικές που επιτρέπουν τόσο την συντακτική αναγνώριση των γλωσσών χωρίς συμφραζόμενα αλλά επιπλέον επιτρέπουν και την εκτέλεση υπολογισμών σχετικών με αυτές.
Οι κατηγορικές γραμματικές ορίστηκαν από τον Knuth [9] το 1968. Η προσθήκη κατηγορημάτων και σημασιολογικών κανόνων στις γραμματικές χωρίς συμφραζόμενα, βελτίωσε τις εκφραστικές δυνατότητες και ανήγαγε τις κατηγορικές γραμματικές σε ένα πολύ χρήσιμο εργαλείο για μια πλειάδα εφαρμογών, όπως η τεχνητή νοημοσύνη

(Artificial Intelligence), η αναγνώριση προτύπων (pattern recognition), η δημιουργία μεταγλωττιστών (compiler construction), η επεξεργασία κειμένου κ.α. Μια κατηγορική γραμματική, βασίζεται σε μια γραμματική χωρίς συμφραζόμενα και ορίζεται από την ακόλουθη τριάδα $\{G,\ A,\ SR\}$, όπου :

- G είναι μια γραμματική χωρίς συμφραζόμενα.
- A είναι ένα πεπερασμένο σύνολο κατηγορημάτων (ιδιοτήτων) που σχετίζονται με κάθε σύμβολο της γραμματικής G.
- SR είναι ένα σύνολο από σημασιολογικούς κανόνες που ορίζουν τα κατηγορήματα για κάθε σύμβολο σε κάθε κανόνα παραγωγής της γραμματικής G, σε σχέση με κατηγορήματα άλλων συμβόλων του ίδιου κανόνα.

Τα κατηγορήματα χωρίζονται σε δύο μεγάλες κατηγορίες, στα συντιθέμενα (synthesized) και στα κληρονομούμενα (inherited):

- Συντιθέμενο ονομάζεται ένα κατηγορούμενο, εάν η τιμή του σε κάποιο κόμβο του συντακτικού δέντρου υπολογίζεται από τις τιμές άλλων κατηγορημάτων που ανήκουν στους απογόνους του κόμβου.
- Κληρονομούμενο ονομάζεται ένα κατηγορούμενο, εάν η τιμή του σε κάποιο κόμβο του συντακτικού δέντρου υπολογίζεται από τις τιμές άλλων κατηγορημάτων που ανήκουν στους προγόνους του κόμβου ή ίδιου επιπέδου κόμβους (αδέλφια).

Ανάλογα με τα υποστηριζόμενα κατηγορήματα και των τρόπο αποτίμησής τους, δύο σημαντικά σύνολα των κατηγορικών γραμματικών μπορούν να οριστούν, οι S-Attributed και οι L-Attributed κατηγορικές γραμματικές [7]:

- S-Attributed ονομάζεται μια κατηγορική γραμματική που έχει μόνο συντιθέμενα κατηγορήματα και επιτρέπουν την αποτίμηση των κατηγορημάτων με μια διάσχιση του συντακτικού δέντρου, από αριστερά προς τα δεξιά. Δηλαδή, για την αποτίμηση ενός κατηγορήματος πιθανώς χρειάζονται τιμές άλλων κατηγορημάτων, τα οποία όμως βρίσκονται αριστερότερά του στον συντακτικό κανόνα και άρα έχουν ήδη υπολογιστεί
- L-Attributed ονομάζονται οι κατηγορικές γραμματικές, οι οποίες επιτρέπουν επιπλέον και κληρονομούμενα κατηγορήματα των οποίων η αποτίμηση μπορεί να γίνει με μια διάσχιση του συντακτικού δέντρου, από αριστερά προς τα δεξιά. Δηλαδή, οι L-Attributed κατηγορικές γραμματικές είναι υπερσύνολο των S-Attributed.

Αν και μπορούν να θεωρηθούν πολύ απλοϊκές οι δύο αυτές κατηγορίες, έχουν αποδειχθεί εξαιρετικά πρακτικές για την υλοποίηση ενός μεγάλου αριθμού συντακτικών αναλυτών. Για παράδειγμα, το YACC [29], ένα πασίγνωστο βοηθητικό εργαλείο για

την συντακτική ανάλυση γλωσσών, ακολουθεί την S-attributed προσέγγιση. Επιπλέον, ένα ακόμη κίνητρο για την χρησιμοποίηση των συγκεκριμένων συνόλων είναι ότι οι σημασιολογικοί κανόνες επιτρέπουν την αποτίμησή τους με μια μόνο διάσχιση του συντακτικού δέντρου.
Η γραμματική χωρίς συμφραζόμενα του παραδείγματος 1 μπορεί να επεκταθεί στην ακόλουθη κατηγορική γραμματική G_2, η οποία παράλληλα με την αναγνώριση των μαθηματικών εκφράσεων, θα υπολογίζει και την τιμή τους.

Παράδειγμα 2 *Σε κάθε σύμβολο της G_1 αντιστοιχίζεται ένα συντιθέμενο κατηγόρημα με όνομα res, η τιμή του οποίου είναι φυσικός αριθμός. Η τιμή του κατηγορήματος δεν είναι τίποτε άλλο από την τιμή της μαθηματικής μέχρι στιγμής υποέκφρασης. Ειδικά για το τερματικό σύμβολο α, του αποδίδεται η αυθαίρετη τιμή 5. Συνεπώς, η κατηγορική γραμματική θα έχει τη μορφή:*

G_2

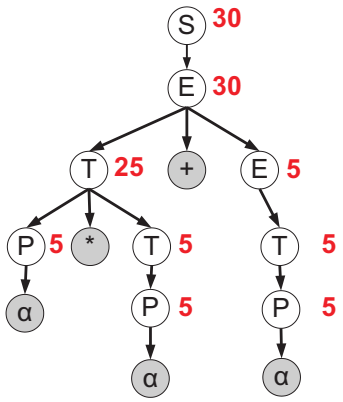
Συντακτικοί Κανόνες	*Σημασιολογικοί Κανόνες*
$S \to E$	$S_{res} = E_{res}$
$E \to T + E$	$E_{res} = T_{res} + E_{res}$
$E \to T$	$E_{res} = T_{res}$
$T \to P * T$	$T_{res} = P_{res} * T_{res}$
$T \to P$	$T_{res} = P_{res}$
$P \to \alpha$	$P_{res} = 5$

Πράγματι, με τη βοήθεια της συγκεκριμένης γραμματικής υπολογίζονται αριθμητικές πράξης της μορφής $\alpha + \alpha = 10$, $\alpha * \alpha = 25$, $\alpha * \alpha + \alpha = 30$, ...

2.2 Συντακτικοί Αναλυτές

Μέχρι τώρα έχει εξεταστεί πώς μπορεί, με τη βοήθεια των κανόνων παραγωγής μιας γραμματικής, να παραχθεί μια απλή συμβολοσειρά. Το πρόβλημα αυτό δεν είναι πάντοτε απλό και ούτε μπορεί να βρεθεί λύση "με το μάτι". Για το λόγο αυτό, έχουν αναπτυχθεί ειδικοί αλγόριθμοι που εξετάζουν τις συμβολοσειρές. Έτσι, ένας συντακτικός αναλυτής (parser) μπορεί να οριστεί ως ένα πρόγραμμα που δεχόμενο ως είσοδο μια συμβολοσειρά, μπορεί να αποφανθεί εάν ανήκει σε μια συγκεκριμένη γραμματική. Στην πιο απλή του μορφή, ένας τέτοιος αλγόριθμος απαντά απλώς με ένα "ναι" ή ένα "όχι" και ονομάζεται αναγνωριστής (recognizer). Στις πιο σύνθετες μορφές, εκτός από την δυαδική απάντηση παράγεται και το συντακτικό δέντρο (parse tree) για την αντίστοιχη συμβολοσειρά εισόδου. Το συντακτικό δέντρο έχει ρίζα το αρχικό (μη τερματικό) σύμβολο της γραμματικής, φύλλα τα τερματικά σύμβολα

της συμβολοσειράς και κόμβους τους κανόνες παραγωγής που χρησιμοποιούνται. Το συντακτικό δέντρο για τη συμβολοσειρά $\alpha*\alpha+\alpha$ του παραδείγματος 1 αναπαρίσταται στο σχήμα 2.1, ενώ δίπλα σε κάθε κόμβο υπάρχει η τιμή του κατηγορήματος res, όπως υπολογίζεται με βάση τους σημασιολογικούς κανόνες της γραμματικής G_2 του παραδείγματος 2. Στην επόμενη παράγραφο 2.2.1, παρουσιάζεται πιο αναλυτικά ο αλγόριθμος μεταγλώττισης του Earley [19] ενώ στην μεθεπόμενη 2.2.2 μια παράλληλη υλοποίηση αυτού, από τους Chiang & Fu [16], πάνω στην οποία έχουν βασιστεί οι προτεινόμενες υλοποιήσεις.



Σχήμα 2.1: Συντακτικό δέντρο για τη συμβολοσειρά $\alpha * \alpha + \alpha$.

2.2.1 Αλγόριθμος του Earley

Το 1970 ο Jay Earley παρουσίασε έναν αλγόριθμο συντακτικής αναγνώρισης [19], από πάνω προς τα κάτω, για γραμματικές χωρίς συμφραζόμενα. Ο αλγόριθμος εξετάζει μια συμβολοσειρά εισόδου και την αποδέχεται ή την απορρίπτει σε χρόνο n^3, όπου n είναι το μέγεθος της συμβολοσειράς εισόδου. Η βασική καινοτομία του αλγόριθμου του Earley είναι η εισαγωγή ενός νέου συμβόλου "•", της τελείας, η οποία δεν ανήκει στην γραμματική. Ο ρόλος της τελείας είναι να χωρίζει το δεξιό μέλος του κανόνα παραγωγής σε δύο μέρη, το αριστερά και το δεξιά της τελείας. Για το μέρος του κανόνα που βρίσκεται αριστερά της τελείας, έχει ήδη εξακριβωθεί ότι όντως μπορεί να παράγει τη συμβολοσειρά εισόδου που έχει εξεταστεί μέχρι στιγμής ενώ το μέρος που βρίσκεται στα δεξιά της τελείας πρέπει να εξεταστεί. Έτσι, όλοι οι κανόνες της γραμματικής μετατρέπονται σε κανόνες με τελεία (dotted-rules). Τελικό συμπέρασμα για το εάν ανήκει μια συμβολοσειρά σε μια γραμματική προκύπτει όταν η τελεία φτάσει στο τέλος του πρώτου κανόνα παραγωγής, εκείνου δηλαδή που έχει για αριστερό μέλος το αρχικό σύμβολο της γραμματικής, το S. Πιο αναλυτικά, ο αλγόριθμος διατρέχει μια συμβολοσειρά εισόδου $\alpha_1\alpha_2\ldots\alpha_n$ από τα αριστερά προς τα δεξιά, και για κάθε σύμβολο α_i που εξετάζεται, ένα αντίστοιχο σύνολο από καταστάσεις κατασκευάζεται,

έστω S_i. Κάθε τέτοιο σύνολο περιέχει τις εξής 4 πληροφορίες $\{r,\ j,\ f,\ h\}$:

- Τον αριθμό r του κανόνα παραγωγής.
- Τη θέση j της τελείας στο δεξιό μέλος του κανόνα παραγωγής.
- Ένα δείκτη f στο σύνολο στο οποίο πρωτοδημιουργήθηκε ο κανόνας.
- Μια συμβολοσειρά h στην οποία αποθηκεύονται τα k επόμενα σύμβολα της συμβολοσειράς εισόδου (k-lookahead string).

Για μια συμβολοσειρά εισόδου μεγέθους n σύμφωνα με τον αλγόριθμο θα κατασκευαστούν $n+1$ σύνολα (από 0 μέχρι n) ενώ το αρχικό (S_0) θα αρχικοποιηθεί κατάλληλα. Σε κάθε κατάσταση μπορεί να εκτελεστεί μια από τις τρεις βασικές διαδικασίες που ορίζονται στον συγκεκριμένο αλγόριθμο:

- Predictor: Η συγκεκριμένη διαδικασία εφαρμόζεται όταν δεξιά της τελείας υπάρχει μη τερματικό σύμβολο. Οδηγεί στην προσθήκη μιας νέας κατάστασης για κάθε κανόνα παραγωγής που έχει στο αριστερό μέλος το συγκεκριμένο μη τερματικό σύμβολο. Στις νέες αυτές καταστάσεις, η τελεία είναι στην αρχή του κανόνα παραγωγής.
- Scanner: Η συγκεκριμένη διαδικασία εφαρμόζεται όταν δεξιά της τελείας υπάρχει τερματικό σύμβολο. Το σύμβολο αυτό συγκρίνεται με το εξεταζόμενο σύμβολο εισόδου a_i και εάν είναι ίδιο, προστίθεται στο επόμενο σύνολο S_{i+1} ο κανόνας αφού μετακινηθεί η τελεία μια θέση δεξιά, δηλώνοντας δηλαδή ότι το συγκεκριμένο τερματικό σύμβολο αναγνωρίστηκε.
- Completer: Η συγκεκριμένη διαδικασία εφαρμόζεται όταν η τελεία έχει φτάσει στο τέλος του κανόνα παραγωγής. Συγκρίνει τη συμβολοσειρά h με τα επόμενα k σύμβολα της συμβολοσειράς εισόδου, δηλαδή με τα $\alpha_{i+1} \ldots \alpha_{i+k}$. Εάν ταυτίζονται, πηγαίνει στο σύνολο που καθορίζεται από το δείκτη f και φέρνει όλους τους κανόνες για τους οποίους τα συγκρινόμενα σύμβολα ταιριάζουν, αφού προηγουμένως μετακινηθεί η τελεία μια θέση δεξιά.

Για να γίνει πιο κατανοητός ο αλγόριθμος, θα δοθεί ένα παράδειγμα εφαρμογής για την γνωστή γραμματική G_1 και με συμβολοσειρά εισόδου $\alpha * \alpha$ και $k = 1$.

Παράδειγμα 3 *Το αρχικό σύνολο S_0 έχει αρχικοποιηθεί κατάλληλα με όλους τους κανόνες με την τελεία στην πρώτη θέση άρα:*

$$S_0 \begin{cases} P \rightarrow_\bullet a \\ T \rightarrow_\bullet P * T \\ T \rightarrow_\bullet P \\ E \rightarrow_\bullet T \\ E \rightarrow_\bullet T + E \\ S \rightarrow_\bullet E \end{cases}$$

Εφόσον δεξιά της τελείας για τον κανόνα $P \rightarrow_\bullet a$ υπάρχει τερματικό σύμβολο, το οποίο είναι ίδιο με το σύμβολο εισόδου $\alpha_0 = a$ τότε στο σύνολο S_1 θα προστεθεί ο κανόνας $P \rightarrow a_\bullet$ (scanner). Όμως, για τον κανόνα αυτό έχει φτάσει η τελεία στο τέλος και επομένως θα ενεργοποιηθεί ο Completer και θα φέρει τους κανόνες που έχουν δημιουργηθεί στο S_0 και έχουν δεξιά της τελείας το P, αφού τους μετακινήσει την τελεία μια θέση δεξιά. Άρα θα έρθουν οι κανόνες $T \rightarrow P_\bullet$ και $T \rightarrow P_\bullet * T$. Για τον πρώτο από τους δύο αυτούς κανόνες, η τελεία είναι στο τέλος του κανόνα παραγωγής, επομένως με την ίδια διαδικασία θα έρθουν οι κανόνες $E \rightarrow T_\bullet$ και $E \rightarrow T_\bullet + E$. Τέλος, πάλι λόγω του πρώτου από τους νέους αυτούς κανόνες, θα έρθει ο κανόνας $S \rightarrow E_\bullet$. Άρα το σύνολο S_1 θα είναι:

$$S_1 \quad \begin{cases} P \rightarrow a_\bullet \\ T \rightarrow P_\bullet * T \\ T \rightarrow P_\bullet \\ E \rightarrow T_\bullet \\ E \rightarrow T_\bullet + E \\ S \rightarrow E_\bullet \end{cases}$$

Τώρα $\alpha_1 = *$, επομένως λόγω Scanner στον κανόνα $T \rightarrow P_\bullet * T$ θα δημιουργηθεί στο S_2 ο κανόνας $T \rightarrow P *_\bullet T$. Επειδή η τελεία είναι αριστερά από το μη τερματικό σύμβολο T, θα ενεργοποιηθεί ο Predictor και θα φέρει τους κανόνες που έχουν στο αριστερό τους μέλος το T, δηλαδή τους κανόνες $T \rightarrow_\bullet P * T$ και $T \rightarrow_\bullet P$. Πάλι λόγω Predictor θα έρθει και ο κανόνας $P \rightarrow_\bullet a$. Συνεπώς, το σύνολο S_2 θα είναι:

$$S_2 \quad \begin{cases} T \rightarrow P *_\bullet T \\ T \rightarrow_\bullet P * T \\ T \rightarrow_\bullet P \\ P \rightarrow_\bullet a \end{cases}$$

Τέλος, $\alpha_2 = a$ και επομένως λόγω Scanner στον κανόνα $P \rightarrow_\bullet a$ θα δημιουργηθεί στο S_3 ο κανόνας $P \rightarrow a_\bullet$. Όπως και στην περίπτωση του S_1, ο Completer θα ενεργοποιηθεί και θα φέρει τους κανόνες που έχουν δημιουργηθεί στο S_2 και έχουν δεξιά της τελείας το P, αφού τους μετακινήσει την τελεία μια θέση δεξιά. Άρα θα έρθουν οι κανόνες $T \rightarrow P_\bullet$ και $T \rightarrow P_\bullet * T$. Για τον πρώτο από τους δύο αυτούς κανόνες, η τελεία είναι στο τέλος του κανόνα παραγωγής, επομένως με την ίδια διαδικασία θα έρθει ο κανόνας $T \rightarrow P * T_\bullet$. Εδώ η τελεία έχει φτάσει πάλι στο τέλος, άρα από το S_0 θα έρθουν οι κανόνες $E \rightarrow T_\bullet$ και $E \rightarrow T_\bullet + E$. Τέλος, ο πρώτος κανόνας θα φέρει τον κανόνα $S \rightarrow E_\bullet$. Τελικά συνεπώς, το σύνολο S_3 θα

είναι:

$$S_3 \begin{cases} P \to a_\bullet \\ T \to P_\bullet \\ T \to P_\bullet * T \\ T \to P * T_\bullet \\ E \to T_\bullet \\ E \to T_\bullet + E \\ S \to E_\bullet \end{cases}$$

Επομένως, αφού μέσα στο σύνολο S_3 η τελεία έχει φτάσει στο τέλος του κανόνα $S \to E_\bullet$, βγαίνει το συμπέρασμα ότι η συμβολοσειρά $\alpha * \alpha$ ανήκει στην γραμματική G_1.

2.2.2 Αλγόριθμος των Chiang & Fu

Το 1980 προτάθηκε από τους Graham et al [14] μια βελτίωση του αλγορίθμου του Earley, αντικαθιστώντας τις δομές των συνόλων S_i με έναν πίνακα, το συντακτικό πίνακα (Parse Table - PT). Έτσι, τώρα πια το στοιχείο (i, j) του πίνακα PT περιέχει τους κανόνες που ανήκουν στο σύνολο S_j και αρχικά δημιουργήθηκαν στο σύνολο S_i. Από τον πίνακα, μόνον τα στοιχεία που ανήκουν στην κύρια διαγώνια και πάνω χρησιμοποιούνται ($i < j$). Δύο νέοι τελεστές παρουσιάστηκαν, με τους οποίους αντικατέστησαν τις διαδικασίες Scanner, Completer και Predictor. Επεκτείνοντας την βελτιωμένη αυτή έκδοση, οι Chiang & Fu [16] παρουσίασαν μια παράλληλη εκδοχή του αλγορίθμου, που χρησιμοποιεί ένα μόνο τελεστή πια, τον $\otimes$ και καταργεί την ανάγκη του Predictor, αρχικοποιώντας κατάλληλα τα στοιχεία του πίνακα PT που ανήκουν στην κύρια διαγώνιο. Ο αλγόριθμος χωρίς τη λειτουργία Predictor καλείται weakened Earley και όλες οι προτεινόμενες υλοποιήσεις είναι βασισμένες σε αυτόν. Για το λόγο αυτό θα εξεταστεί πιο αναλυτικά ο συγκεκριμένος αλγόριθμος, αλλά πρώτα θα δοθούν ορισμένοι ορισμοί.

Ορισμός 1 *Έστω ένα μη τερματικό σύμβολο A. Τα σύνολα* Predecessors(A), Descendants(A) *και* Predict(A) *ορίζονται ως εξής:*

- Predecessors(A) $= \{B \in N | B \xrightarrow{*} A\}$, *το συγκεκριμένο σύνολο περιέχει όλα τα μη τερματικά σύμβολα που μπορούν να δημιουργήσουν το μη τερματικό σύμβολο A.*
- Descendants(A) $= \{B \in N | A \xrightarrow{*} B\}$, *το συγκεκριμένο σύνολο περιέχει όλα τα μη τερματικά σύμβολα που μπορούν να παραχθούν από το μη τερματικό σύμβολο A.*
- Predict(A) $= \{A \to \gamma_\bullet \delta | A \to_\bullet \gamma\delta \in P$ και $\gamma \xrightarrow{*} \varepsilon\}$, *το συγκεκριμένο σύνολο περιέχει όλους τους κανόνες παραγωγής με τελεία, στους οποίους στο αριστερό μέλος υπάρχει το μη τερματικό σύμβολο A και έχουν την τελεία στην πρώτη*

θέση ή γενικά σε οποιαδήποτε θέση πριν από το πρώτο σύμβολο που δεν μπορεί να παράγει το κενό σύμβολο.

- $\text{Predict(N)} = \bigcup_{A \in N} \text{Predict(A)}$

Ο συμβολισμός $A \xrightarrow{*} \alpha$ *σημαίνει ότι το* α *μπορεί να παραχθεί με καμία ή περισσότερες παραγωγές από το A. Επομένως, το A είναι πάντα μέλος των συνόλων Predecessors(A) και Descendants(A).*

Ορισμός 2 *Έστω Q και R δύο σύνολα από κανόνες παραγωγής με τελεία και Y=Predict(N), τότε ο τελεστής* $\otimes$ *ορίζεται ως εξής:*
$Q \otimes R = \{A \rightarrow \alpha U \beta_{\bullet} \gamma \mid A \rightarrow \alpha_{\bullet} U \beta\gamma \in Q,\ \beta \xrightarrow{*} \lambda \text{ και } U \rightarrow \delta_{\bullet} \in R\}$
και

$$\{B \rightarrow \delta C \xi_{\bullet} \eta \mid \gamma = \lambda,\ B \rightarrow \delta_{\bullet} C \xi \eta \in Y \text{ και } \xi \xrightarrow{*} \lambda,\ C \xrightarrow{*} A\}$$

Εάν το R είναι απλώς ένα τερματικό σύμβολο τότε:
$Q \otimes R = \{A \rightarrow \alpha U \beta_{\bullet} \gamma \mid A \rightarrow \alpha_{\bullet} U \beta\gamma \in Q,\ \beta \xrightarrow{*} \lambda \text{ και } U \in R\}$
και

$$\{B \rightarrow \delta C \xi_{\bullet} \eta \mid \gamma = \lambda,\ B \rightarrow \delta_{\bullet} C \xi \eta \in Y \text{ και } \xi \xrightarrow{*} \lambda,\ C \xrightarrow{*} A\}$$

Χρησιμοποιώντας τη γραμματική G_1 στο παρακάτω παράδειγμα, θα φανεί αναλυτικά πώς δουλεύει ο τελεστής $\otimes$.

Παράδειγμα 4 *Έστω τα σύνολα* $Q=\{E \rightarrow_{\bullet} T,\ T \rightarrow_{\bullet} a\}$, $U=\{T \rightarrow a_{\bullet}\}$ *και Y= Predict(N)* $= \{E \rightarrow_{\bullet} T,\ E \rightarrow_{\bullet} E+T,\ T \rightarrow_{\bullet} a\}$. *Για τα συγκεκριμένα σύνολα ισχύει:*

- $Q \otimes U = \{E \rightarrow T_{\bullet},\ E \rightarrow E_{\bullet} + T\}$
  Σύμφωνα με τον τελεστή $\otimes$, *αρχικά το σύνολο U ελέγχεται για κανόνες με την τελεία στο τέλος* ($T \rightarrow a_{\bullet}$). *Ακολούθως, το σύνολο Q ελέγχεται για κανόνες, στους οποίους το σύμβολο δεξιά της τελείας είναι το αριστερό μέλος των κανόνων παραγωγής που προέκυψαν από την προηγούμενη διαδικασία (T σε αυτή την περίπτωση). Για τους κανόνες που πληρούν τις προϋποθέσεις* ($E \rightarrow_{\bullet} T \in Q$), *μετακινείται η τελεία μια θέση δεξιά και προστίθενται στο σύνολο* $Q \otimes U$ ($E \rightarrow T_{\bullet} \in Q \otimes U$). *Επιπλέον, στην περίπτωση που η τελεία έχει φτάσει στο τέλος του κανόνα, το σύνολο* Z_1=*Predecessors(αριστερό μέλος των προστεθέντων κανόνων) καθορίζεται (μόνο* $E \in Z_1$). *Τέλος, το σύνολο Y ελέγχεται για κανόνες, όπου δεξιά της τελείας υπάρχει μη τερματικό σύμβολο που ανήκει στο* Z_1 ($E \rightarrow_{\bullet} E+T \in Y$). *Αυτοί οι κανόνες προστίθενται στο σύνολο* $Q \otimes U$, *αφότου μετακινηθεί η τελεία μια θέση δεξιά* ($E \rightarrow E_{\bullet} + T \in Q \otimes U$).
- $Q \otimes \alpha = \{T \rightarrow a_{\bullet},\ E \rightarrow T_{\bullet},\ E \rightarrow E_{\bullet} + T\}$
  Σύμφωνα με τον τελεστή $\otimes$, *το σύνολο Q ελέγχεται για κανόνες στους οποίους δεξιά της τελείας υπάρχει το τερματικό σύμβολο a* ($T \rightarrow_{\bullet} a \in Q$). *Οι κανόνες*

αυτοί προστίθενται στο σύνολο Q⊗a αφότου μετακινηθεί η τελεία μια θέση δεξιά (T→a$_\bullet$ ∈ Q⊗a). Επιπλέον, στην περίπτωση που η τελεία έχει φτάσει στο τέλος του κανόνα, το σύνολο Z$_2$=Predecessors(αριστερό μέλος των προστεθέντων κανόνων) καθορίζεται (E, T ∈ Z$_2$). Τέλος, το σύνολο Y ελέγχεται για κανόνες, όπου δεξιά της τελείας υπάρχει μη τερματικό σύμβολο που ανήκει στο Z$_2$ (E→$_\bullet$E+T και E→$_\bullet$T ∈Y). Και αυτοί οι κανόνες προστίθενται στο σύνολο Q⊗a αφού μετακινηθεί η τελεία μια θέση δεξιά (E→T$_\bullet$ και E→E$_\bullet$+T ∈ Q⊗a).

Προφανώς, ένας κανόνας προστίθεται στο σύνολο μόνο εάν δεν υπάρχει ήδη.

Προκειμένου να αποφασιστεί εάν μια συμβολοσειρά $\alpha_1\alpha_2\ldots\alpha_n$ μήκους n ανήκει σε μία γραμματική χωρίς συμφραζόμενα, ένας δισδιάστατος πίνακας μεγέθους $(n+1)\times(n+1)$ δημιουργείται, με στοιχεία $pt(i,j)$ κανόνες παραγωγής με τελεία. Αφού μόνον τα κελιά της διαγωνίου και πάνω χρησιμοποιούνται, χρειάζονται $\frac{(n+1)\times(n+2)}{2}$ υπολογιστικά στοιχεία. Η διαδικασία αναγνώρισης, μετατρέπεται στην ισοδύναμη της συμπλήρωσης των κελιών του πίνακα. Αρχικά, όλα τα κελιά της διαγωνίου αρχικοποιούνται με το σύνολο Y=Predict(N), όπου N είναι το σύνολο των μη τερματικών συμβόλων. Η συμπλήρωση του πίνακα παραλληλοποιείται ως προς το μέγεθος n της συμβολοσειράς εισόδου. Πιο συγκεκριμένα, σε κάθε βήμα εκτέλεσης k $(1 \le k \le n)$ υπολογίζονται τα κελιά $pt(i,j)$ που ανήκουν στην ίδια διαγώνιο (δευτερεύουσα διαγώνιο). Για τον υπολογισμό του κελιού $pt(i,j)$, όλα τα κελιά $pt(i,m)$ και $pt(m,j)$ για τα οποία $(i+1) \le m \le (j-1)$ (βλέπε πίνακα 2.1) πρέπει να έχουν υπολογιστεί. Σε κάθε βήμα εκτέλεσης, κάθε υπολογιστικό στοιχείο υπολογίζει ένα κελί $pt(i,j)$ βάσει της εξίσωσης:

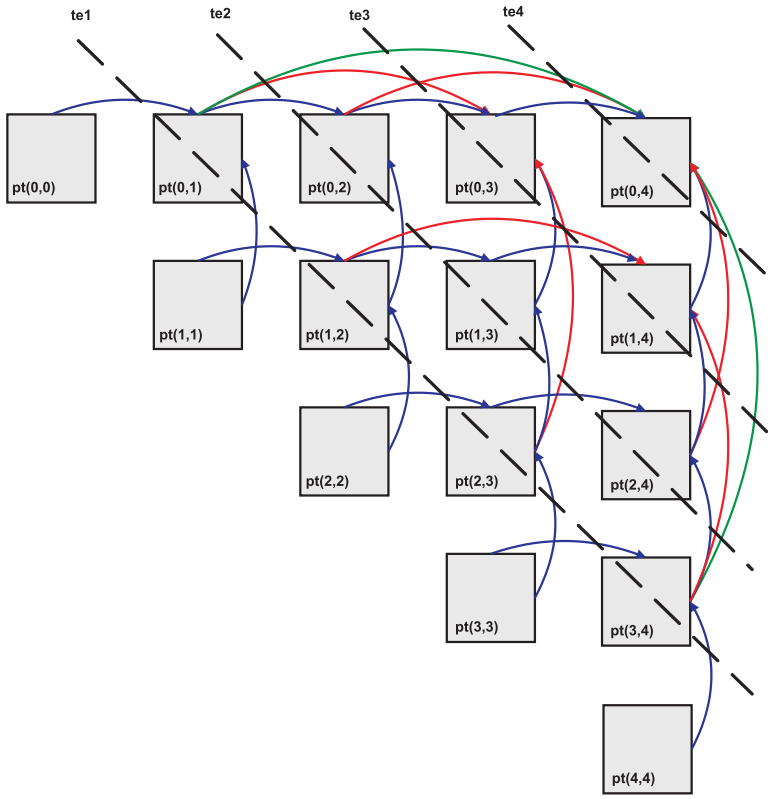
$$pt(i,j) = \begin{cases} pt(i,i+1) \otimes pt(i+1,j) \bigcup \\ pt(i,i+2) \otimes pt(i+2,j) \bigcup \\ \ldots \\ pt(i,j-1) \otimes pt(j-1,j) \bigcup \\ pt(j-1,i) \otimes \alpha_j \end{cases} \tag{2.1}$$

Μετά το πέρας του βήματος εκτέλεσης, ακολουθεί το βήμα επικοινωνίας. Κάθε υπολογιστικό στοιχείο μεταδίδει τα δεδομένα που υπολόγισε στα κελιά που θα τα χρειαστούν ή αντίστροφα λαμβάνει δεδομένα από άλλα κελιά. Αφού συμπληρωθεί όλος ο πίνακας, η ύπαρξη στο κελί $pt(0,n)$ του αρχικού κανόνα που έχει στο αριστερό μέλος του το αρχικό σύμβολο S με την τελεία στο τέλος, ισοδυναμεί με αναγνώριση της συμβολοσειράς εισόδου. Σχηματικά τα παραπάνω φαίνονται στο σχήμα 2.2, όπου η επικοινωνία μεταξύ των κελιών συμβολίζεται με βέλη. Συνοπτικά στον πίνακα 2.1 φαίνονται τα κελιά που υπολογίζονται σε κάθε βήμα, καθώς και τα κελιά που πρέπει να έχουν εκ των προτέρων υπολογιστεί. Σημειώνεται ότι και στις δύο περιπτώσεις η συμβολοσειρά εισόδου έχει μήκος $n = 4$.

Βήμα Εκτέλεσης	Υπολογιζόμενο Κελί	Αναγκαία για τον υπολογισμό Κελιά
1	pt(0,1)	pt(0,0)
	pt(1,2)	pt(1,1)
	pt(2,3)	pt(2,2)
	pt(3,4)	pt(3,3)
2	pt(0,2)	pt(0,1) και pt(1,2)
	pt(1,3)	pt(1,2) και pt(2,3)
	pt(2,4)	pt(2,3) και pt(3,4)
3	pt(0,3)	pt(0,1), pt(0,2), pt(1,3) και pt(2,3)
	pt(1,4)	pt(1,2), pt(1,3), pt(2,4) και pt(3,4)
4	pt(0,4)	pt(0,1), pt(0,2), pt(0,3), pt(1,4), pt(2,4) και pt(3,4)

Πίνακας 2.1: Συνοπτικός Πίνακας Υπολογισμού Κελιών για μήκος συμβολοσειράς εισόδου $n = 4$.

Προκειμένου να γίνουν κατανοητά όλα τα παραπάνω, στο ακόλουθο Παράδειγμα 5 παρουσιάζεται ο πίνακας PT για την γραμματική G_1 και συμβολοσειρά εισόδου $\alpha * \alpha$.



Σχήμα 2.2: Σχηματική Αναπαράσταση της Αρχιτεκτονικής των Fu & Chiang για $n = 4$.

Παράδειγμα 5 *Εφόσον η συμβολοσειρά εισόδου έχει μέγεθος 3, χρειάζεται ένας πίνακας* 4×4. *Όλα τα κελιά που ανήκουν στην κύρια διαγώνιο αρχικοποιούνται με το σύνολο* Y=Predict(N) = $\{S \rightarrow_\bullet E,\ E \rightarrow_\bullet T,\ E \rightarrow_\bullet T+E,\ T \rightarrow_\bullet T,\ T \rightarrow_\bullet P*T,\ P \rightarrow_\bullet a\}$. *Ο συμπληρωμένος πίνακας παρουσιάζεται στον πίνακα 2.2.*
Η ύπαρξη του κανόνα $S \rightarrow E_\bullet$ *στο pt(0,3) σηματοδοτεί την αναγνώριση της συμβολοσειράς εισόδου.*

	α	*	α
S→•E E→•T E→•T+E T→•P T→•P*T P→•a	P→ a• T→ P• T→ $P_\bullet * T$ E→T• E→T•+E S→E•	T→P*•T	T→P*T• E→ T• E→ $T_\bullet + E$ **S→ E•**
	S→•E E→•T E→•T+E T→•P T→•P*T P→•a	∅	∅
		S→•E E→•T E→•T+E T→•P T→•P*T P→•a	P→ a• T→ P• T→ $P_\bullet * T$ E→T• E→T•+E S→E•
			S→•E E→•T E→•T+E T→•P T→•P*T P→•a

Πίνακας 2.2: Συμπληρωμένος πίνακας PT για την συμβολοσειρά εισόδου $\alpha * \alpha$.

Τελειώνοντας την παρουσίαση του συγκεκριμένου αλγορίθμου, θα πρέπει να σημειωθεί ότι η προτεινόμενη παράλληλη υλοποίηση των Chiang & Fu παρουσίαζε κάποιες αδυναμίες αφού:

1. Απαιτεί οι γραμματικές να μην έχουν κανόνες με το κενό στο δεξιό μέλος. Βέβαια, κάθε γραμματική χωρίς συμφραζόμενα με κενό στο δεξιό μέλος μπορεί

να μετατραπεί σε ισοδύναμη χωρίς το κενό, αλλά με τη μετατροπή αυξάνεται δραματικά το πλήθος των κανόνων παραγωγής.

2. Το μέγεθος της γραμματικής παίζει καθοριστικό ρόλο για την ταχύτητα της εκτέλεσης.
3. Δεν γίνεται κατάλληλη εκμετάλλευση της εγγενούς παραλληλίας του τελεστή $\otimes$.
4. Ένα πολύ περίπλοκο πρωτόκολλο επικοινωνίας απαιτείται για την επικοινωνία των κατανεμημένων μονάδων.

2.3 Ενσωματωμένα Συστήματα

Ως Ενσωματωμένο Σύστημα, μπορεί να ορισθεί κάθε υπολογιστικό σύστημα ειδικού σκοπού, το οποίο έχει σχεδιαστεί για κάποια ειδική λειτουργία ή λειτουργίες. Δηλαδή, εν αντιθέσει με τα συστήματα γενικού σκοπού, όπως οι προσωπικοί υπολογιστές, που μπορούν ανάλογα με το πρόγραμμα που εκτελούν να λειτουργούν διαφορετικά κάθε φορά, τα ενσωματωμένα συστήματα έχουν πολύ περιορισμένες επιλογές λειτουργιών. Συνήθως είναι μικρά σε μέγεθος για να αποτελούν υποσύστημα κάποιου μεγαλύτερου συστήματος, χωρίς όμως αυτό να αποτελεί προϋπόθεση. Η εξειδίκευση του υλικού είναι που κάνει ένα ενσωματωμένο σύστημα επιθυμητό. Αντί για την επιλογή ενός συστήματος γενικής χρήσης, το οποίο είναι πιο ακριβό καθώς έχει περισσότερες δυνατότητες και συνήθως πιο αργό, προτιμάται το εξειδικευμένο ενσωματωμένο σύστημα. Ενσωματωμένα συστήματα βρίσκονται παντού στη σημερινή καθημερινότητα, από το πλυντήριο πιάτων και το κινητό τηλέφωνο μέχρι τα συστήματα πέδησης και ασφαλείας του αυτοκινήτου. Στόχος είναι η υλοποίηση ενός συστήματος που επιτελεί μια επιθυμητή λειτουργία πιο γρήγορα και πιο οικονομικά από ένα σύστημα γενικής χρήσης χωρίς να είναι λιγότερο αξιόπιστο και παράλληλα, καταναλώνει λιγότερη ενέργεια και έχει μικρότερο μέγεθος. Η πολυπλοκότητα ποικίλει από συστήματα με έναν απλό μικροελεγκτή μέχρι σύνθετα συστήματα παράλληλων επεξεργαστών.

Από τα πρώτα ενσωματωμένα συστήματα, όπως το Apollo Guidance Computer (1960), που αναπτύχθηκε στο MIT για τη βοήθεια στην πλοήγηση των διαστημοπλοίων των αποστολών Apollo μέχρι τα τελευταία υπερσύγχρονα κινητά η φιλοσοφία σχεδιασμού παραμένει η ίδια. Σχεδιάζονται προκειμένου να εκτελούν επιλεγμένες λειτουργίες με χαμηλό κόστος - οικονομικό, σε χώρο και σε κατανάλωση. Δίνεται ιδιαίτερη προσοχή σε συγκεκριμένες λειτουργίες που απαιτείται να εκτελούνται γρήγορα, ενώ για άλλες όχι. Άλλωστε, πολλές εφαρμογές δεν απαιτούν μεγάλη υπολογιστική ισχύ ή πόρους και αυτό επιτρέπει την ελαχιστοποίηση του κόστους παραγωγής χρησιμοποιώντας έναν αργό - για τα σημερινά δεδομένα - επεξεργαστή και μικρό μέγεθος μνήμης. Παρά την εξειδίκευσή τους και το μέγεθός τους, δεν παύουν να είναι υπολογιστικά συστήματα και άρα ακολουθούν ανάλογες αρχιτεκτονικές. Ενσωματωμένα Συστήματα

μεγάλης κλίμακας μπορούν μάλιστα να έχουν ακριβώς την ίδια μορφή. Για παράδειγμα, εάν λειτουργούν ως δρομολογητές, απαιτούνται μία ή περισσότερες διασυνδέσεις δικτύου, μεγάλη μνήμη και γρήγορη λειτουργία. Μπορούν επίσης να απαιτούν κάποια μορφή διεπαφής με το χρήστη ως μέρος της ενσωματωμένης εφαρμογής και, κατά μιαν έννοια, μπορεί απλά να πρόκειται για συμβατικό υπολογιστή αφιερωμένο σε ένα συγκεκριμένο έργο. Έτσι, από την οπτική του υλικού, πολλά ενσωματωμένα συστήματα υψηλών επιδόσεων δεν είναι και τόσο διαφορετικά από ότι ένα συμβατικό μηχάνημα. Μικρότερα ενσωματωμένα συστήματα χρησιμοποιούν μικροελεγκτές, ενσωματώνοντας μεγάλο μέρος της λειτουργικότητας ενός υπολογιστή σε ένα μόνο τσιπ. Ο μικροελεγκτής έχει τουλάχιστον μια κεντρική μονάδα επεξεργασίας, μια μικρή εσωτερική μνήμη (ROM ή / και RAM) και κάποια μορφή I/O, τα οποία έχουν υλοποιηθεί σαν υποσύστηματα και παρέχουν την αναγκαία πρόσθετη λειτουργικότητα. Οι είσοδοι και έξοδοι του συστήματος μπορεί να είναι αναλογικές ή και ψηφιακές, ανάλογα με τις ανάγκες της εφαρμογής. Όποια αρχιτεκτονική και να ακολουθηθεί όμως στο σχεδιασμό ενός ενσωματωμένου συστήματος, εκείνο με το οποίο θα έρθει αντιμέτωπος ο σχεδιαστής του είναι ο συγκερασμός κατά βάση αντικρουόμενων απαιτήσεων, όπως το κόστος κατασκευής, η κατανάλωση ενέργειας, η υπολογιστική ισχύς, η αξιοπιστία, το φυσικό μέγεθος κ.α. Τέλος, μια ακόμη μεγάλη διαφοροποίηση στα ενσωματωμένα συστήματα, είναι το κατά πόσον πρέπει να αντεπεξέλθουν αυστηρά σε εφαρμογές πραγματικού χρόνου (hard real time embedded system) ή έχουν λιγότερη σημασία οι χρονικοί περιορισμοί (soft real time embedded system). Προφανώς, ένα σύστημα σε ένα οικιακό ψυγείο που δεν θα "αντιληφθεί" αμέσως την αύξηση της θερμοκρασία δεν θα δημιουργήσει την ίδια ζημιά με ένα αντίστοιχο σύστημα σε έναν πυρηνικό αντιδραστήρα.

Γίνεται κατανοητό από τα παραπάνω, ότι η έννοια του ενσωματωμένου συστήματος είναι δύσκολο να οριστεί με ακρίβεια, αφού είναι πολύ ευρεία. Όποια και να είναι όμως η μορφή του συστήματος, σίγουρα τα δομικά του στοιχεία θα ανήκουν σε μια από τις δύο μεγάλες κατηγορίες: σε αυτή του επαναπρογραμματιζόμενου υλικού (PLD, FPGA) ή σε αυτή του μη επαναπρογραμματιζόμενου υλικού (VLSI, ASIC). Στην περίπτωση της συγκεκριμένης εργασίας, η υλοποίηση έχει βασιστεί στην τεχνολογία των FPGA, η οποία θα αναλυθεί ακολούθως.

2.3.1 Field Programmable Gate Arrays

Τα FPGA (Field Programmable Gate Arrays) είναι ψηφιακά ολοκληρωμένα κυκλώματα, που περιέχουν προγραμματιζόμενα λογικά μπλοκ καθώς και επίσης προγραμματιζόμενες διασυνδέσεις μεταξύ αυτών των μπλοκ. Προφανώς, οι συσκευές αυτές μπορούν να προγραμματιστούν για ποικίλες εφαρμογές. Αναλόγως με την τεχνολογία κατασκευής τους, τα FPGA μπορεί να είναι προγραμματιζόμενα μόνον μια φορά ή

επαναπρογραμματιζόμενα. Όπως φαίνεται και από το πρώτο κομμάτι της ονομασίας τους (Field=πεδίο), ο προγραμματισμός τους δεν απαιτεί ειδικές εργαστηριακές συνθήκες αλλά μπορεί να γίνει εύκολα και απλά, γεγονός που έδωσε ώθηση στην ευρεία ανάπτυξή τους.

Η βασική ιδέα των FPGA, ή έστω των προγραμματιζόμενων μπλοκ (Gate arrays), έχει αναπτυχθεί ήδη από τα τέλη της δεκαετίας του 1960 από εταιρείες όπως η IBM και η Fujitsu. Ωστόσο, δεν ήταν ευρείας κατανάλωσης και μόνο μετά τα μέσα του 70 η τεχνολογία CMOS επέτρεψε την όποια διάδοσή τους, αφού το κόστος τους εξακολουθούσε να είναι μεγάλο. Μέχρι τις αρχές του 80 μάλιστα, η επιλογή ανάμεσα σε ASIC (Application Specific Integrated Circuit) και FPGA ήταν μονόδρομος προς την πρώτη επιλογή. Τα ASIC προσφέρουν δυνατότητες εξειδικευμένου υλικού αλλά είναι πολύ ακριβά για μικρή ποσότητα παραγωγής και η διαδικασία σχεδίασής τους εξαιρετικά χρονοβόρα. Επιπλέον, άπαξ και προγραμματιστούν, καμία αλλαγή δεν μπορεί να γίνει στη λειτουργία τους. Ακριβώς αυτές τις "αδυναμίες" ήρθε να ανατρέψει η Xilinx το 1984, δημιουργώντας μια νέα κατηγορία ολοκληρωμένων κυκλωμάτων, τα FPGA. Τα πρώτα αυτά FPGA στηρίζονταν στην τεχνολογία CMOS και χρησιμοποιούσαν μνήμη SRAM για την διαμόρφωσή τους. Κάθε μπλοκ, εμπεριείχε βασικά ένα Lookup Table (LUT) 3 εισόδων, έναν καταχωρητή που λειτουργούσε ως flip-flop ή μανταλωτής (latch) και έναν πολυπλέκτη. Κάθε FPGA αποτελείτο από μεγάλο αριθμό τέτοιων μπλοκ, τα οποία με τη βοήθεια της μνήμης SRAM, κάθε ένα μπλοκ μπορούσε να επιτελεί διαφορετική λειτουργία. Όλα τα μπλοκ αυτά ενώνονταν με κατάλληλες προγραμματιζόμενες διασυνδέσεις, ενώ προφανώς υπήρχαν και τα κατάλληλα σήματα εισόδου/εξόδου (I/O) για την επικοινωνία με τον πραγματικό κόσμο. Σήμερα, η βασική ιδέα παραμένει η ίδια αλλά έχουν αλλάξει η τεχνολογία και τα μεγέθη. Πια τα LUT στα μπλοκ είναι 4 εισόδων, αν και έχουν κατασκευαστεί μέχρι και 6 εισόδων. Οι μεγάλες αλλαγές έχουν γίνει στο τί υπάρχει ενσωματωμένο πια σε ένα FPGA. Πλέον, μπορεί να υπάρχει μνήμη RAM, πολλαπλασιαστές (multipliers) και κυκλώματα για πρόσθεση (adders) αλλά και πιο σύνθετα στοιχεία όπως ακόμη και ένας επεξεργαστής. Μάλιστα υπάρχει επιλογή ανάμεσα στους επεξεργαστές, οι Hard Core και οι Soft Core :

- Οι *Hard Core* επεξεργαστές, έχουν υλοποιηθεί ως ένα ξεχωριστό μπλοκ, το οποίο διασυνδέεται κατάλληλα με τα υπόλοιπα στοιχεία του FPGA. Ο επεξεργαστής μπορεί να βρίσκεται στο ίδιο τσιπ πυριτίου με το υπόλοιπο FPGA ή να βρίσκεται μεν φυσικά στο FPGA αλλά όχι στο ίδιο τσιπ. Μάλιστα, μπορούν να ενσωματωθούν περισσότεροι του ενός επεξεργαστές.
- Οι *Soft Core* επεξεργαστές είναι μπλοκ του FPGA προγραμματισμένα ώστε να λειτουργούν σαν ένας επεξεργαστής. Προφανώς είναι πιο απλοί από τους πραγματικούς (hard wired) επεξεργαστές και λιγότερο ταχείς, αλλά προσφέ-

ρουν το πλεονέκτημα ότι ο χρήστης τους υλοποιεί μόνον εάν τους χρειάζεται η εφαρμογή του και επιπλέον, μπορούν να υλοποιηθούν πολλοί στο ίδιο FPGA.

Ακόμη, ειδική μνεία έχει ληφθεί για το ρολόι, ώστε να μην εμφανίζονται καθυστερήσεις, με την χρήση ειδικών, ταχύτερων διασυνδέσεων σε όλο το κύκλωμα. Ενώ πια το I/O μπορεί να γίνει και από εξειδικευμένα μπλοκ όπως κάρτες δικτύου, USB, κάρτες ήχου, παραμένουν και τα "παραδοσιακά" διακοπτάκια.
Ο όλος προγραμματισμός των κυκλωμάτων αυτών, γίνεται με ειδικές γλώσσες προγραμματισμού, τις γλώσσες περιγραφής υλικού (Hardware Description Languages - HDL) με κυριότερους εκπροσώπους τις Verilog και VHDL. Οι γλώσσες αυτές είναι υψηλού επιπέδου και θυμίζουν έντονα C. Επιτρέπουν την περιγραφή σε διάφορα επίπεδα, από το χαμηλότερο του τρανζίστορ έως το υψηλότερο της περιγραφής συναρτήσεων και μηχανών καταστάσεων (Finite State Machines - FSM). Οι περιγραφές είναι συνήθως ανεξάρτητες του μοντέλου του FPGA, δηλαδή δεν περιορίζονται μόνο σε αυτό, αλλά μπορούν να βρουν εφαρμογή και σε άλλα μοντέλα. Με τον τρόπο αυτό, κομμάτια κώδικα μπορούν να ξαναχρησιμοποιηθούν - μάλιστα για να διευκολυνθεί ο χρήστης πολλοί προμηθευτές FPGA παρέχουν και μια δική τους ποικιλία από έτοιμο κώδικα (intellectual property IP) στους πελάτες τους. Μια πιο εμπεριστατωμένη παρουσίαση των FPGA γίνεται στο Παράρτημα Α.

2.4 Λογικά Προγράμματα, Κατηγορικές Γραμματικές και Ενσωματωμένα Συστήματα

Προσεγγίσεις για την ενσωμάτωση της γνώσης σε υπολογιστικά συστήματα (Knowledge Engineering) έχουν βρει εφαρμογές σε πολλά πεδία, όπως η ιατρική, ο αυτόματος έλεγχος, η τεχνητή νοημοσύνη [39] κτλ. Ιδιαίτερα στο πεδίο του ευφυούς ελέγχου και της ρομποτικής [40], προσεγγίσεις ενσωμάτωσης της γνώσης έχουν χρησιμοποιηθεί, ώστε να βελτιώσουν εφαρμογές ελέγχου με εγγενή ευφυΐα. Οι απαιτήσεις που συνήθως επιβάλλονται για τέτοιες εφαρμογές, όπως η χαμηλή κατανάλωση ενέργειας, οι μικρές διαστάσεις και η απόκριση σε πραγματικό χρόνο, οδηγούν στην ανάγκη για σχεδιασμό εξειδικευμένων ενσωματωμένων συστημάτων. Συνεπώς, η εις πέρας εκμετάλλευση των τεχνικών ενσωμάτωσης της γνώσης είναι μεγίστης σημασίας. Οι δύο πιο σημαντικοί παράγοντες, οι οποίοι επηρεάζουν την αποδοτικότητα τέτοιων σχεδιασμών είναι η προγραμματιστική απλότητα και η απόδοση της τελικής υλοποίησης.
Οι μέθοδοι ενσωμάτωσης της γνώσης έχουν υλοποιηθεί κυρίως με χρήση εργαλείων βασισμένων στο δηλωτικό μοντέλο προγραμματισμού (declarative programming model). Αντίθετα, η φύση του υποστηριζόμενου προγραμματισμού, από τους σημερινούς επεξεργαστές, είναι μόνον του διαδικαστικού μοντέλου προγραμματισμού (procedural programming model). Απόρροια αυτού του γεγονότος, είναι ότι η δημιουργία μιας

βάσης γνώσης με λογικούς κανόνες (λογικός προγραμματισμός) για τους υπάρχοντες επεξεργαστές, δεσμεύεται από τη χρήση ενός μεταγλωττιστή που επιτελεί την μετατροπή του δηλωτικού σε διαδικαστικό, προκειμένου να μπορεί να εκτελεστεί. Ο μηχανισμός που ακολουθείται για την μετατροπή αυτή, επηρεάζει τόσο την πολυπλοκότητα όσο και την ταχύτητα της τελικής υλοποίησης, αφού συνήθως οδηγεί στην προσθήκη σημαντικού μεγέθους επιπλέον κώδικα, υπεύθυνου για την εξομοίωση της δηλωτικής εκτέλεσης. Ακόμη, πάλι με επιπλέον κώδικα, δημιουργείται μια στοίβα λογισμικού για τις λογικές αναγωγές, η φύση της οποίας αυξάνει εκθετικά τις αναφορές I/O στην μνήμη. Συνεπώς, οι υλοποιήσεις λογικών προγραμμάτων σε λογισμικό, στα υπάρχοντα ενσωματωμένα συστήματα, επηρεάζει αρνητικά την απόδοση από πλευράς πολυπλοκότητας και ταχύτητας. Η ύπαρξη επεξεργαστών ικανών να υποστηρίξουν το μοντέλο δηλωτικού προγραμματισμού, θα βελτίωνε σημαντικά την απόδοση και την απλότητα του παραγόμενου κώδικα.

Η πρώτη μηχανή που παρουσιάστηκε για την υλοποίηση λογικών προγραμμάτων (PROLOG) ήταν η Warren Abstract Machine (WAM) [41]. Η WAM είναι μια αφηρημένη αρχιτεκτονική επεξεργαστή με εξειδικευμένο σετ εντολών, ικανή να κάνει λογικές παραγωγές (logic derivation), που απευθύνεται αποκλειστικά στην εκτέλεση λογικών προγραμμάτων. Εκτενείς προσπάθειες έχουν καταβληθεί επίσης προς την κατεύθυνση μηχανών για λογικά προγράμματα με βελτιωμένη απόδοση, σύμφωνα με το όραμα της $5^{ης}$ γενιάς υπολογιστών για πλήθος διασυνδεδεμένων παράλληλων μηχανών για εφαρμογές τεχνητής νοημοσύνης [42], [43]. Ισχυροί επεξεργαστές έχουν παρουσιαστεί να λειτουργούν σε υπολογιστές UMA και NUMA [44], [42], σε μια προσπάθεια για αύξηση της παραλληλοποίησης των δηλωτικών προγραμμάτων, σχεδιασμένων για λογικές μηχανές PROLOG. Αν και η συνολική επιτάχυνση που επιτυγχάνεται, ακολουθώντας τέτοιες προσεγγίσεις, είναι απολύτως ικανοποιητική, το κόστος κατασκευής τέτοιων συστημάτων καθώς και το μέγεθός τους, έχει αποτρέψει την χρήση τους σε εφαρμογές μικρού μεγέθους στο χώρο των ενσωματωμένων συστημάτων.

Η εμφάνιση των ενσωματωμένων συστημάτων παρουσιάζει νέες προκλήσεις και ανάγκες για την υλοποίηση επεξεργαστών βελτιστοποιημένων για εφαρμογές λογικών προγραμμάτων. Αφού ένα ενσωματωμένο σύστημα έχει μικρό φάσμα εφαρμογών και η υπολογιστικής ισχύς του είναι περιορισμένη (ακριβώς όση είναι αναγκαία για την εφαρμογή), προσπάθειες προς την κατεύθυνση βελτίωσης των επιδόσεων βοηθούν στην συνολική απόδοση των υλοποιήσεων. Συνεπώς, η προσπάθεια σχεδιασμού υλικού ικανού να υποστηρίξει το δηλωτικό μοντέλο προγραμματισμού για λογικά προγράμματα, μπορεί να οδηγήσει σε ευφυή ενσωματωμένα συστήματα, τα οποία είναι αξιοσημείωτα πιο αποδοτικά εν συγκρίσει με τα συμβατικά. Σε αυτή την κατεύθυνση έχει κινηθεί η συγκεκριμένη έρευνα [28], [45], [46], [47], όπου δοκιμάστηκαν διαφορετικές αρχιτεκτονικές, διαφορετικοί συντακτικοί αναλυτές καθώς και διαφορετικά πεδία εφαρμογών, όπως θα αναπτυχθεί και στα επόμενα κεφάλαια.

2.4.1 Λογικά Προγράμματα και Κατηγορικές Γραμματικές

Ανάμεσα στο μοντέλο των λογικών προγραμμάτων της PROLOG και στις κατηγορικές γραμματικές υπάρχει μια πλήρης αναλογία [48], [49], [50], [51], [52]. Μάλιστα έχουν αναπτυχθεί το απαραίτητο θεωρητικό υπόβαθρο και τα κατάλληλα εργαλεία, που επιτρέπουν την προσθήκη επιπλέον κατηγορημάτων και τη συνδυασμένη χρήση δηλωτικών και διαδικαστικών χαρακτηριστικών. Η βασική ιδέα για την μετατροπή ενός λογικού προγράμματος σε μια ισοδύναμη κατηγορική γραμματική είναι η εξής: Κάθε κανόνας εξαγωγής συμπερασμάτων (inference rule) στο αρχικό λογικό πρόγραμμα μπορεί να μετατραπεί σε ένα ισοδύναμο συντακτικό κανόνα, αποτελούμενο μόνο από μη τερματικά σύμβολα. Για παράδειγμα, ο κανόνας $R_0(t_{01}, t_{02}, \ldots, t_{0k_0}) \leftarrow R_1(t_{11}, t_{12}, \ldots, t_{1k_1}) \ldots R_m(t_{m1}, t_{m2}, \ldots, t_{mk_1})$ μετατρέπεται στον συντακτικό κανόνα $R_0 \leftarrow R_1 R_2 \ldots R_m\$$, όπου "$" σηματοδοτεί το τέλος του κανόνα. Τα δεδομένα (facts) των κανόνων εξαγωγής συμπερασμάτων μετατρέπονται σε τερματικούς κόμβους του συντακτικού δέντρου, αναφερόμενα στο κενό σύμβολο. Για παράδειγμα, τα δεδομένα $R_g(a, b), R_g(c, d), R_g(e, f)$ μετατρέπονται σε $R_g \leftarrow nil\$$ (τρεις παρεμφερείς συντακτικούς κανόνας, που διαφοροποιούνται από τους σημασιολογικούς κανόνας). Σημειώνεται ότι δεν υπάρχουν άλλα τερματικά σύμβολα πέραν του κενού. Επομένως, η διαδικασία της συντακτικής ανάλυσης έχει εκφυλιστεί. Για κάθε μεταβλητή που υπάρχει στις αρχικές δηλώσεις (predicates), δύο κατηγορήματα αποδίδονται στον αντίστοιχο κόμβο του συντακτικού δέντρου, ένα συντιθέμενο και ένα κληρονομούμενο. Αυτά τα κατηγορήματα βοηθούν στην διαδικασία ενοποίησης (unification process) της μηχανής εξαγωγής συμπερασμάτων (inference engine). Οι κανόνες απόδοσης τιμών στα κατηγορήματα δημιουργούνται βάσει του αρχικού λογικού προγράμματος. Οι τιμές στα κατηγορήματα των φύλλων του συντακτικού δέντρου ανατίθενται από τις σταθερές των δεδομένων του λογικού προγράμματος. Η διαδικασία εξαγωγής συμπερασμάτων εκτελείται κατά τη διάσχιση του δέντρου και μια συνάρτηση καλείται κατά την εισαγωγή ενός κόμβου ή την επίσκεψη σε αυτόν, η οποία συνάρτηση αποτιμά τους αντίστοιχους σημασιολογικούς κανόνες. Οι τιμές των αποτιμήσεων ελέγχονται βάσει σημασιολογικών συνθηκών και καθορίζουν την επιτυχία ή αποτυχία της αποτίμησης με χρήση μιας μέτα-μεταβλητής ($flag$ στον πίνακα 2.3). Γενικά, η προσθήκη σημασιολογικών κανόνων που καθορίζουν τη δημιουργία και τη μορφή του συντακτικού δέντρου οδηγούν στη δημιουργία ενός πλήρως εκφυλισμένου συντακτικού αναλυτή οδηγούμενου από τη σημασιολογία (fully degenerated semantically driven parser), ο οποίος μπορεί να γίνει πολύ αποδοτικός για την περίπτωση του λογικού προγραμματισμού.

Προκειμένου να εμπεδωθεί καλύτερα η αναφερόμενη μεθοδολογία μετασχηματισμού, παρουσιάζεται το επόμενο εκπαιδευτικό παράδειγμα μετατροπής ενός λογικού προγράμματος (γενεαλογικό δέντρο) στην ισοδύναμη κατηγορική γραμματική.

Παράδειγμα 6 *Έστω η βάση γνώσης γενεαλογικού δέντρου που έχει τα ακόλουθα χαρακτηριστικά και αναπαρίσταται στην πρώτη στήλη του πίνακα 2.3:*

- *Σκοπός : εύρεση απογόνων ή προγόνων*
- *Κανόνες:*
  - *Εάν ο Z είναι γονέας του X και ο Z είναι απόγονος του Y τότε ο X είναι απόγονος του Y.*
  - *Εάν ο Y είναι γονέας του X τότε ο X είναι απόγονος του Y.*
- *Δεδομένα:*
  - *Ο j είναι γονέας του b.*

Ανεπίσημος ορισμός της Βάσης Γνώσης	Ισοδύναμοι συντακτικοί και σημασιολογικοί κανόνες της Κατηγορικής Γραμματικής
Goal(X,Y) ← Successor(X,Y)	**Goal** ← **Successor**$ Successor.ia_1=Goal.ia_1; Goal.sa_1=Successor.sa_2;
Successor(X,Y) ← Parent(Z,X) and Successor(Z,Y)	*Successor*$_1$ ← **Parent** *Successor*$_2$$ Parent.ia_2=Successor[1].ia_1; Successor[2].ia_1=Parent.sa_1;
Successor(X,Y) ← Parent(Y,X)	**Successor** ← **Parent**$ Successor.sa_2=Parent.sa_1;

Parent(j,b)	**Parent**←$
	if ((Parent.ia_1!=nil) AND (Parent.ia_1!='j'))
	THEN
	flag=0; else Parent.sa_1='j';
	if ((Parent.ia_2!=nil) AND Parent.ia_2!='b'))
	THEN
	flag=0; else Parent.sa_2='b';

Parent(j,l)	**Parent**←$
	if((Parent.ia_1!=nil) AND (Parent.ia_1!='j'))
	THEN
	flag=0; else Parent.sa_1='j';
	if ((Parent.ia_2!=nil) AND Parent.ia_2!='l'))
	THEN
	flag=0; else Parent.sa_2='b';

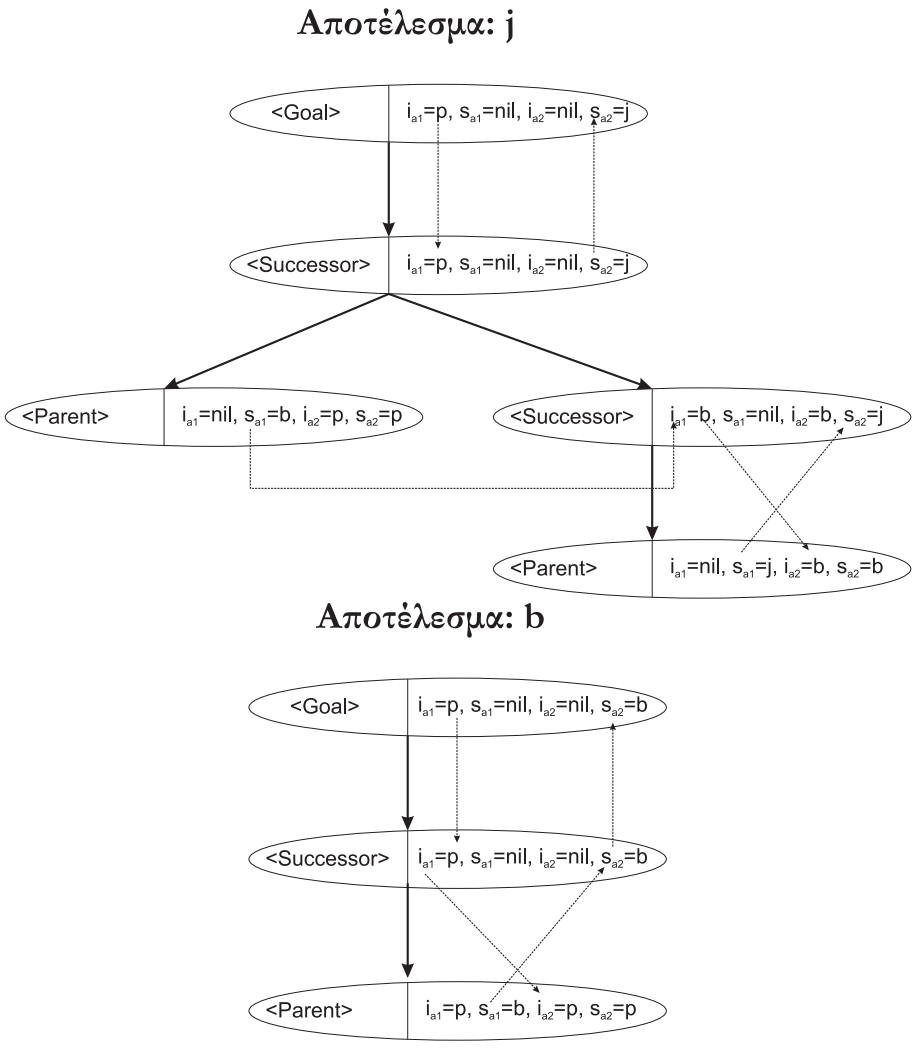
Parent (b,a)	...
Parent (b,p)	...

Πίνακας 2.3: Ισοδύναμη αναπαράσταση σε Κατηγορική Γραμματική της βάσης γνώσης του προβλήματος "απογόνου".

- *Ο j είναι γονέας του l.*
- *Ο b είναι γονέας του a.*
- *Ο b είναι γονέας του p.*

- *Ερώτημα : "Το p είναι διάδοχος ποιανού?".*

Οι συντακτικοί κανόνες της ισοδύναμης κατηγορικής γραμματικής παρουσιάζονται επίσης στον πίνακα 2.3, στην δεύτερη στήλη, μαζί με τους σημασιολογικούς κανόνες και τις συνθήκες που χρησιμοποιούνται για την εξαγωγή συμπεράσματος. Σύμφωνα με τον ακολουθούμενο συμβολισμό, σε κάθε μη τερματικό σύμβολο αντιστοιχίζονται μέχρι n συντιθέμενα κατηγορήματα (sa_1 - sa_n) και μέχρι n κληρονομούμενα (ia_1 - ia_n).
Η συγκεκριμένη ερώτηση έχει δύο απαντήσεις, τους "j" και "b". Τα αντίστοιχα συντακτικά δέντρα μαζί με τα κατηγορήματα παρουσιάζονται στο σχήμα 2.3.



Σχήμα 2.3: Συντακτικά Δέντρα για το απλό παράδειγμα του "διαδόχου".

Κεφάλαιο 3

Αποτίμηση Κατηγορημάτων Με Χρήση Συντακτικού Αναλυτή Και Εξωτερικού Επεξεργαστή

Όπως εξηγήθηκε και στο προηγούμενο κεφάλαιο, το μοντέλο στο οποίο θα στηριχθεί η όλη προσπάθεια είναι αυτό των κατηγορικών γραμματικών. Συνεπώς, είναι αναγκαία η δημιουργία ενός συστήματος ικανού να αναγνωρίσει συντακτικά μία συμβολοσειρά εισόδου και επιπλέον να υπολογίσει τις τιμές των κατηγορημάτων. Ακολούθως, θα παρουσιαστεί η πρώτη προσπάθεια για αποτίμηση κατηγορημάτων με τη βοήθεια συντακτικού αναλυτή.

3.1 Επέκταση Συντακτικού Αναλυτή σε Αποτιμητή Κατηγορικών Γραμματικών

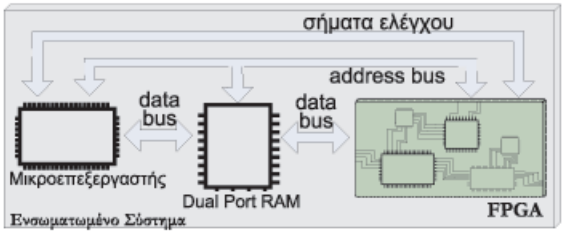
Η συγκεκριμένη υλοποίηση βασίστηκε στο συντακτικού αναλυτή των Pavlatos et al [28]. Ασφαλώς, ο συγκεκριμένος αναλυτής χρησιμεύει μόνο για την αποτίμηση του συντακτικού μέρους των κατηγορικών γραμματικών και συνεπώς, για το σημασιολογικό κομμάτι έπρεπε να επεκταθεί. Ο συγκεκριμένος συντακτικός αναλυτής υλοποιεί μια βελτιωμένη εκδοχή του παραλλήλου αλγορίθμου του Earley, όπως παρουσιάστηκε στην παράγραφο 2.2.2. Συγκεκριμένα, έχει τις ακόλουθες βελτιώσεις:

1. Το πλήθος των αναγκαίων υπολογιστικών στοιχείων έχει ελαττωθεί κατά έναν παράγοντα $\frac{n+1}{2}$.

2. Έχει βελτιωθεί σημαντικά το πρωτόκολλο επικοινωνίας, αφού έχει εξαλειφθεί η ανάγκη για ανταλλαγή πληροφοριών σε κελιά που ανήκουν στην ίδια στήλη του συντακτικού πίνακα PT.

3. Το μέγεθος των δεδομένων αναπαράστασης (data representation) έχει ελαττωθεί.

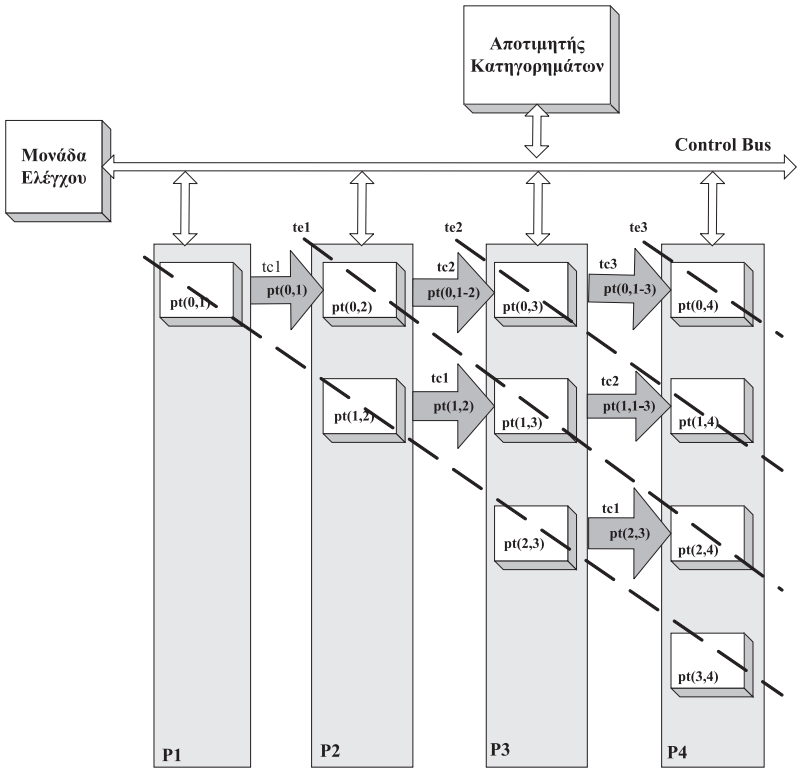
Βάσει λοιπόν του συγκεκριμένου αναλυτή, υλοποιήθηκε ένα σύστημα στο οποίο η σημασιολογική επεξεργασία, δηλαδή η αποτίμηση των κατηγορημάτων, θα εκτελείται από ένα εξωτερικό μικροεπεξεργαστή ενώ ο παράλληλος αλγόριθμος θα εκτελείται σε ένα FPGA. Για την επικοινωνία των δύο αυτών δομικών μονάδων πέραν των συνηθισμένων τεχνικών συσχεδίασης υλικού/λογισμικού, χρησιμοποιείται μια μνήμη RAM με διπλές εισόδους (Dual Port RAM) προκειμένου να επιτυγχάνεται η ανταλλαγή πληροφοριών. Η προτεινόμενη αρχιτεκτονική φαίνεται στο σχήμα 3.1.



Σχήμα 3.1: Αφαιρετική απεικόνιση του ευφυούς ενσωματωμένου συστήματος.

Οι συνηθισμένες μέθοδοι [53], [54] αποτίμησης των κατηγορικών γραμματικών βασίζονται στην κατασκευή του συντακτικού δέντρου. Κατόπιν το δέντρο διατρέχεται μια ή περισσότερες φορές, ανάλογα με τη φύση της γραμματικής, προκειμένου να αποτιμηθούν τα κατηγορήματα. Ο διαχωρισμός της όλης εργασίας σε δύο ξεχωριστές διαδικασίες κάνει την εκτέλεση πιο αργή και άρα ανεπαρκή. Στα [55], [56] ο διαχωρισμός καταργείται αλλά η συντακτική αναγνώριση γίνεται ακολουθιακά και μόνον για γραμματικές με συντιθέμενα κατηγορήματα.
Στην προτεινόμενη υλοποίηση, η συντακτική αναγνώριση και η σημασιολογική αποτίμηση γίνονται παράλληλα προσφέροντας την δυνατότητα, ελέγχοντας συγκεκριμένους κανόνες, για έγκαιρη ανίχνευση συντακτικών λαθών, οδηγώντας στην ταχύτερη απόρριψη μέρους ή όλου του συντακτικού δέντρου. Καθοδηγείται συνεπώς σημασιολογικά η διαδικασία της συντακτικής αναγνώρισης [57]. Επιπλέον, η περιγραφική δύναμη της γραμματικής μπορεί να επεκταθεί, προκειμένου να υποστηρίζει και χαρακτηριστικά με συμφραζόμενα (context-sensitive). Κύριος στόχος είναι η επιτάχυνση ευφυών εφαρμογών, οι οποίες βασίζονται σε λογικό προγραμματισμό. Για την επίτευξη του στόχου, τροποποιήθηκε ο συντακτικός αναλυτής [28] προκειμένου να ενσωματωθεί η δυνατότητα της σημασιολογικής αποτίμησης μέσα στην διαδικασία συντακτικής αναγνώρισης. Σε κάθε στήλη του πίνακα PT αντιστοιχεί πια από ένα υπολογιστικό στοιχείο (P_i). Κάθε υπολογιστικό στοιχείο υπολογίζει ένα κελί $pt(i,j)$ σε κάθε βήμα εκτέλεσης ενώ κατά το επόμενο βήμα υπολογίζει το κελί που ανήκει στην ίδια στήλη αλλά είναι μια σειρά πιο πάνω, δηλαδή το $pt(i-1,j)$. Με τον τρόπο αυτό, έχει

ελαττωθεί ο απαραίτητος αριθμός υπολογιστικών στοιχείων σε μόνο n, ή κατά ένα λόγο $\frac{n+1}{2} = \frac{\frac{(n+1)n}{2}}{n}$. Επιπλέον, έχει εξαλειφθεί η ανάγκη για κατακόρυφη επικοινωνία. Τέλος, ένα επιπλέον υπολογιστικό στοιχείο χρειάζεται για να κατευθύνει την όλη διαδικασία (Μονάδα Ελέγχου) και ένα για την αποτίμηση των κατηγορημάτων (Αποτιμητής Κατηγορημάτων). Συνεπώς, η προτεινόμενη αρχιτεκτονική είναι αυτή που απεικονίζεται στο σχήμα 3.2. Επιπλέον, αναπτύχθηκε κατάλληλη μεθοδολογία για την διάσχιση του συντακτικού δέντρου, που επιτρέπει την ταυτόχρονη αποτίμηση συντακτικών και σημασιολογικών κανόνων. Η όλη συνδυασμένη διαδικασία απεικονίζεται στο διάγραμμα ροής του σχήματος 3.3. Μετά το πέρας κάθε βήματος εκτέλεσης k, το πρώτο από αριστερά κάθε φορά υπολογιστικό στοιχείο, υπολογίζει το κελί που βρίσκεται στην δική του στήλη και στην πρώτη γραμμή του πίνακα. Μετά από αυτό τον υπολογισμό, το στοιχείο σταματάει τους υπολογισμούς, αφού δεν έχει κάτι άλλο να υπολογίσει. Κατά το επόμενο βήμα εκτέλεσης $k+1$, το συγκεκριμένο στοιχείο μεταδίδει όλα τα δεδομένα από τα κελιά που υπολόγισε στον αποτιμητή κατηγορημάτων. Τα δεδομένα που μεταφέρονται, περιέχουν τους κανόνες μέσω των οποίων μπορεί να κατασκευαστεί η συμβολοσειρά εισόδου που έχει εξεταστεί μέχρι τώρα. Στη συνέχεια, οι τιμές των κατηγορημάτων για τους συγκεκριμένους κανόνες υπολογίζονται κατά τη διάρκεια του επόμενου βήματος εκτέλεσης, ενόσω όλα τα υπολογιστικά στοιχεία είναι απασχολημένα από τον υπολογισμό των κελιών που τους αντιστοιχούν.



Σχήμα 3.2: Προτεινόμενη Αρχιτεκτονική.

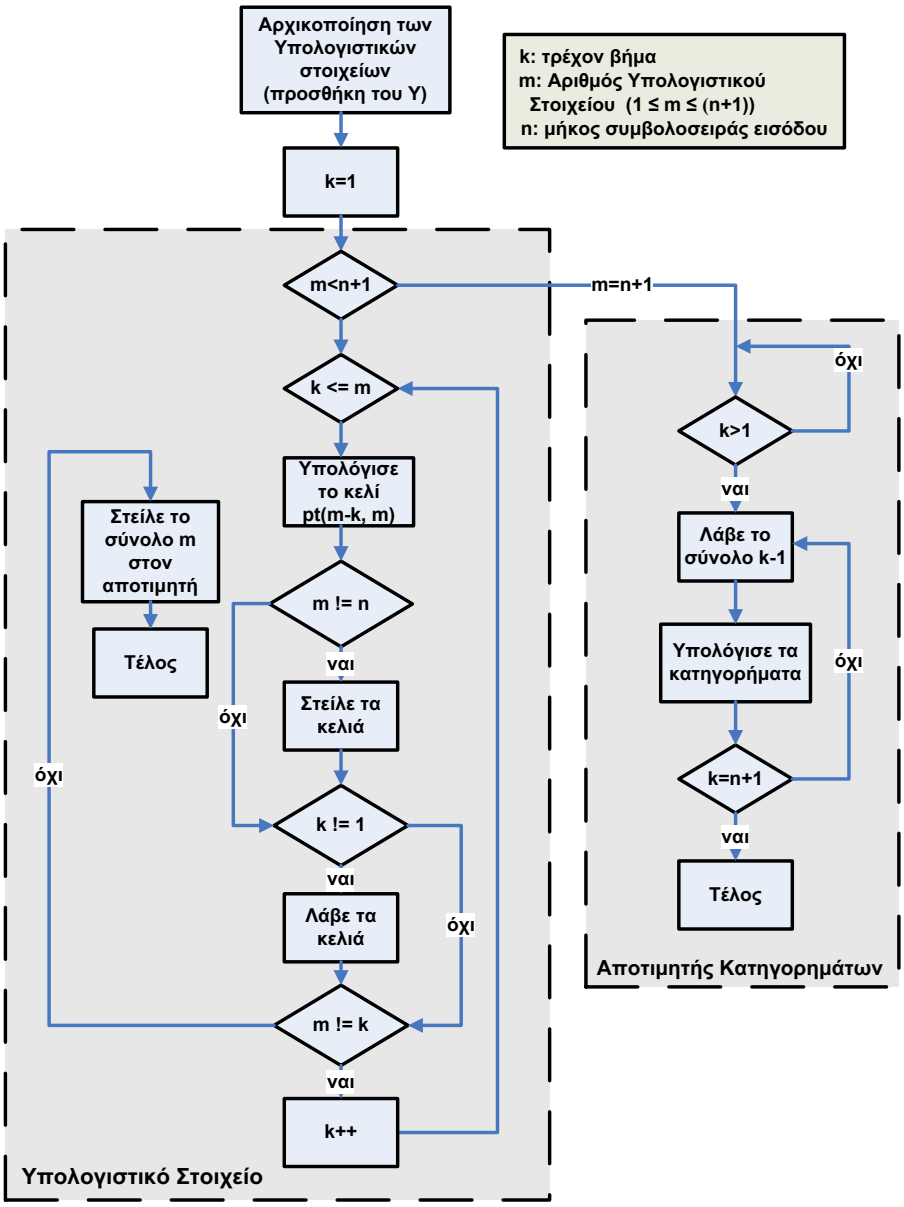
Βήμα (Εργασία)	Υπολογιστικό στοιχείο				Αποτιμητής κατηγορημάτων
	P_1	P_2	P_3	P_4	
$\kappa = 1$ (Υπολογισμός)	pt(0,1)	pt(1,2)	pt(2,3)	pt(3,4)	-
$\kappa = 1$ (Αποστολή)	pt(0,1)	pt(1,2)	pt(2,3)	-	-
$\kappa = 1$ (Λήψη)	-	pt(0,1)	pt(1,2)	pt(2,3)	-
$\kappa = 1$ (Αποτίμηση)	-	-	-	-	-
$\kappa = 2$ (Υπολογισμός)	-	pt(0,2)	pt(1,3)	pt(2,4)	-
$\kappa = 2$ (Αποστολή)	pt(0,1)	pt(0,1-2)	pt(1,2-3)	-	-
$\kappa = 2$ (Λήψη)	-	-	pt(0,1-2)	pt(1,2-3)	pt(0,1)
$\kappa = 2$ (Αποτίμηση)	-	-	-	-	pt(0,1)
$\kappa = 3$ (Υπολογισμός)	-	-	pt(0,3)	pt(1,4)	-
$\kappa = 3$ (Αποστολή)	-	pt(0-1,2)	pt(0,1-3)	-	-
$\kappa = 3$ (Λήψη)	-	-	-	pt(0,1-3)	pt(0-1,2)
$\kappa = 3$ (Αποτίμηση)	-	-	-	-	pt(0-1,2)
$\kappa = 4$ (Υπολογισμός)	-	-	-	pt(0,4)	-
$\kappa = 4$ (Αποστολή)	-	-	pt(0-2,3)	-	-
$\kappa = 4$ (Λήψη)	-	-	-	-	pt(0-2,3)
$\kappa = 4$ (Αποτίμηση)	-	-	-	-	pt(0-2,3)
$\kappa = 5$ (Υπολογισμός)	-	-	-	-	-
$\kappa = 5$ (Αποστολή)	-	-	-	pt(0,4)	-
$\kappa = 5$ (Λήψη)	-	-	-	-	pt(0,4)
$\kappa = 5$ (Αποτίμηση)	-	-	-	-	pt(0,4)

Πίνακας 3.1: Σχήμα Εκτέλεσης και Επικοινωνίας για μήκος συμβολοσειράς εισόδου $n = 4$.

Συνεπώς, κάθε υπολογιστικό στοιχείο επαναλαμβάνει την ακόλουθη διαδικασία :

1. Υπολογίζει ένα κελί.
2. Ελέγχει εάν πρέπει να στείλει ή να λάβει κελιά από κάποιο άλλο υπολογιστικό στοιχείο.
3. Ελέγχει εάν τελείωσαν τα προς υπολογισμό κελιά, οπότε στέλνει τα κελιά που έχει υπολογίσει στον αποτιμητή κατηγορημάτων.

Από την άλλη πλευρά, ο αποτιμητής κατηγορημάτων περιμένει να λάβει κελιά και μόλις τα λάβει, όσο τα υπολογιστικά στοιχεία υπολογίζουν τα κελιά τους, υπολογίζει τα κατηγορήματα. Προφανώς, το πρώτο στοιχείο ποτέ δεν λαμβάνει κελιά και το τελευταίο ποτέ δεν στέλνει. Η πλήρης διαδικασία για μήκος συμβολοσειράς $n = 4$ αναπαρίσταται στον πίνακα 3.1 καθώς και στο σχήμα 3.2.
Για τον υπολογισμό των τιμών των κατηγορημάτων, ο ορισμός 2 της ενότητας 2.2.2



Σχήμα 3.3: Διάγραμμα Ροής Αρχιτεκτονικής.

πρέπει να επεκταθεί κατάλληλα. Για κάθε κανόνα του τελεστή $\otimes$, ένα σύνολο εντολών για τα κατηγορήματα πρέπει να εισαχθεί. Οι εντολές αυτές, θα εκτελούνται από τον αποτιμητή κατηγορημάτων ενόσω οι κανόνες του τελεστή $\otimes$ εκτελούνται από τα υπολογιστικά στοιχεία. Οι εντολές που εισήχθησαν είναι οι ακόλουθες:

- Εάν δημιουργηθεί νέος κανόνας, εκτός από τη μετακίνηση της τελείας του παλαιού κανόνα μία ή περισσότερες θέσεις δεξιά, τα κατηγορήματα των συμβόλων του προηγούμενο κανόνα αντιγράφονται στο νέο.
- Εάν η τελεία φτάσει στο τέλος του κανόνα, τα συντιθέμενα κατηγορούμενα των συμβόλων που βρίσκονται στο αριστερό μέλος του κανόνα παραγωγής, πρέπει να αποτιμηθούν.

- Εάν η τελεία μετακινηθεί πάνω από ένα σύμβολο U εξαιτίας ενός κανόνα $U_1 \rightarrow \delta$ ή ενός κανόνα $U_2 \in R$, τότε οι τιμές των κατηγορημάτων του U_1 ή U_2 πρέπει να αντιγραφούν στο U και όλα τα κληρονομούμενα κατηγορήματα να αποτιμηθούν.
- Εάν κατά την αποτίμηση ενός κατηγόρηματος ή κατά την αντιγραφή του διαπιστωθεί ότι έχει μη αποδεκτή τιμή, τότε ο κανόνας πρέπει να σβηστεί προκειμένου να οδηγηθεί από τη σημασιολογία ο συντακτικός αναλυτής (semantic drive).

Τα παραπάνω, σε συνδυασμό με τον ορισμό 2, οδηγούν στην ακόλουθη επέκταση:

Ορισμός 3

$Q \otimes R = \{A \rightarrow \alpha U \beta_\bullet \gamma \mid A \rightarrow \alpha_\bullet U \beta\gamma \in Q,\ \beta \xrightarrow{*} \lambda \text{ και } U \rightarrow \delta_\bullet \in R\}$

- *Αντίγραψε τα υπολογισμένα κατηγορήματα του κανόνα* $A \rightarrow \alpha_\bullet U\beta\gamma$ *στον καινούργιο* $A \rightarrow \alpha U\beta_\bullet\gamma$. *Εάν προκύψει σφάλμα, σβήσε τον κανόνα.*
- *Αντίγραψε τα υπολογισμένα κατηγορήματα του συμβόλου* U *από τον κανόνα* $U \rightarrow \delta_\bullet$ *στον* $A \rightarrow \alpha U\beta_\bullet\gamma$. *Εάν προκύψει σφάλμα, σβήσε τον κανόνα.*

και

$\{B \rightarrow \delta C\xi_\bullet\eta \mid \gamma = \lambda,\ B \rightarrow \delta_\bullet C\xi\eta \in Y \text{ και } \xi \xrightarrow{*} \lambda,\ C \xrightarrow{*} A\}$

- *Αντίγραψε τα υπολογισμένα κατηγορήματα του κανόνα* $B \rightarrow \delta_\bullet C\xi\eta$ *στον καινούργιο* $B \rightarrow \delta C\xi_\bullet\eta \mid \gamma = \lambda$. *Εάν προκύψει σφάλμα, σβήσε τον κανόνα.*
- $\forall\ C \xrightarrow{*} A$ *υπολόγισε τα κληρονομούμενα κατηγορήματα.*
- *Υπολόγισε τα συντιθέμενα κατηγορήματα του* B. *Εάν προκύψει σφάλμα, σβήσε τον κανόνα.*

Εάν το R *είναι απλώς ένα τερματικό σύμβολο τότε:*

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

- *Αντίγραψε τα υπολογισμένα κατηγορήματα του κανόνα* $A \rightarrow \alpha_\bullet U\beta\gamma$ *στον καινούργιο* $A \rightarrow \alpha U\beta_\bullet\gamma$. *Εάν προκύψει σφάλμα, σβήσε τον κανόνα.*

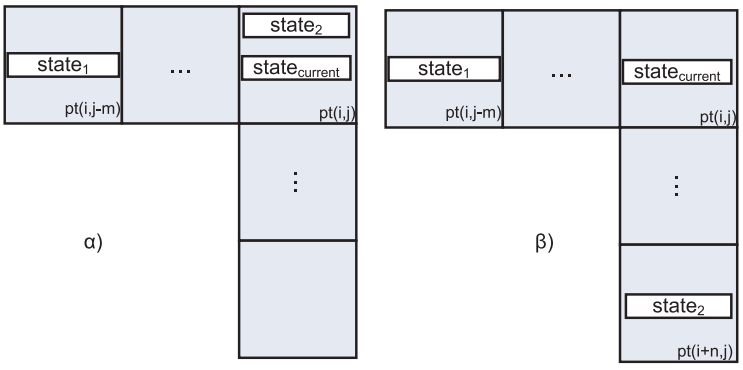
και

$\{B \rightarrow \delta C\xi_\bullet\eta \mid \gamma = \lambda,\ B \rightarrow \delta_\bullet C\xi\eta \in Y \text{ και } \xi \xrightarrow{*} \lambda,\ C \xrightarrow{*} A\}$

- *Αντίγραψε τα υπολογισμένα κατηγορήματα του κανόνα* $B \rightarrow \delta_\bullet C\xi\eta$ *στον καινούργιο* $B \rightarrow \delta C\xi_\bullet\eta \mid \gamma = \lambda$. *Εάν προκύψει σφάλμα, σβήσε τον κανόνα.*
- $\forall\ C \xrightarrow{*} A$ *υπολόγισε τα κληρονομούμενα κατηγορήματα.*
- *Υπολόγισε τα συντιθέμενα κατηγορήματα του* B. *Εάν προκύψει σφάλμα, σβήσε τον κανόνα.*

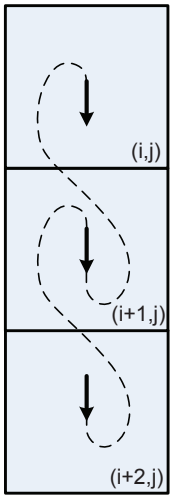
Όπως έχει ήδη αναφερθεί, η αποτίμηση των κατηγορημάτων λαμβάνει χώρα για ένα υπολογιστικό στοιχείο, μόνο αφού έχει τελειώσει όλους τους υπολογισμούς του, δηλαδή έχουν υπολογιστεί όλα τα κελιά της στήλης του. Για το λόγο αυτό, είναι κρίσιμο να διατρεχθούν τα κελιά με τέτοιο τρόπο ώστε να διασφαλίζεται η σωστή αποτίμηση με ένα μόλις πέρασμα. Διαφορετικά, μπορεί στον υπολογισμό ενός κατηγορήματος να μην ληφθεί υπόψη κάποιο άλλο κατηγόρημα που το επηρεάζει, οδηγώντας έτσι σε λάθος αποτέλεσμα.

Ακριβώς επειδή ο αποτιμητής λαμβάνει όλα τα στοιχεία της στήλης και όχι μόνον εκείνα που παράγουν το τελικό συντακτικό δέντρο και επομένως και την συμβολοσειρά εισόδου, έπρεπε να ληφθεί ειδική μνεία ώστε να μην χρειάζεται ο αποτιμητής να διατρέχει εκ νέου το συντακτικό δέντρο. Προκειμένου λοιπόν να μην σπαταλάται χρόνος, ο συντακτικός αναλυτής έχει επεκταθεί ώστε να κρατάει για κάθε κανόνα πληροφορίες από που ήρθε, δηλαδή λόγω ποιον κανόνων δημιουργήθηκε. Κάθε κανόνας μπορεί να έχει δημιουργηθεί το πολύ από άλλους δύο. Στην περίπτωση της διαδικασίας που αντιστοιχεί στον scanner του Earley, ο κανόνας προέρχεται από έναν μόνον κανόνα, ενώ σε κάθε άλλη περίπτωση έχει παραχθεί από δύο προηγούμενους. Με βάση τα επιπλέον στοιχεία προέλευσης που κρατούνται, μπορεί ο αποτιμητής να εντοπίσει αμέσως τους κανόνες προέλευσης και να κάνει τις μεταφορές κατηγορημάτων που επιβάλλονται από τις εξισώσεις του ορισμού 3, σύμφωνα με το ακόλουθο σκεπτικό. Για τον υπολογισμό των κληρονομούμενων κατηγορημάτων ενός κανόνα ($state_{current}$) σε κάποιες περιπτώσεις δεδομένα από δύο άλλους κανόνες ($state_1$ και $state_2$) χρειάζονται (βλέπε σχήμα 3.4), ένας από την ίδια σειρά και ένας από την ίδια στήλη. Ο κανόνας που ανήκει στην ίδια στήλη, μπορεί να βρίσκεται είτε στο ίδιο κελί (σχήμα 3.4α) είτε σε κάποιο χαμηλότερο (σχήμα 3.4β). Για την αντιμετώπιση και των δύο περιπτώσεων, ο τρόπος που θα διασχίζεται μια στήλη είναι από κάτω προς



Σχήμα 3.4: Οι δύο περιπτώσεις αποτίμησης κατηγορήματος ενός κανόνα.

τα πάνω, όσον αφορά στα κελιά και από πάνω προς τα κάτω, όσον αφορά στο εσωτερικό των κελιών, όπως παρουσιάζεται στο σχήμα 3.5. Σχετικά με τα συντιθέμενα κατηγορήματα, ακριβώς λόγω της φύσης του αλγορίθμου (από πάνω προς τα κάτω και αριστερά προς τα δεξιά), μπορούν να υπολογιστούν σωστά με τα δεδομένα που υπάρχουν ήδη. Επομένως, μόλις η τελεία φτάσει στο τέλος του κανόνα παραγωγής, μπορεί να γίνει η αποτίμησή τους.



Σχήμα 3.5: Προτεινόμενη Διάσχιση της Στήλης.

3.2 Το "Hunt the Wumpus"

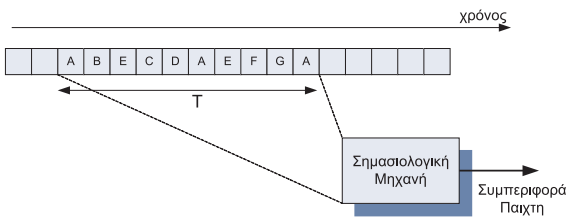
Το "Hunt the Wumpus" είναι ένα από τα πρώτα παιχνίδια υπολογιστή, βασισμένο στην ιδέα του "κρυφτού". Ο ήρωας προσπαθεί να αποφύγει ένα τέρας (Wumpus) που παραφυλάει σε δαιδαλώδεις διαδρόμους. Ο ήρωας μπορεί να κινείται οριζοντίως και καθέτως σε ένα πλέγμα, προσπαθώντας να βρει χρυσάφι. Ο Wumpus κινείται τυχαία προσπαθώντας να εντοπίσει και να φάει τον ήρωα. Μέσα στο πλέγμα, σε ένα τετραγωνάκι υπάρχει το χρυσάφι ενώ σε άλλα τάφροι. Ο ήρωας μπορεί κάθε χρονική στιγμή είτε να κινηθεί προς μια μόνον από τις 4 κατευθύνσεις είτε να μείνει στο ίδιο σημείο και να αναγνωρίσει τί υπάρχει στα γειτονικά του τετραγωνάκια. Η μόνη πληροφορία που λαμβάνεται ανεξαρτήτως επιλογής είναι το κατά πόσο ο Wumpus είναι σε διπλανό τετραγωνάκι. Το συγκεκριμένο παιχνίδι έχει ιδιαίτερη σημασία, καθώς μπορεί να παρομοιαστεί με το πραγματικό πρόβλημα ελέγχου μιας αναπηρικής καρέκλας, όπου ήρωας είναι πια η καρέκλα, τάφροι τα σταθερά εμπόδια, Wumpus τα κινούμενα εμπόδια και χρυσάφι ο τελικός προορισμός.

Το 2004 οι Panagopoulos et al [58] μετέτρεψαν το συγκεκριμένο παράδειγμα από το πεδίο της τεχνητής νοημοσύνης στην αντίστοιχη κατηγορική γραμματική. Το παιχνίδι παίζεται ανάμεσα στον υπολογιστή και έναν άνθρωπο. Ο παίχτης μπορεί να ακολου-

N={BEHAVIOUR, ACTIONS, ACTION, OFFENSIVE, DEFENSIVE}
T={a,b,c,d,e,f}
P={ BEHAVIOUR→ ACTIONS |. ACTIONS→ ACTION ACTIONS | ACTION|. ACTION→ DEFENSIVE | OFFENSIVE |.OFFENSIVE→ d | e | f |. DEFENSIVE→ a | b | c |.}
A={attack, defend, result} όπου d(attack) ∈ I, d(defend) ∈ I, d(result) ∈ I
SR(BEHAVIOUR→ ACTIONS |.)= {BEHAVIOUR result=ACTIONS.attack - ACTIONS.defend}
SR(ACTIONS→ACTION ACTIONS ACTION|.) {ACTIONS[1].attack=ACTIONS[2].attack+ACTION.attack, ACTIONS[1].defend=ACTIONS[2].defend+ACTION.defend}
SR(ACTION→OFFENSIVE |DEFENSIVE|.)= {ACTION.attack=OFFENSIVE .attack, ACTION.defend=DEFENSIVE.defend }
SR(DEFENSIVE→a | b | c |.)={DEFENSIVE.defend=f(a|b|c)}
SR(OFFENSIVE→d| e | f |.)={OFFENSIVE.attack=g(d|e|f)}

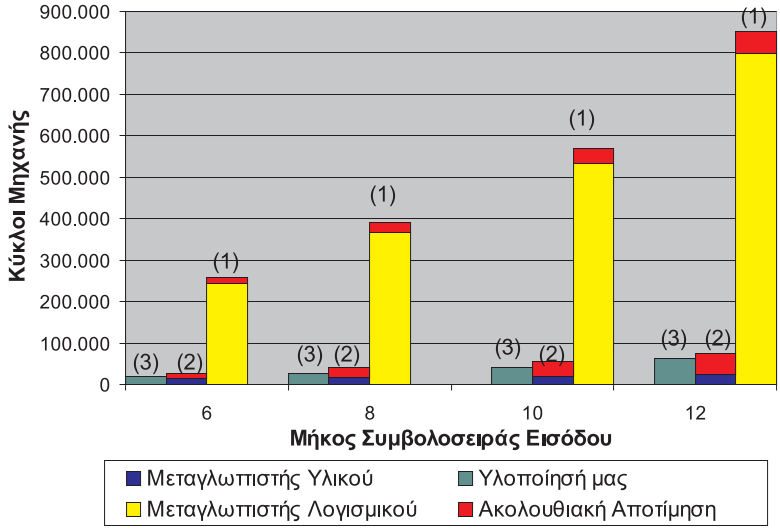
Πίνακας 3.2: Η Κατηγορική Γραμματική G_{Wumpus} του Wumpus.

θεί επιθετική ή αμυντική στρατηγική και ο υπολογιστής, προκειμένου να υπολογίσει τις επόμενες κινήσεις του, χρειάζεται έναν μηχανισμό αναγνώρισης και πρόβλεψης. Έτσι, οι ενέργειες του παίχτη μπορούν να αντιστοιχιθούν σε ένα ειδικό αλφάβητο, στο οποίο κάθε τερματικό σύμβολο συμβολίζει μια συγκεκριμένη ενέργεια. Κατά τη διάρκεια του παιχνιδιού, τα σύμβολα αυτά παράγουν συμβολοσειρές, τις οποίες αναλύει και αποτιμά ο υπολογιστής κατά συγκεκριμένα χρονικά διαστήματα. Τέτοιες συμβολοσειρές μπορούν να ληφθούν υπόψιν από τη σημασιολογική ανάλυση, οδηγώντας σε ακριβή πρόβλεψη της επόμενης κίνησης (βλέπε σχήμα 3.6). Μια τέτοια γραμματική G_{Wumpus} δίνεται στον πίνακα 3.2. Κατά τη διάρκεια της αναγνώρισης, υπολογίζεται η τιμή ενός κατηγορήματος BEHAVIOR.result. Αυτό το κατηγόρημα είναι στη ρίζα του συντακτικού δέντρου και θετική τιμή δείχνει επιθετική συμπεριφορά ενώ αρνητική αμυντική. Γενικότερα, η απόσταση από το μηδέν είναι ένα μέτρο της συμπεριφοράς του παίχτη.



Σχήμα 3.6: Διαδικασία πρόβλεψης της συμπεριφοράς του παίχτη.

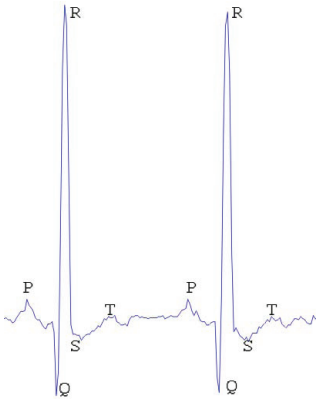
Στο σχήμα 3.7 παρουσιάζεται η αποδοτικότητα (σε κύκλους μηχανής) τριών διαφορετικών μεθόδων για ποικίλο μήκος συμβολοσειράς εισόδου (6, 8, 10 και 12) της γραμματικής G_{Wumpus}. Οι τρεις μέθοδοι είναι: αναλυτής σε λογισμικό και ξεχωριστή αποτίμηση των κατηγορημάτων (1), παράλληλος αναλυτής σε υλικό με ξεχωριστή αποτίμηση των κατηγορημάτων (2) και τέλος η δική μας προσέγγιση με παράλληλο αναλυτή σε υλικό με ταυτόχρονη αποτίμηση των κατηγορημάτων (3). Φαίνεται ότι η επιτάχυνση σε σχέση με τον αναλυτή σε λογισμικό είναι σχεδόν μια τάξη μεγέθους. Προφανώς, οι κύκλοι που χρειάζονται για να δημιουργηθεί το συντακτικό δέντρο και κατόπιν να διατρεχθεί είναι περισσότεροι από εκείνους της προτεινόμενης προσέγγισης.



Σχήμα 3.7: Αποτελέσματα Επιδόσεων για τη γραμματική G_{Wumpus}.

3.3 Ηλεκτροκαρδιογράφημα

Στη συντακτική αναγνώριση προτύπων, η διαδικασία της αναγνώρισης μετατρέπεται στην ισοδύναμη της αναγνώρισης γλωσσολογικών αναπαραστάσεων των προς αναγνώριση προτύπων. Οι γλωσσικές αυτές αναπαραστάσεις, περιγράφονται από μια συγκεκριμένη γραμματική, τη γραμματική προτύπων (pattern grammar) [2]. Η γραμματική αυτή περιγράφει τα προς αναγνώριση πρότυπα με ένα συμβατικό τρόπο. Στην περίπτωση του ηλεκτροκαρδιογραφήματος (ECG), όπου εμφανίζεται μεγάλο πλήθος διαφορετικών μορφολογιών, λόγω παραγόντων όπως ο προστιθέμενος θόρυβος ενώ πολλές παράμετροι πρέπει να αξιολογηθούν, χρειάζονται ισχυρές γραμματικές, ικανές να περιγράψουν τόσο το συντακτικό όσο και το σημασιολογικό κομμάτι της αναγνώρισης. Ακριβώς λόγω της περιγραφικής τους ικανότητας, συνήθως επιλέγονται οι κατηγορικές γραμματικές [13], [59].

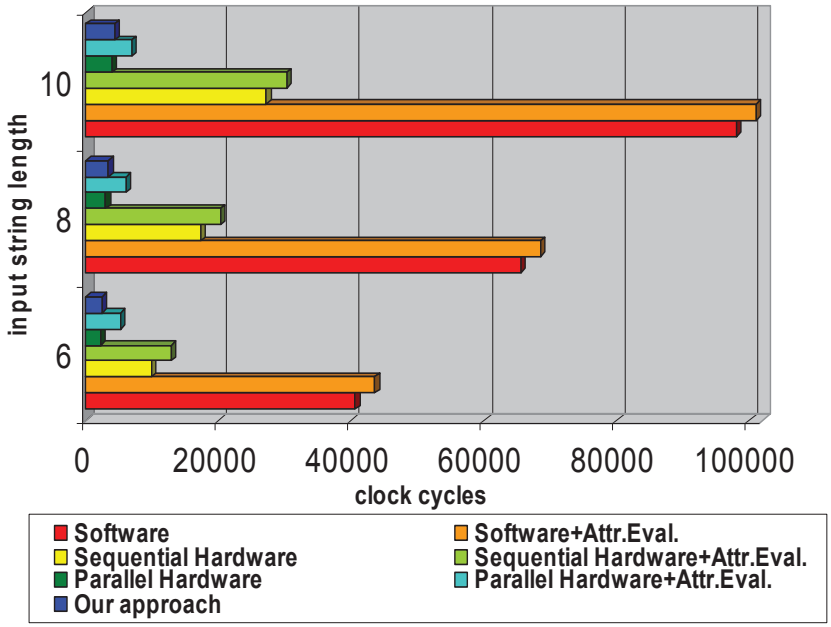


Σχήμα 3.8: Μια τυπική μορφή ECG.

Το ECG δεν είναι τίποτε άλλο από ηλεκτρικό δυναμικό ανάμεσα σε διάφορα σημεία του σώματος, το οποίο μπορεί να μετρηθεί με χρήση κατάλληλου εξοπλισμού και ηλεκτροδίων κατάλληλα τοποθετημένων στο σώμα του εξεταζόμενου. Το μετρούμενο δυναμικό, είναι άμεσα συνδεδεμένο με τη μυική δραστηριότητα της καρδιάς. Αρχικά μετράται ένα θετικό έπαρμα, το οποίο καλείται P και ξεκινά την διέγερση. Ακολούθως, καθώς η διέγερση αρχίζει να εξαπλώνεται, εμφανίζεται ένα αρνητικό έπαρμα (Q) ακολουθούμενο από μια μεγάλη και ταχεία αντιστροφή (έπαρμα R). Καθώς η διέγερση πλησιάζει στο πέρας της, ένα νέο αρνητικό έπαρμα εμφανίζεται, το S. Τέλος, η επαναπόλωση των κοιλιών της καρδιάς σηματοδοτείται από ένα ακόμη θετικό έπαρμα, το T. Μια τυπική μορφή ηλεκτροκαρδιογραφήματος παρουσιάζεται στο σχήμα 3.8. Για τη γλωσσολογική περιγραφή των παραπάνω, στο συγκεκριμένο παράδειγμα, έγινε χρήση της κατηγορικής γραμματικής των Trahanias et al [13], G_{ECG}. Περιληπτικά, η G_{ECG} έχει τα τερματικά σύμβολα $\{K^+, K^-, E, \Pi\}$, όπου το K^+ αντιπροσωπεύει τις θετικές ακμές, το K^- τις αρνητικές, το E τα ευθύγραμμα τμήματα και Π τα παραβολικά. Συνεπώς, κάθε κυματομορφή ECG μετατρέπεται σε μια συμβολοσειρά των παραπάνω συμβόλων. Για κάθε συμβολοσειρά εισόδου, επτά κατηγορήματα (συντιθέμενα και κληρονομούμενα) υπολογίζονται: το πλήθος των καρδιακών κύκλων, το πλήθος των κλάσεων QRS, η διάρκεια του αριστερού/δεξιού μέρους μια ακμής, το ύψος του αριστερού/δεξιού μέρους μια ακμής και τέλος η πιθανότητα για μια ακμή να είναι P ή T. Η γραμματική αποτελείται από 34 συντακτικούς κανόνες με μέγιστο πλήθος συμβόλων σε κάθε κανόνα το 7.

Στο σχήμα 3.9 παρουσιάζεται η επίδοση για 7 διαφορετικές υλοποιήσεις και διαφορετικό μέγεθος συμβολοσειρών εισόδου. Η προτεινόμενη αρχιτεκτονική δίνει τα καλύτερα αποτελέσματα, επιτυγχάνοντας επιτάχυνση περίπου x20 εν συγκρίσει με την προσέγγιση του καθαρού λογισμικού, σε ένα συμβατικό μικροεπεξεργαστή RISC. Δεδομένου ότι η τεχνολογία που χρησιμοποιείται για την υλοποίηση στο υλικό είναι η ίδια με εκείνη που χρησιμοποιείται για το μικροεπεξεργαστή, η συχνότητα των

ρολογιών τους μπορεί να είναι η ίδια. Συνεπώς, η απόδοση όλων των υλοποιήσεων μετράται σε κύκλους ρολογιού. Στο σχήμα 3.9, φαίνονται οι κύκλοι ρολογιού για αναλυτή σε λογισμικό, για αναλυτή σε λογισμικό και τον αποτιμητή κατηγορημάτων, για ακολουθιακό αναλυτή σε υλικό, για ακολουθιακό αναλυτή σε υλικό και τον αποτιμητή κατηγορημάτων, για τον προτεινόμενο παράλληλο αναλυτή και τέλος για τον παράλληλο αναλυτή μας και την ταυτόχρονο αποτίμηση των κατηγορημάτων. Όπως φαίνεται, ο αριθμός των κύκλων που χρειάζονται για να δημιουργηθεί πρώτα το συντακτικό δέντρο και μετά να διασχισθεί προκειμένου να υπολογιστούν τα κατηγορήματα είναι σε κάθε περίπτωση - σε λογισμικό ή σε υλικό, ακολουθιακά ή παράλληλα - περισσότεροι από τον αναγκαίο αριθμό κύκλων της προτεινόμενης υλοποίησης.

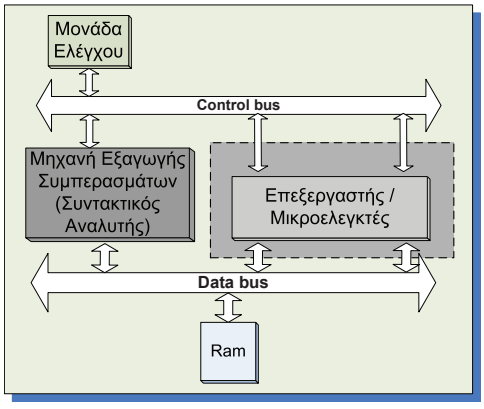


Σχήμα 3.9: Αποτελέσματα Επιδόσεων για τη γραμματική G_{ECG}.

Κεφάλαιο 4

Αποτίμηση Κατηγορημάτων Με Χρήση Συντακτικού Αναλυτή και Εσωτερικού Επεξεργαστή

Στην αρχιτεκτονική που παρουσιάστηκε στο προηγούμενο κεφάλαιο, η αποτίμηση των κατηγορημάτων βάσει των σημασιολογικών κανόνων γινόταν από έναν εξωτερικό επεξεργαστή. Δηλαδή η όλη υλοποίηση του συντακτικού αναλυτή είχε "κατέβει" σε ένα FPGA και ο επεξεργαστής συνδεόταν εξωτερικά με την υλοποίηση. Συνεπώς, εγγενή πηγή καθυστέρησης της προηγούμενης υλοποίησης αποτελεί η χρονοβόρως επικοινωνία μεταξύ του FPGA και του επεξεργαστή. Σε μια προσπάθεια να απαλειφθεί αυτή η καθυστέρηση, αντικαταστάθηκε σε πρώτο στάδιο ο επεξεργαστής από εσωτερικό μικροελεγκτή - τον PicoBlaze [60] - και κατόπιν από εσωτερικό μικροεπεξεργαστή soft core, τον MicroBlaze [61]. Επομένως, η προτεινόμενη αρχιτεκτονική του σχήματος 3.1 μετασχηματίζεται σε εκείνη του σχήματος 4.1.



Σχήμα 4.1: Προτεινόμενη Αρχιτεκτονική με Ενσωματωμένο Επεξεργαστή/ Μικροελεγκτή.

4.1 Αντικατάσταση Εξωτερικού Επεξεργαστή με το Μικροελεγκτή PicoBlaze

Οι μικροελεγκτές και τα FPGA μπορούν να υλοποιήσουν οποιαδήποτε λογική συνάρτηση. Όμως, κάθε επιλογή έχει διαφορετικά οφέλη από πλευράς κόστους, απόδοσης και απλότητας χρήσης. Οι μικροελεγκτές είναι κατάλληλοι για εφαρμογές ελέγχου, ιδίως με ευρέως μεταβαλλόμενες απαιτήσεις. Οι αναγκαίοι πόροι του FPGA που χρειάζονται για να υλοποιηθεί ένας μικροελεγκτής είναι σχετικά σταθεροί. Η ίδια λογική του FPGA επαναχρησιμοποιείται από τις διάφορες εντολές του μικροελεγκτή, εξοικονομώντας πόρους. Οι απαιτήσεις για μνήμη προγράμματος αυξάνουν, καθώς αυξάνει η πολυπλοκότητα.

Ο προγραμματισμός ακολουθιών ελέγχου ή μηχανών καταστάσεων (state machines) σε assembly είναι συχνά ευκολότερος από τη δημιουργία αντίστοιχων δομών στην λογική του FPGA. Βέβαια, οι μικροελεγκτές τυπικά έχουν περιορισμένες επιδόσεις. Κάθε εντολή εκτελείται ακολουθιακά και όσο αυξάνει η πολυπλοκότητα της εφαρμογής, αυξάνουν και οι αναγκαίες εντολές για την υλοποίηση της εφαρμογής. Έτσι, το σύστημα οδηγείται σε απώλεια επιδόσεων. Αντιθέτως, η απόδοση στα FPGA είναι πιο ευέλικτη. Για παράδειγμα, ένας αλγόριθμος μπορεί να υλοποιηθεί ακολουθιακά ή πλήρως παράλληλα, αναλόγως με τις απαιτήσεις της απόδοσης. Μια πλήρως παράλληλη υλοποίηση είναι ταχύτερη αλλά καταναλώνει περισσότερους πόρους στο FPGA. Ένας μικροελεγκτής ενσωματωμένος μέσα στο FPGA, προσφέρει τα πλεονεκτήματα και των δύο προσεγγίσεων. Ο μικροελεγκτής υλοποιεί σύνθετες συναρτήσεις που δεν έχουν χρονικούς περιορισμούς ενώ συναρτήσεις που υπόκεινται σε περιορισμούς χρόνου και ταχύτητας υλοποιούνται με τη χρήση των λογικών στοιχείων του FPGA. Για παράδειγμα, ένας μικροελεγκτής δεν μπορεί να αντεπεξέλθει σε συμβάντα ταχύτερα από μερικά microseconds ενώ το FPGA μπορεί να αντεπεξέλθει σε πολλαπλά, ταυτόχρονα συμβάντα σε μερικά nanoseconds. Αντίστροφα, ένας μικροελεγκτή είναι πιο αποδοτικός από οικονομικής σκοπιάς και απλούστερος για την εκτέλεση μετατροπών μορφής ή πρωτοκόλλων.

Υπάρχουν δεκάδες 8-bit αρχιτεκτονικές μικροελεγκτών και ρεπερτόρια εντολών. Τα σύγχρονα FPGA μπορούν να υλοποιήσουν αποδοτικά κάθε 8-bit μικροελεγκτή και υπάρχουν διαθέσιμοι soft cores πυρήνες που υποστηρίζουν δημοφιλή ρεπερτόρια εντολών, όπως του PIC, 8051, AVR, 6502, 8080, Z80 κ.α.. Επομένως, ποιος ο λόγος επιλογής του μικροελεγκτή PicoBlaze αντ'αυτών; Ο μικροελεγκτής PicoBlaze έχει σχεδιαστεί και βελτιστοποιηθεί ειδικά για τα FPGA των αρχιτεκτονικών Spartan-3, Virtex-II και Virtex-II Pro. Η συμπαγής αρχιτεκτονική τους καταναλώνει πολύ λιγότερους πόρους σε σχέση με αντίστοιχους 8-bit μικροελεγκτές στο FPGA. Επιπλέον, ο PicoBlaze δίδεται δωρεάν από τη Xilinx και μάλιστα σε επίπεδο κώδικα VHDL. Προφανώς, μετά από κάποιο διάστημα οι μικροελεγκτές θεωρούνται απαρχαιωμένοι.

Ακριβώς όμως λόγω της δυναμικής του φύσης (κώδικας VHDL), ο PicoBlaze δεν κινδυνεύει από κάτι τέτοιο. Επιπλέον, είναι επεκτάσιμος και ευέλικτος. Οι μέχρι πρότινος μικροελεγκτές των FPGA, βρίσκονταν εξωτερικά του FPGA, περιορίζοντας έτσι την συνδεσιμότητα με άλλες λειτουργίες υλοποιημένες στο FPGA και κατ' επέκταση και την απόδοση. Αντιθέτως, ο PicoBlaze είναι πλήρως ενσωματωμένος στο FPGA με εκτεταμένες και ευέλικτες διασυνδέσεις on-chip σε άλλους πόρους του FPGA. Τα σήματα παραμένουν στο ίδιο FPGA, βελτιώνοντας την τελική απόδοση. Επιπροσθέτως, βοηθάει στην ελάττωση του κόστους καθώς είναι μια λύση που δεν απαιτεί επιπλέον τσιπ και μπορεί να καταναλώσει ακόμη και πόρους που έχουν περισσέψει από την υπόλοιπη υλοποίηση. Τέλος, είναι πολύ αποδοτικός στην κατανάλωση πόρων, επιτρέποντας την κατανεμημένη εκτέλεση πολύπλοκων εφαρμογών σε περισσότερους του ενός PicoBlaze.

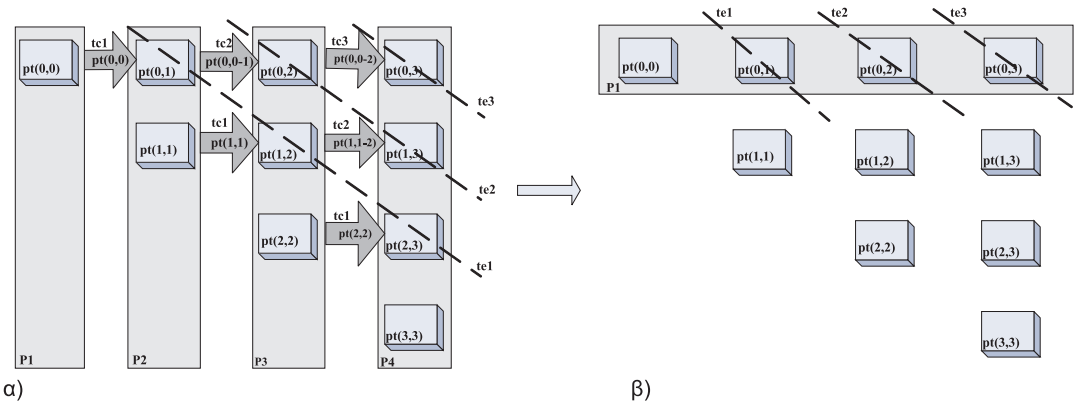
4.1.1 Υλοποίηση με PicoBlaze

Μια αρχιτεκτονική που κάνει χρήση του μικροεπεξεργαστή PicoBlaze και ενοποιεί το συντακτικό αναλυτή και τον αποτιμητή σε ένα FPGA, θα παρουσιαστεί ακολούθως. Η συγκεκριμένη υλοποίηση σε υλικό, επιταχύνει ευφυείς εφαρμογές βασισμένες στο λογικό προγραμματισμό. Η προηγούμενη προσέγγιση του κεφαλαίου 3 έχει βελτιωθεί, ελαττώνοντας το πλήθος των αναγκαίων υπολογιστικών στοιχείων κατά έναν παράγοντα n (μήκος συμβολοσειράς εισόδου). Το γεγονός αυτό, επέτρεψε την χρήση ενός μόνο στοιχείου, ενός FPGA, απαλείφοντας την ανάγκη για εξωτερικό μικροεπεξεργαστή. Επιπλέον, ο αλγόριθμος αποτίμησης των κατηγορημάτων βελτιώθηκε και χωρίστηκε σε δύο μέρη, τα οποία μπορούν να εκτελεστούν παράλληλα σε δύο μικροελεγκτές. Και οι δύο μικροελεγκτές βρίσκονται στο ίδιο FPGA (Xilinx Spartan-II), μαζί με την μηχανή εξαγωγής συμπερασμάτων. Συνεπώς, η προτεινόμενη αρχιτεκτονική είναι κατάλληλη για χρήση σε ενσωματωμένα συστήματα όπου η μικρή κατανάλωση, η μεταφερσιμότητα και το χαμηλό κόστος έχουν πρωτεύουσα σημασία. Οι δύο μικροελεγκτές, υπεύθυνοι για την αποτίμηση των κατηγορημάτων, είναι PicoBlaze. Η διεπαφή των μικροελεγκτών με το συντακτικό αναλυτή ακολουθεί τη φιλοσοφία του σχήματος 4.1 ενώ όλα τα δεδομένα αποθηκεύονται σε μοιραζόμενη μνήμη RAM. Η υλοποίηση έχει δοκιμαστεί στο Xilinx Spartan-II FPGA και η επιτάχυνση εξαρτάται από την εφαρμογή.

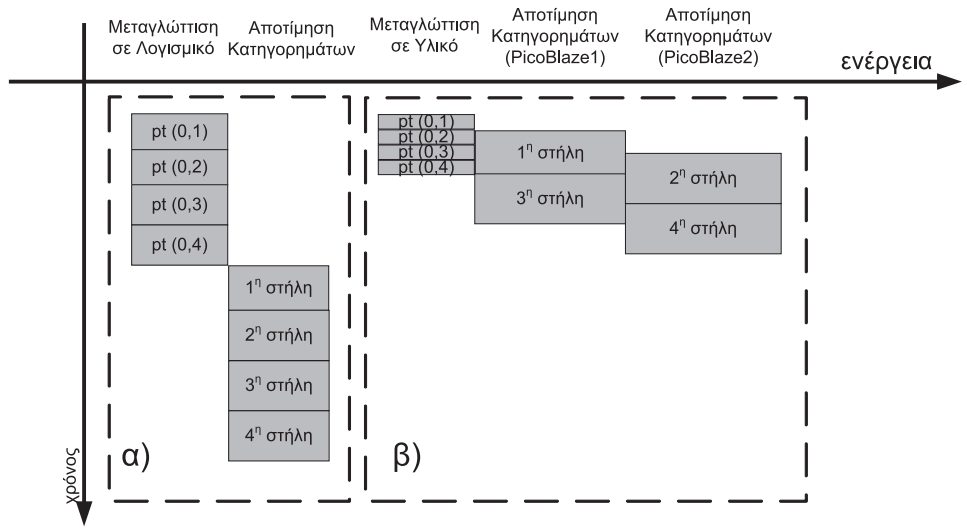
Η χρήση κατηγορικών γραμματικών για την αναπαράσταση της γνώσης οδηγεί στην χρήση ενός μόνον τερματικού συμβόλου, του κενού συμβόλου (NULL symbol). Συνεπώς, ο αναλυτής αναγνωρίζει μόνο συμβολοσειρές αποτελούμενες από τέτοια σύμβολα. Κατά τη διάρκεια αναγνώρισης μιας τέτοιας συμβολοσειράς, η σημασιολογία μπορεί να είναι τέτοια ώστε κατά την αποτίμηση των κατηγορημάτων, να καλύπτεται και η εξαγωγή συμπερασμάτων. Προκειμένου να γίνει συμβατή μια γραμματική χωρίς

τερματικά σύμβολα με τον αναλυτή, εισάγεται ένα νέο πλασματικό σύμβολο, το "d" (από το dummy). Συνεπώς, τώρα πια ο αναλυτής αναγνωρίζει συμβολοσειρές της μορφής $dd\ldots d$. Το μήκος των συμβολοσειρών εισόδου εξαρτάται από το μέγεθος του ερωτήματος της εφαρμογής.

Αφού $\alpha_i = d$ για $1 \leq i \leq n$, τα κελιά του πίνακα PT τα οποία υπολογίζονται κατά το βήμα εκτέλεσης t_{e1} έχουν τιμή, σύμφωνα με την εξίσωση 2.1, $pt(i,j) = pt(i,j-1) \otimes d$. Όμως, τα κελιά που ανήκουν στην κύρια διαγώνιο είναι όμοια, από συντακτικής πλευράς. Επομένως, και τα κελιά της επόμενης διαγωνίου, που παράγονται βάσει της κυρίας είναι όμοια από συντακτικής πλευράς. Επαγωγικά, μπορεί να αποδειχθεί ότι όλα τα κελιά που ανήκουν στην ίδια διαγώνιο περιέχουν τους ίδιους συντακτικούς κανόνες. Πρέπει βέβαια να τονισθεί ότι αν και τα κελιά έχουν τους ίδιους συντακτικούς κανόνες, οι τιμές των κατηγορημάτων τους είναι προφανώς διαφορετικές, αφού εξαρτώνται από τη θέση τους στον πίνακα PT και από τις τιμές άλλων κατηγορημάτων, συμβόλων που προηγούνται ή έπονται. Συνεπώς, η διαδικασία της συντακτικής ανάλυσης μπορεί να γίνει με χρήση ενός μόνο επεξεργαστικού στοιχείου, αντί για n, το οποίο υπολογίζει μόνο τα κελιά που ανήκουν στην πρώτη γραμμή του PT, όπως φαίνεται και στο σχήμα 4.2β. Μόλις υπολογιστεί ένα κελί, τα περιεχόμενά του αντιγράφονται στα υπόλοιπα κελιά της ίδιας διαγωνίου, ώστε να είναι διαθέσιμα προς χρήση κατά τον υπολογισμό των κατηγορημάτων. Για παράδειγμα, το κελί $pt(0,1)$ θα αντιγραφεί στα κελιά $pt(1,2)$ και $pt(2,3)$. Η επιβάρυνση για αυτή την ενέργεια είναι αμελητέα εν συγκρίσει με τη συνολική διαδικασία. Η αρχιτεκτονική που ακολουθείται για την συντακτική ανάλυση δίνει επιτάχυνση κατά ένα παράγοντα περίπου ίσο με 5, συγκρινόμενη με την προσέγγιση σε λογισμικό. Επιπλέον, το γεγονός ότι υπολογίζονται μόνον τα κελιά της πρώτης γραμμής, βελτιώνει θεαματικά την επιτάχυνση. Καθώς αυξάνει το μήκος της συμβολοσειράς εισόδου και κατ' επέκταση το μέγεθος του πίνακα PT, αυξάνει και η επιτάχυνση. Πειραματικά αποτελέσματα παρουσιάζο-



Σχήμα 4.2: Αρχιτεκτονική Αναλυτή για α) Γραμματικές με Τερματικά Σύμβολα β) Γραμματικές χωρίς Τερματικά Σύμβολα.



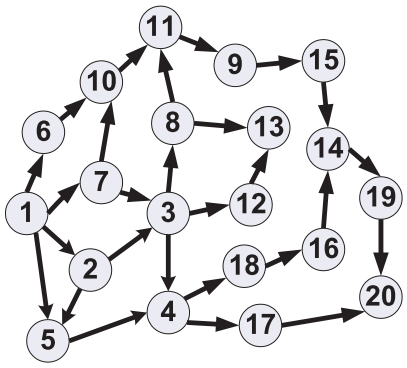
Σχήμα 4.3: Σύγκριση της προτεινόμενης αρχιτεκτονικής με χρήση PicoBlaze (β) με την υλοποίηση σε λογισμικό (α).

νται στα σχήματα 4.5 και 4.6.

Επιπλέον, ο περιορισμός των απαραίτητων υπολογιστικών στοιχείων απλοποιεί την υλοποίηση, ελευθερώνοντας περισσότερο χώρο (πόρους) για την ενσωμάτωση των δύο μικροελεγκτών. Για την αποδοτική χρήση και των δύο μικροελεγκτών, η διαδικασία της αποτίμησης των κατηγορημάτων έχει χωρισθεί σε δύο μέρη. Μόλις ο ένας μικροελεγκτής φτάσει σε κατάλληλο σημείο στους υπολογισμούς μιας στήλης, ειδοποιεί με διακοπή (interrupt) τον άλλο μικροελεγκτή ώστε να ξεκινήσει τους υπολογισμούς της επόμενης στήλης κ.ο.κ. Η διαδικασία αυτή απεικονίζεται στο σχήμα 4.3β, όπου φαίνεται ότι:

- Η συντακτική ανάλυση λαμβάνει χώρα σε υλικό, οπότε ολοκληρώνεται ταχύτερα από ότι στο λογισμικό.
- Η αποτίμηση των κατηγορημάτων γίνεται ταυτόχρονα με την συντακτική ανάλυση και όχι ακολουθιακά μετά τον υπολογισμό όλων των κελιών του PT.
- Ο φόρτος της διαδικασίας αποτίμησης λαμβάνει χώρα σε δύο μικροελεγκτές, μειώνοντας το χρόνο που απαιτείται, λόγω της pipeline παραλληλοποίησης.

Ακολούθως, ένα παράδειγμα θα δοθεί από την περιοχή της αναπαράστασης της γνώσης με χρήση κατηγορικών γραμματικών. Έστω η περίπτωση που μια εφαρμογή αναζητά κατά πόσο υπάρχει σύνδεση στον κατευθυνόμενο ακυκλικό γράφο του σχήματος 4.4 ανάμεσα σε δύο κόμβους και εφόσον υπάρχει, πόσα τέτοια μονοπάτια υπάρχουν. Για ένα γράφο k κόμβων, που κάθε κόμβος είναι αριθμημένος από 0 μέχρι $k-1$, ορίζεται η συνάρτηση $connected(i, j)$ που είναι αληθής όταν υπάρχει μια κατευθυνόμενη ακμή από το i στο j. Ένα απλό λογικό πρόγραμμα για την αναζήτηση μονοπατιών από



Σχήμα 4.4: Κατευθυνόμενος Ακυκλικός Γράφος.

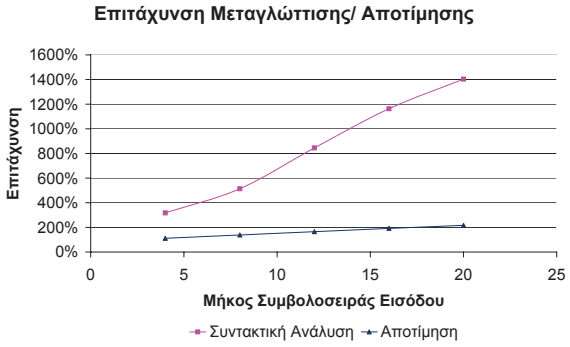
ένα δοσμένο κόμβο x σε έναν άλλο z παρουσιάζεται στην πρώτη στήλη του πίνακα 4.1. Οι συντακτικοί κανόνες της ισοδύναμης κατηγορικής γραμματικής παρουσιάζονται στη δεύτερη στήλη του πίνακα 4.1 ενώ οι σημασιολογικοί στην τρίτη στήλη του πίνακα 4.1. Στους συντακτικούς κανόνες ο στόχος αναπαρίσταται με G, το μονοπάτι με P και η σύνδεση με C. Για την παραπάνω γραμματική ελήφθησαν οι ακόλουθες

Λογικό πρόγραμμα	Συντακτικοί κανόνες	Σημασιολογικοί κανόνες
goal(x,y)← path(1,4)	0. G→P	$P.ia_1$=1; $P.ia_2$=4;
path(x,z)← path(y,z) ∧ connected(x,y)	1. P_1→CP_2	$C_1.ia_1$=$P_1.ia_1$; $P_2.ia_2$=$P_1.ia_2$1; $P_2.ia_1$=$P_1.ia_2$;
path(x,z)← connected(x,z)	3. P→C	$C.ia_1$=$P.ia_1$; $C.ia_2$=$P.ia_2$1;

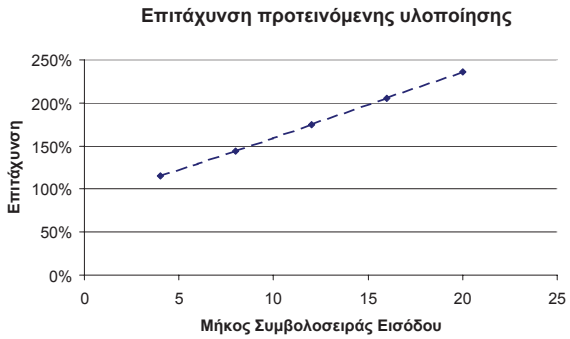
connected (1,2)	4. C→d	if (($C.ia_1$ ==1)
		OR($C.ia_1$ == NULL))
		then $C.sa_1$=1;
		else flag=0;
		
		if (($C.ia_2$ ==2)
		OR($C.ia_2$ == NULL))
		then $C.sa_1$=2;
		else flag=0;

connected (1,5)	5. C→d	...
connected (2,3)	6. C→d	...
...	...	...
connected (19,20)	30. C→d	...

Πίνακας 4.1: Αναζήτηση μονοπατιών σε ακυκλικό γράφο: Λογικό Πρόγραμμα και Κατηγορική Γραμματική.



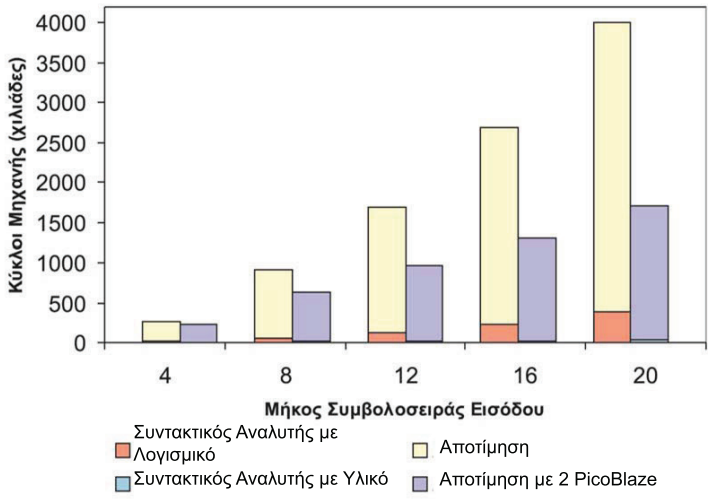
(a)



(b)

Σχήμα 4.5: a) Επιτάχυνση Ανάλυσης/Αποτίμησης b) Επιτάχυνση προτεινόμενης προσέγγισης εν συγκρίσει λογισμικού.

μετρήσεις κύκλων μηχανής: i) Συντακτικός αναλυτής σε λογισμικό, ii) Συντακτικός αναλυτής σε υλικό (υπολογισμός μόνον της πρώτης γραμμής), iii) Συντακτικός αναλυτής σε υλικό και διαδικασία αντιγραφής των κελιών κάθε διαγωνίου, iv) Αποτίμηση κατηγορημάτων με χρήση ενός μόνον επεξεργαστή (Pentium II 350 MHz) και v) Η προτεινόμενη προσέγγιση με χρήση δύο μικροελεγκτών PicoBlaze σε Xilinx Spartan-II FPGA. Στο σχήμα 4.5 παρουσιάζεται η επιτάχυνση για τη συντακτική ανάλυση, τη διαδικασία αποτίμησης και η συνολική επιτάχυνση ως προς την προσέγγιση σε λογισμικό. Ακόμη, στο σχήμα 4.6 συγκρίνεται σε ένα διάγραμμα η προσέγγιση σε υλικό με την αντίστοιχη σε λογισμικό. Δυστυχώς, λόγω της διαφοράς των τάξεων μεγεθών, μερικές μετρήσεις δεν φαίνονται ξεκάθαρα (κυρίως ο συντακτικός αναλυτής σε υλικό που απεικονίζεται κάτω από την αποτίμηση με χρήση 2 μικροελεγκτών).
Όπως φαίνεται από τα σχήματα 4.5 και 4.6, αν και η επιτάχυνση του αναλυτή είναι πολύ μεγάλη, η αντίστοιχη συνολική επιτάχυνση δεν είναι ανάλογη. Αυτό συμβαίνει, γιατί η απόδοση περιορίζεται από την διαδικασία αποτίμησης (bottleneck). Για την αντιμετώπιση αυτού του θέματος, υπάρχουν διάφορες προσεγγίσεις. Μια θα ήταν η χρήση περισσότερων μικροελεγκτών για την περαιτέρω παραλληλοποίηση της διαδικασίας. Μια δεύτερη είναι η χρήση ενός πολύ ισχυρού εξωτερικού επεξεργαστή μόνο



Σχήμα 4.6: Σύγκριση υλοποιήσεων σε λογισμικό και υλικό.

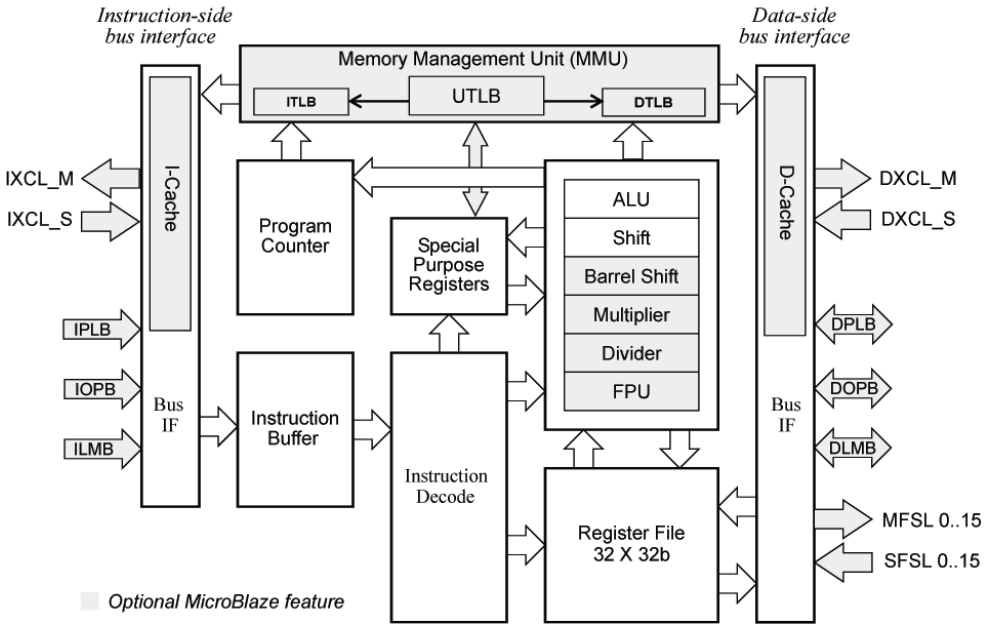
για τις αριθμητικές πράξεις. Μια ακόμη προσέγγιση θα ήταν να γίνει χρήση των λογικών κυκλωμάτων του FPGA, για την σχεδίαση ειδικών μονάδων που θα εκτελούν τις αναγκαίες λειτουργίες ταχύτατα. Τέλος, μπορεί να γίνει χρήση άλλου αλγορίθμου παράλληλης συντακτικής ανάλυσης, που να δίνει το συντακτικό δέντρο σε πιο βολική μορφή για την ταχεία αποτίμηση των κατηγορημάτων.
Η πρώτη προσέγγιση περιορίζεται από το μέγεθος του FPGA, ενώ η δεύτερη παραβιάζει την απαίτηση για ενσωματωμένο σύστημα μικρού μεγέθους. Η τρίτη προσέγγιση μοιάζει η πιο ελπιδοφόρα, για το λόγο αυτό επιλέχθηκε για περισσότερη διερεύνηση η χρήση εξειδικευμένων λογικών κυκλωμάτων ενώ πάντα αναζητάται στην βιβλιογραφία αλγόριθμος συντακτικής ανάλυσης με χαρακτηριστικά καταλληλότερα για την αποτίμηση κατηγορικών γραμματικών. Πάντως, μια ενδιαφέρουσα λύση είναι η αντικατάσταση του PicoBlaze από τον soft core επεξεργαστή MicroBlaze, η οποία παρουσιάζεται ακολούθως.

4.2 Αντικατάσταση Εξωτερικού Επεξεργαστή με Μικροεπεξεργαστή MicroBlaze

Όπως φάνηκε και στα προηγούμενα, οι επιδόσεις του μικροελεγκτή είναι σχετικά περιορισμένες τόσο από πλευράς ταχύτητας όσο και από πλευράς δυνατοτήτων. Αντί για εσωτερικό μικροελεγκτή, μπορεί να προτιμηθεί η επιλογή του εσωτερικού μικροεπεξεργαστή. Ακολουθεί μια ανάλυση των ιδιοτήτων του επεξεργαστή που τελικώς επιλέγει, του soft core MicroBlaze. Ο επεξεργαστής MicroBlaze είναι ένας 32-bit RISC επεξεργαστής soft core, βελτιστοποιημένος για υλοποίηση σε FPGA 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 απεικονίζεται και σχηματικά στο σχήμα 4.7, 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 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 4.7: Μπλοκ Διάγραμμα πυρήνα MicroBlaze.

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

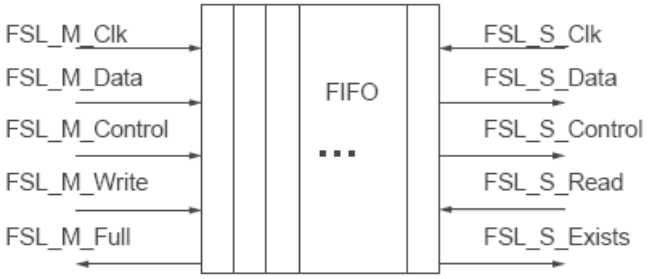
- Μονάδα κινητής υποδιαστολής (FPU).
- Διαιρετής hardware.
- Πολλαπλασιαστής hardware.
- Δυνατότητα ολίσθησης πολλών bit σε έναν κύκλο (barrel shift).
- Γρήγορη μνήμη (cache) τόσο για τις εντολές όσο και για τα δεδομένα.

Ειδικά για τη μονάδα κινητής υποδιαστολής του MicroBlaze, βασίζεται στο πρότυπο IEEE 754 και επομένως:

- Χρησιμοποιεί το μοντέλο IEEE 754 απλής ακρίβειας και περιλαμβάνει ορισμούς για το άπειρο, το μηδέν και τους όχι αριθμούς (Not-A-Number).
- Υποστηρίζει εντολές πρόσθεσης, αφαίρεσης, πολλαπλασιασμού, διαίρεσης και σύγκρισης.
- Υλοποιεί στρογυλλοποίηση στον κοντινότερο αριθμό.
- Παράγει bit κατάστασης για καταστάσεις υπερχείλισης, υποχείλισης και λάθους λειτουργίας.

Ένας από τους βασικούς λόγους επιλογής του MicroBlaze στην προτεινόμενη υλοποίηση ήταν η ύπαρξη των Fast Simplex Link (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 χαμηλής λανθάνουσας περιόδου (latency) με τον επεξεργαστή. Συνεπώς είναι ιδανικά για την επέκταση των λειτουργιών του επεξεργαστή με χρήση προσαρμοσμένων τμημάτων υλικού. Η απόδοση της διεπαφής FSL μπορεί να φτάσει μέχρι και τα 300MB/sec, η οποία όμως 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 .



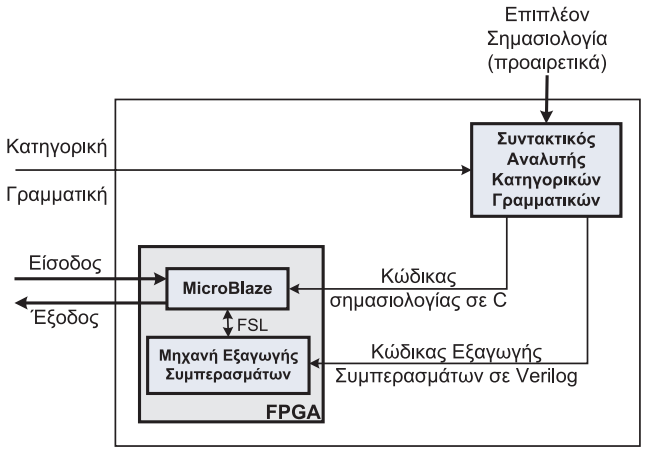
Σχήμα 4.8: Διεπαφή του FSL.

Ο διάδρομος FSL οδηγείται από ένα Master και οδηγεί ένα Slave. Στο σχήμα 4.8 φαίνεται η αρχή λειτουργίας του FSL και τα διαθέσιμα σήματα. Τα FSL περιφερειακά μπορούν να δημιουργηθούν είτε σαν Master είτε σαν Slave. Ένα περιφερειακό συνδεδεμένο σαν Master στέλνει δεδομένα και σήματα ελέγχου στο FSL. Ένα περιφερειακό

συνδεδεμένο σαν Slave διαβάζει δεδομένα και λαμβάνει σήματα ελέγχου από το FSL. Υπάρχουν προκαθορισμένες συναρτήσεις C που είναι διαθέσιμες στο χρήστη για την ευκολότερη χρησιμοποίηση του FSL. Παράδειγμα τέτοιων είναι οι εντολές put και get του MicroBlaze που μπορούν να χρησιμοποιηθούν για τη μεταφορά των περιεχομένων ενός καταχωρητή του επεξεργαστή στο διάδρομο FSL και αντίστροφα. Επομένως, εκτός από την ταχύτητα που προσφέρει συγκριτικά με τον μικροελεγκτή, υπερτερεί στην γλώσσα περιγραφής των εφαρμογών. Υποστηρίζεται η γλώσσα C και παρέχεται ειδικός μεταγλωττιστής που παράγει εκτελέσιμα αρχεία για τον επεξεργαστή.
Μια επιπλέον παράμετρος που πρέπει να εξεταστεί και να αξιολογηθεί για τη χρήση του ενσωματωμένου επεξεργαστή είναι η επιλογή του λειτουργικού συστήματος το οποίο θα εκτελείται στον MicroBlaze. Τυπικές εφαρμογές σε ένα ενσωματωμένο σύστημα αποτελούνται από διάφορες εργασίες οι οποίες πρέπει να εκτελεστούν με συγκεκριμένη σειρά ή σε συγκεκριμένο χρόνο. Όσο αυξάνει ο αριθμός των εργασιών αυτών, δυσκολεύει εξαιρετικά η χειροκίνητη οργάνωσή τους προκειμένου να γίνεται σωστός και δίκαιος χρονικός διαμοιρασμός (time-sharing) του επεξεργαστή. Επιλέγοντας την χρησιμοποίηση λειτουργικού συστήματος, οι διάφορες εργασίες μπορούν να αντιστοιχιθούν σε ξεχωριστά προγράμματα στα οποία κατανέμει το λειτουργικό σύστημα την ΚΜΕ, ανάλογα με τον αλγόριθμο χρονοδρομολόγησης που έχει επιλεγεί. Επιπλέον, η χρήση λειτουργικού συστήματος καθιστά ευκολότερη τη συγγραφή κώδικα, αφού όλα γίνονται πια σε πιο αφαιρετικό επίπεδο και όχι σε χαμηλό επίπεδο ειδικά για τον συγκεκριμένο επεξεργαστή. Παράλληλα, η ύπαρξη λειτουργικού συστήματος επιτρέπει τη χρήση κοινών υπηρεσιών όπως το σύστημα αρχείων ή πιο εξειδικευμένων όπως η στοίβα TCP από τα προγράμματα, χωρίς να χρειάζεται ο χρήστης να μπει στις λεπτομέρειες και να γράψει εξειδικευμένο κώδικα.
Όπως αναφέρθηκε ήδη, ο MicroBlaze είναι παραμετροποιήσιμος σε πολύ μεγάλο βαθμό, δίνοντας τη δυνατότητα να επιτευχθεί εξοικονόμηση χώρου εις βάρος βέβαια των συνολικών δυνατοτήτων και άρα και της ταχύτητας. Δίνεται μάλιστα η δυνατότητα για υλοποίηση ή μη μονάδας ελέγχου της μνήμης (memory management unit - MMU). Εάν λείπει η MMU, τα υποστηριζόμενα λειτουργικά συστήματα πρέπει να έχουν κάποιο απλοποιημένο μοντέλο για προστασία της μνήμης και για την εικονική μνήμη, όπως π.χ. τα uClinux [62] και FreeRTOS [63]. Στην αντίθετη περίπτωση που υπάρχει η MMU, ο MicroBlaze είναι ικανός να "φιλοξενήσει" λειτουργικά συστήματα που απαιτούν από το σύστημα προστασία και έλεγχο της μνήμης, όπως π.χ. ο πυρήνας του Linux, αν και η συνολική απόδοση του επεξεργαστή είναι πολύ κατώτερη από εκείνη ενός αντίστοιχου hard core επεξεργαστή, όπως π.χ. ο PowerPC405. Σε κάθε περίπτωση, η εταιρεία Xilinx παρέχει δικό της λειτουργικό σύστημα XilKernel [64] για το συγκεκριμένο επεξεργαστή, το οποίο όμως είναι κλειστό και επομένως όχι παραμετροποιήσιμο κατά τις ανάγκες του χρήστη.

4.2.1 Υλοποίηση με MicroBlaze

Το σύστημα που παρουσιάστηκε στην ενότητα 4.1.1 περιοριζόταν από την ταχύτητα του μικροελεγκτή. Για το λόγο αυτό αντικαταστήθηκε σε επόμενη υλοποίηση από τον επεξεργαστή MicroBlaze. Επιπλέον, επειδή τώρα οι εντολές που περιγράφουν τον τρόπο αποτίμησης των κατηγορημάτων είναι σε γλώσσα C, αυτοματοποιήθηκε η όλη διαδικασία, ώστε να ακολουθεί την προτεινόμενη αρχιτεκτονική του σχήματος 4.9. Ο χρήστης δίνει την κατηγορική γραμματική, συντακτικούς κανόνες και σημασιολογικούς, σε ένα αυτόματο εργαλείο, το οποίο παράγει αυτόματα τον κώδικα που περιγράφει μια πλατφόρμα που αναγνωρίζει συμβολοσειρές της συγκεκριμένης γραμματικής και αποτιμά τις τιμές των κατηγορημάτων. Η έξοδος του αυτόματου εργαλείου, αποτελείται από κώδικα σε Verilog, που περιγράφει το συντακτικό αναλυτή και κώδικα σε C, που περιγράφει το σημασιολογικό κομμάτι. Ο κώδικας αυτός μεταγλωττίζεται και "κατεβαίνει" στο FPGA. Έτσι, τώρα πια το FPGA έχει δύο βασικές μονάδες που επικοινωνούν μεταξύ τους μέσω FSL, τη μηχανή εξαγωγής συμπερασμάτων και το μικροεπεξεργαστή. Κάθε φορά που χρειάζεται να εκτελεστεί μια πράξη από τον επεξεργαστή, δίνονται πάνω σε ένα FSL οι τελεσταίοι και επιστρέφεται το αποτέλεσμα από το άλλο FSL αντίστροφης κατεύθυνσης. Σημειώνεται, ότι για την επικοινωνία του συστήματος με τον εξωτερικό κόσμο χρησιμοποιείται η σειριακή θύρα RS-232 του FPGA, ενώ για μεγαλύτερο ρυθμό αποστολής δεδομένων μπορεί να χρησιμοποιηθεί και η ενσωματωμένη κάρτα δικτύου (NIC).
Επιπλέον σημασιολογικοί κανόνες μπορούν να προστεθούν, αυξάνοντας τις δυνατότητες εξαγωγής συμπερασμάτων, ξεπερνώντας τις δυνατότητες της PROLOG, οδηγώντας σε υλοποιήσεις όπως μηχανές απόδειξης θεωρημάτων (theorem provers) [48], μεταγλωττιστές οδηγούμενους από σημασιολογία (semantically driven parsers) [57] και μηχανές εξαγωγής συμπερασμάτων με ασάφεια (fuzziness) και αβεβαιότητα (uncertainty) [52].



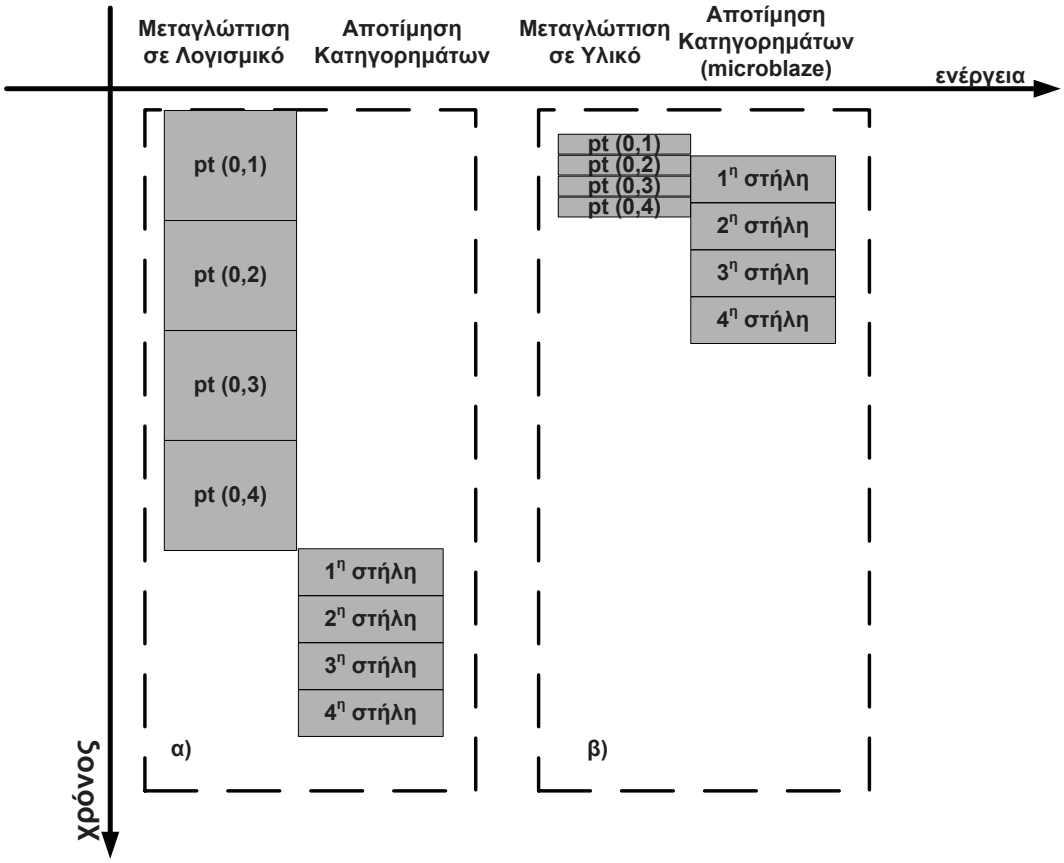
Σχήμα 4.9: Προτεινόμενη αρχιτεκτονική με χρήση MicroBlaze.

Το συγκεκριμένο σύστημα υλοποιήθηκε στο περιβάλλον XILINX ISE 8.2i [65] ενώ ο παραγόμενος κώδικας εξομοιώθηκε (simulated), συντέθηκε (synthesized) και δοκιμάστηκε (tested) σε ένα Xilinx SPARTAN 3E FPGA. Στο σχήμα 4.10 φαίνεται πώς μετασχηματίζεται η αρχιτεκτονική από εκείνη που κάνει χρήση λογισμικού, στην προτεινόμενη που κάνει χρήση του MicroBlaze. Όπως και στην περίπτωση του PicoBlaze:

- Η συντακτική ανάλυση λαμβάνει χώρα σε υλικό, οπότε ολοκληρώνεται ταχύτερα από ότι στο λογισμικό.
- Η αποτίμηση των κατηγορημάτων γίνεται ταυτόχρονα με την συντακτική ανάλυση και όχι ακολουθιακά μετά τον υπολογισμό όλων των κελιών του PT.

Τώρα όμως, η ύπαρξη του επεξεργαστή επιτρέπει την ταχύτερη επεξεργασία εν συγκρίσει με τον μικροελεκτή αλλά κυρίως, δεν υπάρχει περιορισμός ως προς την πολυπλοκότητα της διαδικασίας υπολογισμού των κατηγορημάτων. Η συγκεκριμένη υλοποίηση δοκιμάστηκε για πλειάδα εφαρμογών [28], [45], [46], [47], [36] και έδωσε κατά μέσο όρο επιτάχυνση μιας τάξης μεγέθους, σε σύγκριση με τη προσέγγιση του λογισμικού.

Όσον αφορά στην επιλογή ή όχι λειτουργικού συστήματός, λόγω της χρησιμοοίησης των διασυνδέσεων FSL κρίθηκε απαραίτητη η χρήση του. Προκειμένου να μπορούν να διαχωριστούν απλά και εύκολα οι ενέργειες στις οποίες πρέπει να προβαίνει το σύστημα κατά την λήψη δεδομένων από τον συντακτικό αναλυτή καθώς και κατά την αποστολή των επιβεβαιώσεων, θεωρήθηκε ότι η ύπαρξη λειτουργικού συστήματος θα απλοποιούσε σημαντικά το προγραμματιστικό κομμάτι, όπως και έγινε. Δοκιμάστηκαν τρία λειτουργικά συστήματα πάνω στον MicroBlaze: το uCLinux, το MicroEmpix και το XilKernel. Αρχικά δοκιμάστηκε το uCLinux, το οποίο είναι από τις πιο δημοφιλείς επιλογές στο χώρο των ενσωματωμένων συστημάτων καθώς διέπεται από τους όρους του GPL. Προφανώς υπερκαλύπτει τις ανάγκες της συγκεκριμένης εφαρμογής, δίνοντας δυνατότητες για πολυδιεργασία καθώς και μια βιβλιοθήκη standard C (uClibc) που διευκολύνει τη μεταφορά προγραμμάτων από άλλους επεξεργαστές. Η δεύτερη επιλογή ήταν η μεταφορά του εκπαιδευτικού λειτουργικού συστήματος MicroEmpix στον επεξεργαστή MicroBlaze [66]. Το MicroEmpix αν και είναι σχετικά μικρό σε μέγεθος, διαθέτει τα πιο σημαντικά χαρακτηριστικά των πιο περίπλοκων λειτουργικών συστημάτων, όπως πολυδιεργασία, συγχρονισμό διεργασιών με χρήση σηματοφορέων και λειτουργίες E/E με χρήση περιφερειακών μονάδων. Η μεταφορά του στον MicroBlaze εξακολουθεί να υποστηρίζει όλα τα παραπάνω και επιπλέον υποστηρίζει τη χρήση εξειδικευμένων τμημάτων υλικού συνδεδεμένων στη διεπαφή FSL του επεξεργαστή MicroBlaze. Εκτός από τις δυνατότητες επιτάχυνσης υλικού, το MicroEmpix/FPGA υποστηρίζει λειτουργίες E/E με τη χρήση Universal Asynchronous Receiver Transmitter (UART) και λειτουργίες μέτρησης με τη χρήση



Σχήμα 4.10: Σύγκριση της προτεινόμενης αρχιτεκτονικής με χρήση MicroBlaze (β) με την υλοποίηση σε λογισμικό (α).

ενός μετρητή. Γενικότερα είναι εξαιρετικά κατανοητό όσον αφορά στην υλοποίησή του και συνεπώς σε μεγάλο βαθμό ρυθμιζόμενο και επεκτάσιμο. Εντέλει όμως, αν και τα δύο παραπάνω λειτουργικά συστήματα πληρούσαν τις απαιτήσεις για χρήση στην προτεινόμενη εφαρμογή, καθαρά για λόγους απλότητας[1] επιλέχθηκε το XilKernel για τη συγκεκριμένη εφαρμογή. Εντούτοις, για πιο σύνθετες εφαρμογές η χρήση ενός πιο ευέλικτου και open-source λειτουργικού συστήματος είναι μάλλον αναγκαία.

[1] Ο κώδικας μεταγλωττίζεται μέσα από το περιβάλλον XPS της Xilinx χωρίς να χρειάζεται η χρήση άλλου μεταγλωττιστή και βιβλιοθηκών.

Κεφάλαιο 5

Υλοποίηση Με Υλικό Συντακτικού Αναλυτή Για Γραμματικές Χωρίς Συμφραζόμενα

Σε μια προσπάθεια να βελτιωθεί περαιτέρω η απόδοση του συντακτικού αναλυτή, υλοποιήθηκε μια καινούργια αρχιτεκτονική που κάνει χρήση συνδυαστικών κυκλωμάτων για την αναγνώριση συμβολοσειρών που ανήκουν σε γραμματικές χωρίς συμφραζόμενα. Ακριβώς στην ύπαρξη των συνδυαστικών κυκλωμάτων οφείλει την απόδοσή της η νέα αρχιτεκτονική, μέσω των οποίων υλοποιείται ο θεμελιώδης τελεστής $\otimes$ με πολυπλοκότητα $O(log_2|G|)$, όπου $|G|$ είναι το μέγεθος της γραμματικής χωρίς συμφραζόμενα. Η αρχιτεκτονική έχει σχεδιαστεί πλήρως παραμετροποιήσιμη, ώστε με χρήση ενός αυτόματου εργαλείου να παράγεται για κάθε CFG γραμματική ο HDL κώδικας που περιγράφει το συντακτικό αναλυτή. Η μέθοδος αυτή, αυξάνει την επίδοση κατά έναν παράγοντα μίας ή δύο τάξεων μεγέθους, σε σχέση με τις υλοποιήσεις σε υλικό που έχουν παρουσιαστεί στα προηγούμενα κεφάλαια, ανάλογα με το μέγεθος της γραμματικής και το μήκος της συμβολοσειράς εισόδου. Όσον αφορά τώρα την επιτάχυνση ως προς το λογισμικό, ξεκινά από δύο τάξεις μεγέθους για απλές εφαρμογές και φτάνει τις έξη για μεγάλες γραμματικές πραγματικών εφαρμογών.

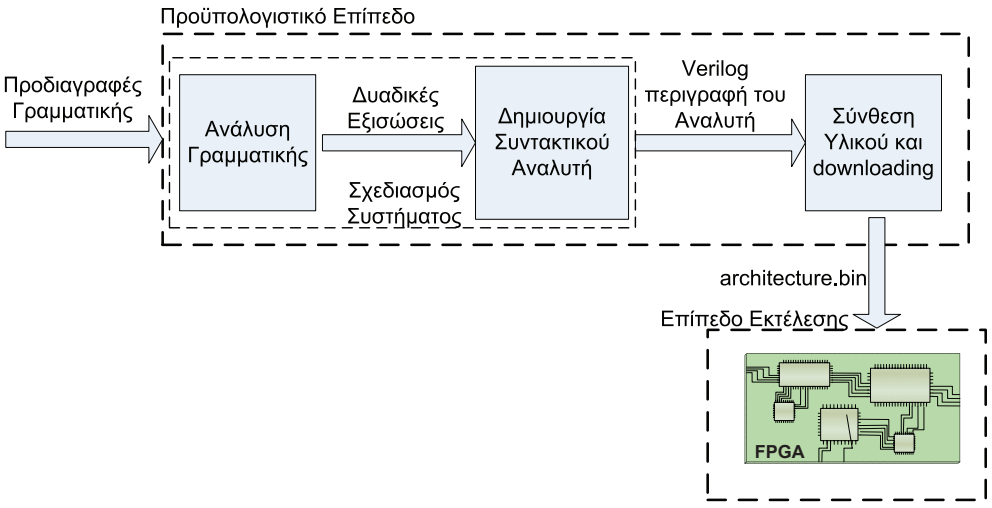
5.1 Γενικά

Προκειμένου να αντιμετωπιστούν οι απαιτήσεις για γρήγορη απόκριση σε πραγματικό χρόνο, οι σχεδιαστές πρέπει να βελτιστοποιούν τους αναλυτές τους. Δύο είναι οι βασικοί παράγοντες που επηρεάζουν την απόδοση ενός αναλυτή:

1. Ο αλγόριθμος που ακολουθείται.

2. Η αρχιτεκτονική υλοποίησης, δηλαδή ακολουθιακή ή παράλληλη, σε υλικό ή σε λογισμικό.

Αναφορικά με τον πρώτο παράγοντα, πολλές τροποποιήσεις [14], [15] και βελτιώσεις μέσω παραλληλοποίησης [16] έχουν προταθεί για τους κλασικούς πλέον αλγορίθμους των Cocke-Younger-Kasami (CYK) [17], [18] και του Earley [19]. Ο αλγόριθμος CYK απαιτεί η γραμματική να είναι σε μορφή CNF (Chomsky Normal Form). Κάθε γραμματική χωρίς συμφραζόμενα μπορεί να μετατραπεί σε CNF μορφή, αλλά όχι χωρίς τίμημα: για μια γραμματική μεγέθους $|G|$, θα χρειαστεί χρόνος $O(|G|^2)$ [20], [21], [22]. Βέβαια, ο μετασχηματισμός αυτός γίνεται μια μόνο φορά, οπότε φαινομενικά δεν επηρεάζει την απόδοση του αναλυτή. Όμως, εάν η νέα γραμματική G' μετά το μετασχηματισμό είναι μεγαλύτερη από την G (στην χειρότερη περίπτωση $|G'| = |G|^2$), η απόδοση του αναλυτή θα επηρεαστεί αφού εξαρτάται από το μέγεθος της γραμματικής ($|G'|$). Επιπλέον, η έλλειψη ανάγκης μετασχηματισμού στην υλοποίησή μας, σε συνδυασμό με την προτεινόμενη αυτοματοποίηση της διαδικασίας σχεδίασης, περιορίζει τον προϋπολογιστικό (precomputational) χρόνο, μόνον σε αυτόν της σύνθεσης του υλικού (βλέπε σχήμα 5.1).



Σχήμα 5.1: Αυτοματοποιημένη διαδικασία υλοποίησης.

Αναφορικά με το δεύτερο παράγοντα, οι Chiang & Fu [16] και Cheng & Fu [23] παρουσίασαν υλοποιήσεις που έκαναν χρήση συστοιχειών VLSI για την υλοποίηση των παραπάνω αλγορίθμων, ενώ οι Ibbara et al. [24] και Ra et al. [25] παρουσίασαν παράλληλες υλοποιήσεις λογισμικού που εκτελούνται σε παράλληλα μηχανήματα. Ο προσανατολισμός σε προσεγγίσεις υλικού αναζωογονήθηκε με την παρουσίαση σε FPGA του αλγορίθμου των CYK (Ciressan [26] και Bordim et al. [27]) και του αλγορίθμου του Earley (Pavlatos et al [28]). Οι προτεινόμενες μέχρι τώρα αρχιτεκτονικές είτε αποτυγχάνουν να εκμεταλλευτούν στο έπακρο τη διαθέσιμη παραλληλοποίηση των αλγορίθμων ή απαιτούν υπερβολικούς αποθηκευτικούς πόρους. Η υλοποίηση σε

λογισμικό, εκτελεί μέρη του αλγορίθμου συντακτικής ανάλυσης ακολουθιακά και έτσι αποτυγχάνει να πετύχει τη μέγιστη δυνατή επιτάχυνση. Από την πλευρά του υλικού, οι υπάρχουσες μεθοδολογίες έρχονται αντιμέτωπες με την πολυπλοκότητα που επιβάλλεται από τους χρησιμοποιούμενους τελεστές των αλγορίθμων, κάτι που οδηγεί στην αυξημένη ανάγκη για χώρο. Προκειμένου να ελαττωθεί η πολυπλοκότητα, οι περισσότερες υλοποιήσεις σε υλικό επιλέγουν τον αλγόριθμο CYK, του οποίου οι βασικές πράξεις είναι πολύ πιο απλές από αυτές του Earley. Η προτεινόμενη υλοποίηση κάνει χρήση συνδυαστικών κυκλωμάτων για τις θεμελιώδεις πράξεις του αλγορίθμου του Earley.
Συνοπτικά, τα χαρακτηριστικά της προτεινόμενης υλοποίησης είναι:

- Η αρχιτεκτονική της απαιτεί λιγότερα υπολογιστικά στοιχεία, σε σχέση με τις προηγούμενες, κατά ένα παράγοντα $\frac{(n+1)(n+2)}{2n}$, όπου n είναι το μήκος της συμβολοσειράς εισόδου, ενώ παράλληλα διατηρείται ένας απλός και κομψός τρόπος επικοινωνίας. Επιπλέον, ορίζεται ένα συνδυαστικό κύκλωμα $C_{\otimes}$ το οποίο υλοποιεί το θεμελιώδη τελεστή $\otimes$ σε εξαιρετικά μικρό χρόνο, ανάλογο με το χρόνο διάδοσης (propagation delay) μερικών λογικών πυλών. Κάνοντας χρήση των ελάχιστων δυνατών κυκλωμάτων $C_{\otimes}$, υπολογίζεται κάθε κελί του συντακτικού πινάκα PT παράλληλα. Με τον τρόπο αυτό, επιτυγχάνονται δύο επίπεδα παραλληλίας: ένα τοπικό (επίπεδο κελιού), που αντιστοιχεί στην εκτέλεση του $\otimes$ και ένα γενικό (επίπεδο αρχιτεκτονικής), το οποίο αντιστοιχεί στην μορφή του αλγορίθμου η οποία αποτελείται από πίνακες. Ταυτόχρονα, ο τρόπος υλοποίησης καταργεί την ανάγκη για αποθήκευση της γραμματικής, καθώς μέσω ενός συστήματος δυαδικών εξισώσεων τα χαρακτηριστικά της γραμματικής ενσωματώνονται στο συνδυαστικό κύκλωμα.

- Η προτεινόμενη αρχιτεκτονική βελτιώνει την απόδοση του θεμελιώδους τελεστή $\otimes$ ελαττώνοντας την χρονική πολυπλοκότητά του σε $O(log_2(|G|))$. Επιπροσθέτως, η συνολική πολυπλοκότητα της συντακτικής ανάλυσης ελαττώνεται σε $O(nlog_2\ (n|G|))$.

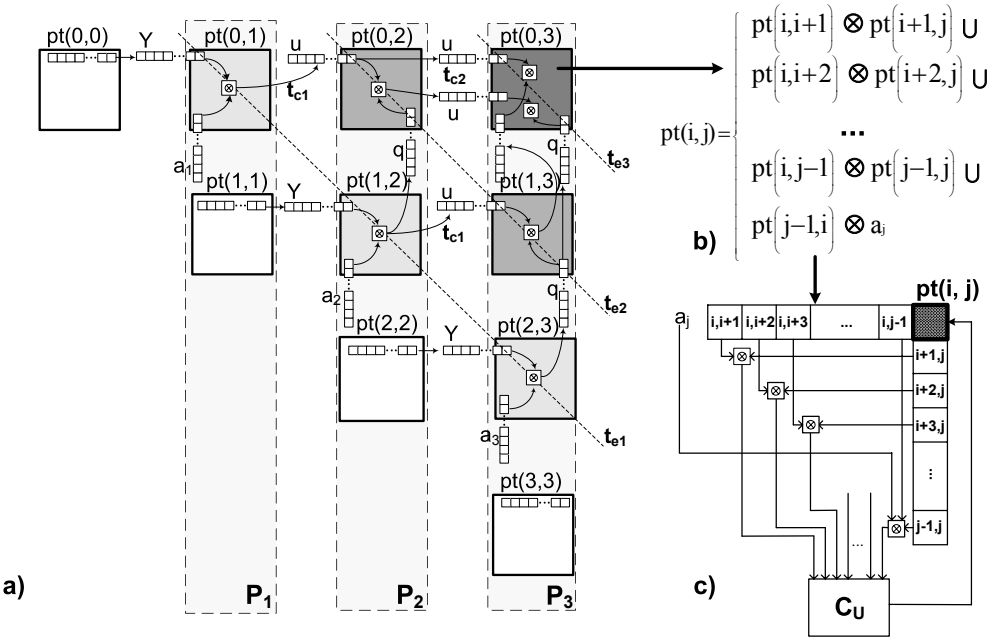
- Δεν απαιτείται η γραμματική να μην έχει κανόνες παραγωγής που να παράγουν το κενό (ε-free). Η απαίτηση για το αντίθετο είναι σημαντικός περιορισμός, αφού ο μετασχηματισμός αυξάνει το πλήθος των κανόνων. Ακόμη δεν υπάρχει απαίτηση για τους κανόνες να είναι σε μορφή CNF (Chomsky Normal Form).

- Τέλος, η αρχιτεκτονική έχει υλοποιηθεί σε παραμετροποιήσιμη μορφή και ένα αυτόματο εργαλείο έχει αναπτυχθεί, το οποίο για δεδομένη γραμματική παράγει την περιγραφή του συντακτικού αναλυτή σε Verilog. Έτσι, ο χρόνος που απαιτείται από το σχεδιαστή για την υλοποίηση ενός συστήματος, ελαττώνεται δραματικά.

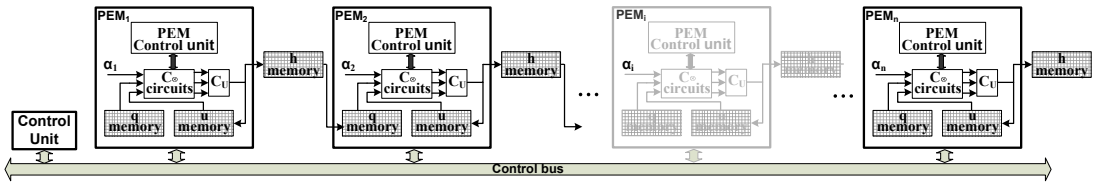
5.2 Προτεινόμενη Αρχιτεκτονική

Όπως έχει ήδη αναφερθεί αρκετές φορές μέχρι τώρα, η διαδικασία υπολογισμού των περιεχομένων κάθε κελιού του πίνακα PT βασίζεται στην εξίσωση 2.1, η οποία για ευκολία ξαναγράφεται στο σχήμα 5.2b. Τα περιεχόμενα λοιπόν των κελιών γεμίζουν με εφαρμογή του προτεινόμενου κυκλώματος $C_{\otimes}$. Το $C_{\otimes}$ κατασκευάζεται βάσει των "χαρακτηριστικών" εξισώσεων κάθε γραμματικής. Οι εξισώσεις αυτές, όπως θα αναλυθεί και πιο κάτω στην ενότητα 5.3.2, παράγονται αλγοριθμικά μέσω συγκεκριμένης διαδικασίας. Όλες οι πράξεις $\otimes$ μπορούν να εκτελεστούν παράλληλα με χρήση πολλαπλών κυκλωμάτων $C_{\otimes}$ και η λογική ένωση (OR) των αποτελεσμάτων παράγει το τελικό περιεχόμενο του κελιού $pt(i,j)$, όπως απεικονίζεται και στο σχήμα 5.2c. Επομένως, μπορούν να οριστούν δύο επίπεδα παραλληλίας: ένα τοπικό ή επίπεδο κελιού, που αντιστοιχεί στην παράλληλη εκτέλεση των τελεστών $\otimes$ στο εσωτερικό κάθε κελιού και ένα γενικό ή επίπεδο αρχιτεκτονικής, που αντιστοιχεί στην παράλληλη συμπλήρωση του πίνακα PT. Η προτεινόμενη αρχιτεκτονική, επιτυγχάνει αναγνώριση της συμβολοσειράς εισόδου μήκους n σε χρόνο $O(nlog_2(n|G|))$, όπως αποδεικνύεται στην ενότητα 5.3.3.

Ένας επιπλέον στόχος, είναι η δημιουργία ενός εργαλείου που θα λαμβάνει σαν είσοδο τις προδιαγραφές της γραμματικής και θα παράγει τις χαρακτηριστικές εξισώσεις του τελεστή $C_{\otimes}$ καθώς και όλη την αρχιτεκτονική σε Verilog του αντίστοιχου συντακτικού αναλυτή. Για να επιτευχθεί αυτό, η προτεινόμενη αρχιτεκτονική είναι τελείως ανεξάρτητη από την γραμματική αλλά αποτελείται από γενικής χρήσης δομικά στοιχεία (μνήμη, πολυπλέκτες, λογικές πύλες κτλ) και μπορεί δυναμικά να τροποποιηθεί



Σχήμα 5.2: Αφηρημένη αρχιτεκτονική για συμβολοσειρά εισόδου μήκους n=3.



Σχήμα 5.3: Μπλοκ Διάγραμμα αρχιτεκτονικής.

για κάθε γραμματική. Μόλις ο χρήστης δώσει τις προδιαγραφές της γραμματικής, το αυτόματο εργαλείο παράγει την περιγραφή του συντακτικού αναλυτή σε υλικό. Είναι σημαντικό να τονισθεί ότι η αρχιτεκτονική που παρουσιάζεται στο σχήμα 5.3 δεν τροποποιείται, αλλά ανάλογα με τις προδιαγραφές του χρήστη, το εργαλείο παράγει υπολογιστικά στοιχεία (PEM) και τα προσθέτει στο πρότυπο (template). Το πρότυπο αυτό, έχει υλοποιηθεί σε Verilog ενώ το αυτόματο εργαλείο σε C. Η σημασία του αυτοματοποιημένου εργαλείου είναι ότι μειώνει δραστικά το χρόνο σχεδιασμού, εκμεταλλεύεται τη δυνατότητα επαναπρογραμματισμού του FPGA και δίνει τη δυνατότητα να κατασκευαστεί για το ίδιο FPGA αναλυτής για άλλη γραμματική δυναμικά (π.χ. αλλαγή από τη μια γλώσσα στην άλλη, σε διεπαφές χρήστη για εφαρμογές που απαιτούν αναγνώριση υποσυνόλων φυσικών γλωσσών). Περισσότερες πληροφορίες για το αυτόματο εργαλείο δίνονται στο παράρτημα Β.

5.3 Το συνδυαστικό κύκλωμα $C_{\otimes}$

Ένα συνδυαστικό κύκλωμα είναι μια γενικευμένη πύλη m εισόδων και l εξόδων. Το κύκλωμα παράγεται με τη χρήση δυαδικών εξισώσεων που περιγράφουν τη σχέση μεταξύ των τιμών των εισόδων και των εξόδων. Επομένως, προκειμένου να κατασκευαστεί ένα συνδυαστικό κύκλωμα για τον τελεστή $\otimes$, πρέπει να οριστούν οι είσοδοι και έξοδοι του κυκλώματος, καθώς και οι δυαδικές εξισώσεις. Σύμφωνα με την εξίσωση 2.1, ο τελεστής $\otimes$ δρα επί δύο συνόλων κανόνων παραγωγής ή επί ενός συνόλου κανόνων παραγωγής και ενός τερματικού συμβόλου. Συνεπώς, οι είσοδοι και έξοδοι του κυκλώματος θα είναι κατάλληλες κωδικοποιήσεις αυτών των συνόλων και των τερματικών συμβόλων. Με τον όρο κατάλληλες, εννοείται δυαδική κωδικοποίηση που μπορεί να περιγράψει σε κάθε χρονική στιγμή ποιοι κανόνες και ποια τερματικά σύμβολα περιέχονται στα σύνολα. Η αναπαράσταση των δεδομένων καθώς και οι δυαδικές εξισώσεις παρουσιάζονται στις ακόλουθες υποενότητες.

5.3.1 Αναπαράσταση δεδομένων

Ορισμός 4 *Κάθε κανόνας r που ανήκει στο σύνολο συμβολίζεται με r_i, όπου i $(1 \leq i \leq |P|)$ είναι ο αύξων αριθμός του κανόνα. Για το συμβολισμό ενός κανόνα παραγωγής με τελεία, ο παραπάνω συμβολισμός επεκτείνεται σε r_i^j, όπου j είναι η*

θέση της τελείας στον κανόνα r_i $(1 \leq j \leq |r_i|)$.

Ορισμός 5 *Για το τερματικό σύμβολο* a *και τα σύνολα των κανόνων παραγωγής με τελεία* Q *και* U*, μπορεί να οριστεί ο τελεστής* $\otimes$ *με χρήση των βοηθητικών συνόλων* h_1*,* h_2 *και* h_Y*:*

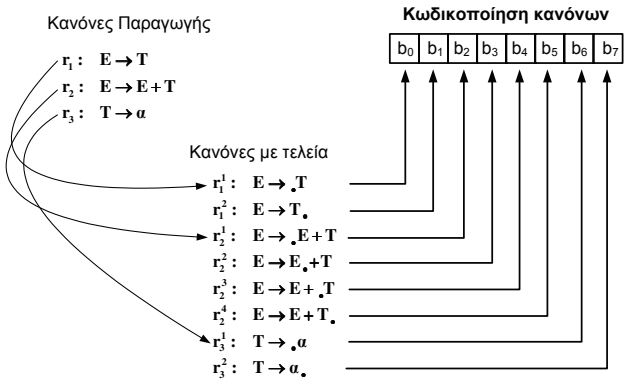
- $h_1(Q,a) = \{A \rightarrow \delta a_\bullet \beta\gamma | A \rightarrow \delta_\bullet a\beta\gamma \in Q\} \bigcup \{A \rightarrow \delta a\beta_\bullet\gamma | A \rightarrow \delta_\bullet a\beta\gamma \in Q$ *εάν* $\beta \xrightarrow{*} \varepsilon$ $\}$
- $h_2(Q,U) = \{A \rightarrow \alpha E_\bullet \beta\gamma | A \rightarrow \alpha_\bullet E\beta\gamma \in Q$ *και* $E \rightarrow \zeta_\bullet \in U$ $\} \bigcup \{A \rightarrow \alpha E\beta_\bullet\gamma | A \rightarrow \alpha_\bullet E\beta\gamma \in Q$ *και* $E \rightarrow \zeta_\bullet \in U$*, εάν* $\beta \xrightarrow{*} \varepsilon\}$

Έστω ένα σύνολο X *με κανόνες παραγωγής με τελείες, τότε το σύνολο* $h_Y(X)$ *ορίζεται ως εξής:*

- $h_Y(X) = \{B \rightarrow \delta C_\bullet \xi\eta | D \rightarrow \kappa_\bullet \in X$, $C \in Predecessors(D)$, $B \rightarrow \delta_\bullet\ C\xi\eta \in Predict(N)\} \bigcup \{B \rightarrow \delta C\xi_\bullet\eta | D \rightarrow \kappa_\bullet \in X$, $C \in Predecessors(D)$, $B \rightarrow \delta_\bullet C\xi\eta \in Predict(N)$ *εάν* $\xi \xrightarrow{*} \varepsilon\}$
- $Q \otimes a = h_1(Q,a) \cup h_Y(h_1(Q,a))$ *και*
- $Q \otimes U = h_2(Q,U) \cup h_Y(h_2(Q,U))$

Λόγω της φύσης του υλικού, όλα τα δεδομένα, δηλαδή οι κανόνες παραγωγής με τελεία, πρέπει να αναπαρασταθούν με διανύσματα δυαδικών ψηφίων (bit vectors). Έστω ότι τα διανύσματα $\vec{q}$ και $\vec{u}$ απεικονίζουν τα σύνολα εισόδου Q και U, ενώ το διάνυσμα $\vec{h}$ το σύνολο εξόδου H, που παράγεται από τον τελεστή $\otimes$. Προκειμένου να υπολογισθεί ένα κελί $pt(i,j)$, είναι απαραίτητη η εφαρμογή του τελεστή $\otimes$ και για τις δύο περιπτώσεις Q$\otimes$U και Q$\otimes a$. Επομένως, από τον ορισμό 5 μπορεί εύκολα να εξαχθεί το συμπέρασμα ότι H=(Q$\otimes$U) $\bigcup$ (Q$\otimes a$) $= h_{12} \bigcup h_Y(h_{12})$, όπου $h_{12} = h_1 \bigcup h_2$. Συνεπώς, θα γίνει χρήση των δύο βοηθητικών διανυσμάτων $\vec{h}_{12}$ και $\vec{h}_Y$. Το πρώτο διάνυσμα $\vec{h}_{12}$ αντιστοιχεί στο σύνολο h_{12}, ενώ το δεύτερο $\vec{h}_Y$ αντιστοιχεί στο σύνολο h_Y. Το διάνυσμα εξόδου, μπορεί να γραφεί στην μορφή $\vec{h}(x) = \vec{h}_{12}(x) + \vec{h}_Y(x)$.
Αφού υπάρχουν $|r|$ θέσεις για την τελεία για κάθε έναν από τους $|P|$ κανόνες, τότε το απαραίτητο μήκος των διανυσμάτων είναι $\sum_{r \in P}(|r|) = |G|$ bit. Όλα τα διανύσματα $\vec{q}$, $\vec{u}$, $\vec{h}_Y$, και $\vec{h}_{12}$ έχουν αυτό το μέγεθος. Κάθε bit b_y στο διάνυσμα ορίζει την ύπαρξη ενός κανόνα με τελεία r_i^j. Προκειμένου να ακολουθηθεί η σημειογραφία που φαίνεται στο σχήμα 5.4, το y πρέπει να ισούται με $\sum_{p=1}^{p=i-1}(|r_p|) + j - 1$.

Παράδειγμα 7 *Ο κανόνας* $r_3^1 : T \rightarrow_\bullet \alpha$ *είναι ο τρίτος κανόνας με την τελεία στην πρώτη θέση,* $i = 3$ *και* $j = 1$*. Ο κανόνας αυτός αντιστοιχεί στο* b_6*, αφού* $\sum_{p=1}^{p=3-1}(|r_p|) + 1 - 1 = 2 + 4 + 1 - 1 = 6$*, όπως φαίνεται στο σχήμα 5.4.*



Σχήμα 5.4: Αναπαράσταση Δεδομένων.

Επιπροσθέτως, το σύνολο $Y = Predict(N)$ μπορεί επίσης να αναπαρασταθεί από ένα διάνυσμα μεγέθους $|G|$. Μάλιστα, έχει προϋπολογιστεί, αντιστοιχιθεί στο $\vec{p}$ και ενσωματωθεί στο $C_{\otimes}$ αφού είναι θεμελιώδες για τον υπολογισμό του $\vec{h}_Y$.
Παρομοίως, κάθε τερματικό σύμβολο αναπαρίσταται από το διάνυσμα $\vec{\tau}$ μεγέθους $|T|$, όπου T είναι το σύνολο των τερματικών συμβόλων. Σε αυτό το διάνυσμα, κάθε bit ορίζει την ύπαρξη (1) ή την απουσία (0) ενός συγκεκριμένου συμβόλου. Προφανώς, κάθε διάνυσμα $\vec{\tau}$, μπορεί να έχει μόνο ένα bit στην τιμή 1 καθώς αναπαριστά μόνο ένα τερματικό σύμβολο. Συνεπώς, η συμβολοσειρά εισόδου αναπαρίσταται με χρήση n διανυσμάτων μεγέθους $|T|$. Το σχήμα 5.4 απεικονίζει το σχήμα κωδικοποίησης, σύμφωνα με τη γραμματική χωρίς συμφραζόμενα του παραδείγματος 1.
Ακολουθώντας αυτό το σχήμα κωδικοποίησης, το προτεινόμενο κύκλωμα έχει ως είσοδο 2 διανύσματα μεγέθους $|G|$ (για τα σύνολα Q, U) και ένα μεγέθους $|T|$ (για το αντίστοιχο σύμβολο εισόδου) και εξάγει ένα διάνυσμα μεγέθου $|G|$ (για το σύνολο H).

Παράδειγμα 8 *Για τη γραμματική G_1 του παραδείγματος 1, ισχύει $|G_1| = 8$ και $|T| = 2$. Επομένως, τα σύνολα Q=$\{E\rightarrow_{\bullet} T,\ T\rightarrow_{\bullet} a\}$, U=$\{T\rightarrow a_{\bullet}\}$, Y=Predict(N) = $\{E\rightarrow_{\bullet} T,\ E\rightarrow_{\bullet} E+T,\ T\rightarrow_{\bullet} a\}$, $Q\otimes U$=$\{E\rightarrow T_{\bullet},\ E\rightarrow E_{\bullet}+T\}$ και $Q\otimes\alpha$=$\{T\rightarrow a_{\bullet},\ E\rightarrow T_{\bullet},\ E\rightarrow E_{\bullet}+T\}$ μπορούν να αντιστοιχιθούν στα ακόλουθα διανύσματα του πίνακα 5.1.*

Σύνολο	Αναπαράσταση Bit
Q	10000010
U	00000001
Y	10100010
Q⊗U	01010000
Q⊗α	01010001

Πίνακας 5.1: Παράδειγμα αναπαράστασης δεδομένων.

5.3.2 Δημιουργία των δυαδικών εξισώσεων

Οι χαρακτηριστικές δυαδικές εξισώσεις που ορίζουν την έξοδο συναρτήσει των εισόδων για το κύκλωμα $C_{\otimes}$, εξαρτώνται από τη γραμματική. Το $x^{το}$ bit του διανύσματος εξόδου $\vec{h}$ ορίζεται από τη συνάρτηση $\vec{h}(x) = \vec{h}_{12}(x) + \vec{h}_Y(x)$, όπου $\vec{h}_{12}()$ είναι μια συνάρτηση που υπολογίζει τις τιμές των bit του διανύσματος $\vec{h}_{12}$ και εξαρτάται από τα διανύσματα $\vec{q}, \vec{u}, \vec{\tau}$ και $\vec{h}_{12}$, ενώ $\vec{h}_Y()$ είναι μια συνάρτηση που υπολογίζει το διάνυσμα $\vec{h}_Y$ και εξαρτάται από τα διανύσματα $\vec{h}_{12}$, $\vec{h}_Y$ και $\vec{p}$. Το αυτόματο εργαλείο παράγει αυτόματα τις δυαδικές εξισώσεις για κάθε γραμματική χωρίς συμφραζόμενα. Ακολούθως, ορίζονται οι πιθανές μορφές των συναρτήσεων. Πρώτα θα εξεταστεί η συνάρτηση $\vec{h}_{12}()$.
Για κάθε κανόνα r_i^j, που δεν έχει φτάσει η τελεία στο τέλος του, δεξιά της τελείας μπορεί να υπάρχει είτε τερματικό είτε μη τερματικό σύμβολο. Οι δύο περιπτώσεις θα εξεταστούν χωριστά:

- Για έναν κανόνα r_i^j της μορφής $A \rightarrow \alpha_{\bullet} a \beta$, που ανήκει στο $\vec{q}$ στην θέση $s-1$, ο κανόνας r_i^{j+1} $A \rightarrow \alpha a_{\bullet} \beta$ θα προστεθεί στο $\vec{h}_{12}$ στη θέση s, εάν το τερματικό σύμβολο a ισούται με το $\vec{\tau}(z)$, όπου $\vec{\tau}(z)$ είναι το σύμβολο της συμβολοσειράς εισόδου προς αναγνώριση από το επεξεργαστικό στοιχείο P_z. Πιο τυπικά, η συνάρτηση $\vec{h}_{12}()$ μπορεί να οριστεί:

$$\vec{h}_{12}(s) = \vec{q}(s-1) \cdot \vec{\tau}(z) \tag{5.1}$$

  όπου "$\cdot$" είναι η λογική πράξη ΚΑΙ. Αυτό σημαίνει ότι εάν το bit $\vec{q}(s-1)$ είναι 1 (σηματοδοτώντας την ύπαρξη του κανόνα $A \rightarrow \alpha_{\bullet} a \beta$) και το bit $\vec{\tau}(z)$ που αντιστοιχεί στο a είναι 1, τότε ο κανόνας $A \rightarrow \alpha a_{\bullet} \beta$ θα πρέπει να προστεθεί στο $\vec{h}_{12}$ (το bit $\vec{h}_{12}(s)$ γίνεται 1).

- Έστω ο κανόνας r_i^j της μορφής $B \rightarrow \gamma_{\bullet} C \delta$, που ανήκει στο $\vec{q}$ στη θέση $s-1$. Αναζητούνται οι κανόνες που έχουν το σύμβολο C στο αριστερό μέλος του κανόνα παραγωγής. Έστω ότι οι κανόνες αυτοί είναι $(r_{c_1}) : C \rightarrow \zeta, (r_{c_2}) : C \rightarrow \eta, \ldots, (r_{c_l}) : C \rightarrow \rho$. Εάν οι κανόνες με την τελεία στο τέλος τους, $C \rightarrow \zeta_{\bullet}, C \rightarrow \eta_{\bullet}, \ldots, C \rightarrow \rho_{\bullet}$ αντιστοιχούν στα bit των θέσεων $v_1, v_2, \ldots, v_l$, τότε ο κανόνας r_i^{j+1} της μορφής $B \rightarrow \gamma C_{\bullet} \delta$ πρέπει να προστεθεί στο $\vec{h}_{12}$ στην θέση s, εφόσον τουλάχιστον ένα εκ των bit $\vec{u}(v_1), \ldots, \vec{u}(v_l)$ είναι 1. Πιο τυπικά, η συνάρτηση $\vec{h}_{12}()$ μπορεί να οριστεί:

$$\vec{h}_{12}(s) = \vec{q}(s-1) \cdot (\vec{u}(v_1) + \ldots + \vec{u}(v_l)) \tag{5.2}$$

  όπου "+" είναι η λογική πράξη Ή. Αυτό σημαίνει ότι εάν το bit $\vec{q}(s-1)$ είναι 1 (σηματοδοτώντας την ύπαρξη του κανόνα $B \rightarrow \gamma_{\bullet} C \delta$) και τουλάχιστον ένα εκ των bit $\vec{u}(v_1), \ldots, \vec{u}(v_l)$ (σηματοδοτώντας την ύπαρξη τουλάχιστον ενός εκ

των $C \to \zeta_\bullet, C \to \eta_\bullet \ldots, C \to \rho_\bullet$) είναι 1 τότε ο κανόνας $B \to \gamma C_\bullet \delta$ πρέπει να προστεθεί στο $\vec{h}_{12}$ (το bit $\vec{h}_{12}(s)$ γίνεται 1). Επιπλέον, εάν το C παράγει το κενό (ε-string), το $\vec{h}_{12}(s)$ πρέπει να γίνει ότι και το $\vec{h}_{12}(s-1)$, που σημαίνει ότι απλώς η ύπαρξη του $B \to \gamma_\bullet C \delta$ αρκεί για την προσθήκη του $B \to \gamma C_\bullet \delta$ στο $\vec{h}_{12}$. Η αναζήτηση για μη τερματικά σύμβολα που παράγουν το κενό καθώς και ο υπολογισμός των συνόλων $Predict(N)$, $Descendants$ και $Predecessors$ είναι διαδικασίες που γίνονται στο προϋπολογιστικό επίπεδο (Ανάλυση Γραμματικής) όπως φαίνεται στο σχήμα 5.1.

Στη συνέχεια, θα εξεταστεί η συνάρτηση $\vec{h}_Y()$. Ένας κανόνας r_i^j της μορφής $B \to \delta C_\bullet \xi \eta$ προστίθεται στη θέση s του $\vec{h}_Y$, εάν ικανοποιούνται οι ακόλουθες συνθήκες:

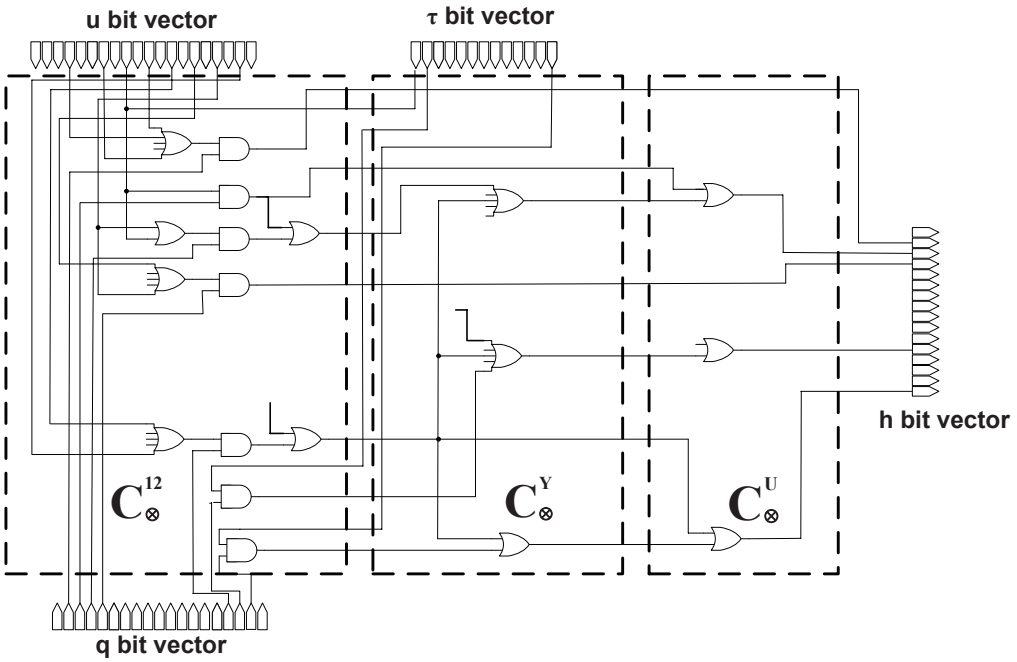
1. Ο κανόνας r_i^{j-1} της μορφής $B \to \delta_\bullet C \xi \eta$ είναι 1 στο $\vec{p}$ ($Predict(N)$), δηλαδή $\vec{p}(s-1) = 1$.
2. Για τουλάχιστον ένα από τα σύμβολα που ανήκουν στο σύνολο $Descendant(C) = \{D_1, \ldots, D_d\}$ υπάρχει ένας κανόνας που έχει την τελεία στο τέλος του και αυτό το σύμβολο στο αριστερό του μέλος, στο $\vec{h}_{12}$, δηλαδή τουλάχιστον ένα $D_i \to \kappa_\bullet$ ανήκει στο $\vec{h}_{12}$, όπου $1 \leq i \leq d$.

Έστω ότι τα σύμβολα $D_1, \ldots, D_d$ είναι στο αριστερό μέλος σε π κανόνες με την τελεία στο τέλος τους και αντιστοιχούν στα bit $w_1, \ldots, w_\pi$. Τότε το $\vec{h}_Y()$ χαρακτηρίζεται από την ακόλουθη εξίσωση:

$$\vec{h}_Y(s) = \vec{p}(s-1) \cdot (\vec{h}_{12}(w_1) + \ldots + \vec{h}_{12}(w_\pi)) \tag{5.3}$$

Αυτό σημαίνει ότι εάν το bit $\vec{p}(s-1)$ είναι 1 (ύπαρξη του $B \to \delta_\bullet C \xi \eta$ στο $Predict(N)$) και τουλάχιστον ένα από τα bit $\vec{h}_{12}(w_1), \ldots, \vec{h}_{12}(w_\pi)$ (ύπαρξη του $D_i \to \kappa_\bullet$) είναι 1 τότε ο κανόνας $B \to \gamma C_\bullet \delta$ πρέπει να προστεθεί στο $\vec{h}_Y$ (το bit $\vec{h}_Y(s)$ γίνεται 1). Επιπλέον, εάν το C παράγει το ε-string, το $\vec{h}_Y(s)$ πρέπει να γίνει 1 εάν και μόνο εάν $\vec{h}_Y(s-1) = 1$, που σημαίνει ότι αρκεί η ύπαρξη του $B \to \delta_\bullet C \xi \eta$ για την προσθήκη του $B \to \delta C_\bullet \xi \eta$ στο $\vec{h}_Y$.
Μια αφηρημένη υλοποίηση της παραπάνω αρχιτεκτονικής απεικονίζεται στο σχήμα 5.5, όπου εμφανίζονται τα κρίσιμα υποκυκλώματα καθώς και η διασύνδεσή τους. Ένα πιο λεπτομερές κύκλωμα για τη γραμματική G_{Chrom}, που θα παρουσιαστεί στην ενότητα 5.4.1, δίνεται στο σχήμα 5.15.

Παράδειγμα 9 *Για άλλη μια φορά, θεωρούμε τη γραμματική G_1 και τα σύνολα Q, U και Y του παραδείγματος 1. Τα σύνολα αυτά έχουν αναπαρασταθεί σε διανύσματα σύμφωνα με το παράδειγμα 7. Θα γίνει επίδειξη της χρήσης των εξισώσεων (5.1) - (5.3) για το τερματικό σύμβολο a ($\vec{\tau}(1) = 1$), όπως έγινε στο παράδειγμα 4 και θα συγκριθούν τα αποτελέσματα.*



Σχήμα 5.5: Προσέγγιση υψηλού επιπέδου - $C_{\otimes}^{12}$ είναι το κύκλωμα που υπολογίζει το $\vec{h}_{12}$, το $\vec{h}_Y$ υπολογίζεται από το κύκλωμα $C_{\otimes}^{Y}$ και το διάνυσμα εξόδου $\vec{h}$ παράγεται ως λογικό Ή μεταξύ των $\vec{h}_{12}$ και $\vec{h}_Y$.

- *Όπως φάνηκε στο παράδειγμα 4, $Q{\otimes}U{=}\{E{\rightarrow}T_{\bullet}, E{\rightarrow}E_{\bullet}{+}T\}$ επειδή, σύμφωνα με τον τελεστή $\otimes$, πρώτα ελέγχεται το σύνολο U για κανόνες με την τελεία στην τελευταία τους θέση ($T{\rightarrow}a_{\bullet} \in U$ σε αυτό το παράδειγμα $\vec{u}(7) = 1$). Ακολούθως, το σύνολο Q ελέγχεται για κανόνες που έχουν δεξιά της τελείας το σύμβολο που βρίσκεται στο αριστερό μέλος των κανόνων με την τελεία στο τέλος τους (T εδώ). Για τους κανόνες που πληρούν τα κριτήρια ($E{\rightarrow}_{\bullet}T \in Q$ σε αυτό το παράδειγμα $\vec{q}(0) = 1$), ο κανόνας προστίθεται στο σύνολο $Q{\otimes}U$ αφού μετακινηθεί η τελεία μια θέση προς τα δεξιά ($E{\rightarrow}T_{\bullet} \in Q{\otimes}U$ εδώ). Σύμφωνα με την εξίσωση (5.2) αυτός ο κανόνας θα προστεθεί στο διάνυσμα $\vec{h}_{12}$ αφού $\vec{h}_{12}(1) = \vec{q}(0) \cdot \vec{u}(7) = 1 \cdot 1 = 1$.*

  Επιπλέον, στην περίπτωση που ο προστιθέμενος κανόνας ($\vec{h}_{12}(1)$: $E{\rightarrow}T_{\bullet}$ στο παράδειγμα) έχει την τελεία στο τέλος του, το σύνολο Z_1= Predecessors(lhss) προσδιορίζεται (μόνο $E \in Z_1$ στο παράδειγμα). Τέλος, το σύνολο Y ελέγχεται για κανόνες, στους οποίους δεξιά της τελείας βρίσκεται ένα από τα μη τερματικά σύμβολα που ανήκουν στο Z_1 ($E{\rightarrow}_{\bullet}E{+}T \in Y$ σε αυτό το παράδειγμα $\vec{p}(2) = 1$). Αυτοί οι κανόνες προστίθενται στο σύνολο $Q{\otimes}U$ αφού μετακινηθεί η τελεία μια θέση προς τα δεξιά ($E{\rightarrow}E_{\bullet}{+}T \in Q{\otimes}U$ στο παράδειγμα). Σύμφωνα με την εξίσωση (5.3) αυτός ο κανόνας θα προστεθεί στο διάνυσμα $\vec{h}_Y(3) = \vec{p}(2) \cdot \vec{h}_{12}(1)$.

- *Όπως φάνηκε στο παράδειγμα 4, $Q{\otimes}\alpha$=$\{T{\rightarrow}a_{\bullet}$, $E{\rightarrow}T_{\bullet}$, $E{\rightarrow}E_{\bullet}{+}T\}$ επειδή, σύμφωνα με τον τελεστή $\otimes$, το σύνολο Q ελέγχεται για κανόνες στους οποίους, δεξιά της τελείας υπάρχει το τερματικό σύμβολο a ($T{\rightarrow}_{\bullet}a \in Q$ στο παράδειγμα $\vec{q}(6) = 1$ και το a αντιστοιχεί στο $\vec{\tau}(1)$). Οι προσδιορισμένοι κανόνες προστίθενται στο σύνολο $Q{\otimes}a$, αφού μετακινηθεί η τελεία μια θέση προς τα δεξιά ($T{\rightarrow}a_{\bullet} \in Q{\otimes}a$ στο παράδειγμα). Σύμφωνα με την εξίσωση (5.1) αυτός ο κανόνας θα προστεθεί στο διάνυσμα $\vec{h}_{12}$ αφού $\vec{h}_{12}(7) = \vec{q}(6) \cdot \vec{\tau}(1) = 1 \cdot 1 = 1$.
  Επιπλέον, στην περίπτωση που ο προστιθέμενος κανόνας ($\vec{h}_{12}(7)$: $T{\rightarrow}a_{\bullet}$ στο παράδειγμα) έχει την τελεία στο τέλος του, το σύνολο $Z_2 = Predecessors(lhss)$ προσδιορίζεται (E, T στο παράδειγμα). Τέλος, το σύνολο Y ελέγχεται για κανόνες, στους οποίους δεξιά της τελείας βρίσκεται ένα από τα μη τερματικά σύμβολα που ανήκουν στο Z_2 ($E{\rightarrow}_{\bullet}E{+}T$ και $E{\rightarrow}_{\bullet}T \in Y$ στο παράδειγμα $\vec{p}(0) = \vec{p}(2) = 1$). Αυτοί οι κανόνες προστίθενται στο σύνολο $Q{\otimes}a$ αφού μετακινηθεί η τελεία μια θέση προς τα δεξιά ($E{\rightarrow}T_{\bullet}$ και $E{\rightarrow}E_{\bullet}{+}T \in Q{\otimes}a$ σε αυτό το παράδειγμα). Σύμφωνα με την εξίσωση (5.3) αυτοί οι κανόνες θα προστεθούν στο διάνυσμα $\vec{h}_Y$, αφού $\vec{h}_Y(3) = \vec{p}(2) \cdot \vec{h}_{12}(7) = 1 \cdot 1 = 1$ και $\vec{h}_Y(1) = \vec{p}(0) \cdot \vec{h}_{12}(7) = 1 \cdot 1 = 1$.

Από το παράδειγμα 4, H =$(Q{\otimes}U) \bigcup (Q{\otimes}a) = \{E{\rightarrow}T_{\bullet}$, $E{\rightarrow}E_{\bullet}{+}T$, $T{\rightarrow}\ a_{\bullet}\}$. Αφού $\vec{h}_{12} = 01000001$ και $\vec{h}_Y = 01010000$, είναι προφανές ότι $\vec{h} = \vec{h}_{12} + \vec{h}_Y = 01010001$ είναι η αναπαράσταση σε διάνυσμα bit του H. Αξίζει να τονιστεί ότι καταλήξαμε στο ίδιο συμπέρασμα με την εφαρμογή λογικών τελεστών, γεγονός που αποδεικνύει την αποτελεσματικότητα της υλοποίησης.

5.3.3 Πολυπλοκότητα

Η πράξη $\otimes$ είναι μια πολύπλοκη ενέργεια, η οποία εάν εκτελεστεί ακολουθιακά απαιτεί στην χειρότερη περίπτωση χρόνο $O(|G|)$. Αυτό συνεπάγεται από τον ορισμό του $\otimes$, όπου φαίνεται ότι μια εκτέλεση του $\otimes$ μπορεί να χρειαστεί να διατρέξει όλο το σύνολο $P^{\bullet}$ των κανόνων, το μέγεθος των οποίων είναι $O(|G|)$. Στην προτεινόμενη υλοποίηση, μια καινοτόμος μεθοδολογία επιτρέπει την παραλληλοποίηση των υπολογισμών μέσα σε κάθε κελί, εκτελώντας την πράξη $\otimes$ παράλληλα. Αυτό γίνεται απλά και κομψά με το σχεδιασμό του ειδικού συνδυαστικού κυκλώματος.

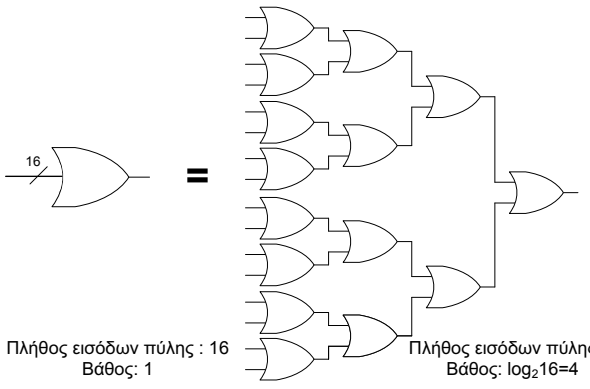
Το συνδυαστικό κύκλωμα $C_{\otimes}$, που φαίνεται σε μια προσέγγιση υψηλού επιπέδου στο σχήμα 5.5 ενώ σε πιο λεπτομερή στο σχήμα 5.15, παράγεται από το σύστημα των εξισώσεων (5.1) - (5.3). Συνεπώς, το $C_{\otimes}$ μπορεί να χωριστεί σε τρία κομμάτια: $C_{\otimes}^{12}$, $C_{\otimes}^{Y}$ και $C_{\otimes}^{\cup}$, όπως καθορίζονται από τις διακεκομμένες γραμμές. Το κύκλωμα $C_{\otimes}^{12}$, που υπολογίζει το βοηθητικό διάνυσμα $\vec{h}_{12}$, κατασκευάζεται από τις εξισώσεις (5.1) και (5.2). Η υλοποίηση σε υλικό της εξίσωσης (5.2) $\vec{h}_{12}(s) = \vec{q}(s-1) \cdot (\vec{u}(v_1) + \ldots + \vec{u}(v_l))$ απαιτεί ένα κύκλωμα φραγμένο από $\lceil log_2|P| \rceil$. Αφού το δεύτερο μέρος

της εξίσωσης: $\vec{u}(v_1) + \ldots + \vec{u}(v_l)$ έχει μέγιστο μήκος $|P|$, ίσο με τον αριθμό των κανόνων με την τελεία στο τέλος τους, μπορεί να υπολογιστεί από ένα κύκλωμα με βάθος $\lceil log_2|P| \rceil$. Μπορεί γενικότερα να αποδειχθεί ότι κάθε λογική πύλη m εισόδων μπορεί να αναλυθεί σε ένα κύκλωμα βάθους $log_2 m$ αποτελούμενο από λογικές πύλες 2 εισόδων, όπως φαίνεται στο σχήμα 5.6. Όπως έχει αναλυθεί στην παρουσίαση της εξίσωσης (5.2), σε ορισμένες περιπτώσεις πρέπει να προστεθεί ένας επιπλέον όρος, ο $\vec{h}_{12}(s-1)$. Αυτό ισχύει μόνο για τα bit που αντιστοιχούν σε όμοιους κανόνες αλλά με την τελεία σε διαφορετική θέση και επομένως μπορούν να υπολογιστούν παράλληλα για κάθε κανόνα της γραμματικής. Το έξτρα υλικό που χρειάζεται για κάτι τέτοιο, φράσσεται άνω από $\lceil log_2|r_m| \rceil$, όπου r_m είναι ο μέγιστος κανόνας της γραμματικής. Το συνολικό βάθος της εξίσωσης (5.2) είναι $\lceil log_2|P| \rceil + \lceil log_2|r_m| \rceil = O(log_2(|P||r_m|)) = O(log_2|G|)$. Το βάθος της εξίσωσης (5.1) είναι 1 και συνεπώς, το βάθος του $C_{\otimes}^{12}$ είναι $\lceil log_2|G| \rceil$. Αντίστοιχα, το κύκλωμα $C_{\otimes}^{Y}$, που υπολογίζει το βοηθητικό διάνυσμα bit $\vec{h}_Y$, κατασκευάζεται βάσει της εξίσωσης 5.3 και το βάθος του επίσης περιορίζεται άνω από $\lceil log_2|G| \rceil$. Το τρίτο κομμάτι, το $C_{\otimes}^{\cup}$ παράγει το διάνυσμα εξόδου $\vec{h}$ με λογικό Ή των αντίστοιχων bit των διανυσμάτων $\vec{h}_{12}$ και $\vec{h}_Y$ και έχει βάθος 1. Τα τρία αυτά υποκυκλώματα είναι συνδεδεμένα στη σειρά, επομένως το συνολικό κύκλωμα $C_{\otimes}$ χρειάζεται

$$O\left(log_2(|G|) + log_2(|G|) + 1\right) = O\left(log_2(|G|)\right) \tag{5.4}$$

χρόνο για να ολοκληρώσει μια πράξη.

Το συνδυαστικό κύκλωμα $C_{\otimes}$ είναι ικανό να φέρει σε πέρας μια πράξη $\otimes$ σε χρόνο $O(log_2|G|)$ όπως αποδείχθηκε παραπάνω. Το σχήμα 5.2c δείχνει πώς $j-i-1$ κυκλώματα $C_{\otimes}$ μπορούν να συνδεθούν με ένα κύκλωμα $C_{\mathbf{U}}$, που είναι υπεύθυνο για τις $j-i-2$ ενώσεις, προκειμένου να υπολογιστεί το κελί $pt(i,j)$. Όπως έχει αναλυθεί, όλα τα κελιά $pt(i,j)$ για τα οποία $j-i=k \geq 1$, υπολογίζονται παράλληλα κατά το βήμα εκτέλεσης k. Εάν $k=1$ ή 2, μια πράξη $C_{\otimes}$ χρειάζεται ανά κελί και καμία



Σχήμα 5.6: Ανάλυση του βάθους ενός λογικού κυκλώματος όσον αφορά στον αριθμό των εισόδων των λογικών πυλών.

ένωση, επομένως χρονικά χρειάζεται $O(log_2|G|)$.
Για $k > 2$ όλα τα $((j-i-1) = (k-1))$ $C_{\otimes}$ κυκλώματα λειτουργούν ανεξάρτητα και παράλληλα, επομένως χρειάζονται μόνον $O(log_2|G|)$ χρόνο. Το βάθος του κυκλώματος $C_{\mathbf{U}}$ καθορίζεται από το πλήθος των εισόδων. Θεωρώντας ότι πύλες m εισόδων είναι υλοποιήσιμες, όπου το m εξαρτάται από την τρέχουσα κάθε φορά τεχνολογία, το βάθος του $C_{\mathbf{U}}$ είναι $\lceil log_m k \rceil$ (η απόδειξη φαίνεται στο σχήμα 5.6). Για απλότητα, το m θεωρείται ίσο με 2 και συνεπώς το $pt(i,j)$ υπολογίζεται σε χρόνο $O(log_2|G| + log_2 k) = O(log_2(k|G|))$. Τελικά, ο υπολογισμός όλων των κελιών απαιτεί χρόνο $O(log_2(k|G|))$ και ο υπολογισμός όλου του πίνακα PT μπορεί να γίνει σε χρόνο

$$O\left(log_2|G| + log_2|G| + \sum_{k=3}^{n} log_2(k|G|)\right) =$$

$$= O\left(log_2|G| + log_2|G| + log_2(3|G|) + \ldots + log_2(n|G|)\right)$$

$$= O\left(log_2|G| + log_2(2|G|) + log_2(3|G|) + \ldots + log_2(n|G|)\right) =$$

$$= O\left(log_2(n!|G|^n)\right) = O\left(nlog_2 n + nlog_2|G|\right) = O(nlog_2(n|G|)) \qquad (5.5)$$

Για να προκύψει το ανωτέρω αποτέλεσμα χρησιμοποιήθηκε το γεγονός ότι $O\left(log_2(n!)\right) = O\left(nlog_2 n\right)$.

5.4 Πειραματικά Αποτελέσματα

Προκειμένου να αναδειχθεί η επίδοση της προτεινόμενης αρχιτεκτονικής, τρεις συντακτικοί αναλυτές υλοποιήθηκαν, ένας για κάθε γραμματική. Αρχικά, μια υλοποίηση καθαρά λογισμικού εκτελέστηκε σε ένα Pentium Celeron @ 2.6 GHz. Ακολούθως, η ίδια υλοποίηση εκτελέστηκε στον soft core επεξεργαστή MicroBlaze @ 75 MHz, υλοποιημένο σε ένα FPGA Spartan2 ή σε ένα FPGA Virtex5, ανάλογα με το μέγεθος της γραμματικής. Τέλος, η προτεινόμενη αρχιτεκτονική υλοποιήθηκε και εκτελέστηκε στα ίδια FPGA, στη συχνότητα των 100 MHz[1] τώρα. Για κάθε υλοποίηση, δύο ειδών μετρήσεις ελήφθησαν: μια για τους κύκλους ρολογιού και μια για το χρόνο εκτέλεσης. Και για τα δύο είδη μετρήσεων, οι υλοποιήσεις δοκιμάστηκαν για διάφορα μεγέθη συμβολοσειράς εισόδου.
Το πρώτο είδος των μετρήσεων (κύκλοι ρολογιού) ελήφθη για να καταστεί δυνατή η καθαρή σύγκριση μεταξύ των δοκιμαζομένων αρχιτεκτονικών, ανεξαρτήτως της χρησιμοποιούμενης τεχνολογίας. Αν και ο Pentium και ο MicroBlaze είναι επεξεργαστές RISC, υπάρχει τεράστια διαφορά στους αναγκαίους κύκλους μηχανής. Προκειμένου

[1] Αν και η μέγιστη ταχύτητα ρολογιού για το Spartan2 είναι 100 MHz, το ρολόι του soft core MicroBlaze μπορεί να φτάσει μόνο μέχρι τα 75 MHz

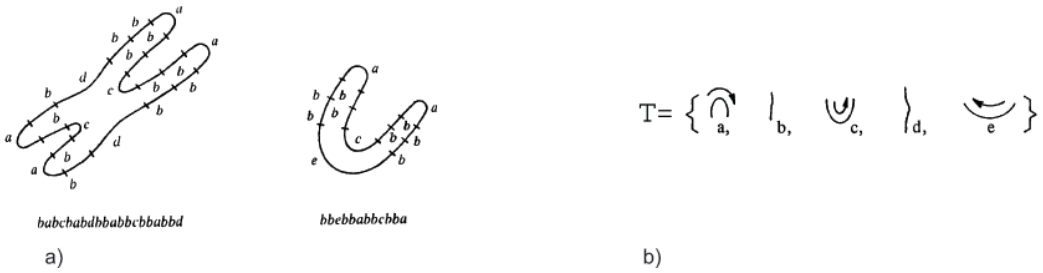
να διασαφηνίσει αυτό, πρέπει να εξεταστεί η αρχιτεκτονική του MicroBlaze. Ο MicroBlaze έχει μια μικρή μνήμη cache 64K, η οποία δεν είναι αρκετά μεγάλη για το εκτελέσιμο και τα δεδομένα, επομένως μια SDRAM χρησιμοποιείται για την αποθήκευση των δεδομένων. Σχεδόν σε κάθε βήμα εκτέλεσης, δεδομένα στέλνονται από την SDRAM στον επεξεργαστή - μια αργή διαδικασία που οδηγεί στην αναμονή του επεξεργαστή για τα δεδομένα. Ως εκ τούτου, προκύπτει ότι οι αυξημένοι κύκλοι ρολογιού οφείλονται στον αργό διάδρομο δεδομένων, που διασυνδέει τον επεξεργαστή με την SDRAM. Επιπλέον, οι κύκλοι ρολογιού που απαιτούνται από την προτεινόμενης αρχιτεκτονική είναι σημαντικά μικρότεροι από εκείνους που απαιτούνται από την εφαρμογή που εκτελείται στον Pentium ή στο MicroBlaze .

Το δεύτερο είδος των μετρήσεων (χρόνος εκτέλεσης) ελήφθη προκειμένου να δοθεί μια πιο χειροπιαστή ένδειξη της επιτάχυνσης. Λόγω της χαμηλής συχνότητας ρολογιού του FPGA (100 MHz), ο απαιτούμενος χρόνος θα μπορούσε να είναι συγκρίσιμος με εκείνο που απαιτείται από εφαρμογή λογισμικού, όταν εκτελείται σε έναν πιο γρήγορο επεξεργαστή (2,6 GHz). Ακόμα κι αν η συχνότητα ρολογιού είναι μία τάξη μεγέθους μεγαλύτερη από εκείνη του FPGA, ο χρόνος που απαιτείται από την αρχιτεκτονική μας είναι σημαντικά μικρότερος, δεδομένου ότι απαιτούνται δραστικά λιγότεροι κύκλοι ρολογιού. Στην περίπτωση του MicroBlaze επεξεργαστή, παρά το γεγονός ότι η συχνότητα είναι συγκρίσιμη με εκείνη FPGA, οι κύκλοι ρολογιού είναι πολύ υψηλότεροι από εκείνους της εφαρμογής σε υλικό. Ως εκ τούτου, αναμένεται το ίδιο μεγάλη επιτάχυνση στον τομέα του χρόνου.

Τα ακριβή αποτελέσματα των χρόνων αναγνώρισης για τις υπό εξέταση γραμματικές και για διαφορετικά μήκη συμβολοσειρών εισόδου, δίνονται στα ακόλουθα.

5.4.1 Αναγνώριση Χρωματοσωμάτων

Όπως έχει αναφερθεί και νωρίτερα, στις εφαρμογές συντακτικής αναγνώρισης προτύπων η διαδικασία της αναγνώρισης μετατρέπεται στην ισοδύναμη της αναγνώρισης γλωσσολογικών αναπαραστάσεων των προς αναγνώριση προτύπων. Οι γλωσσικές αυτές αναπαραστάσεις, περιγράφονται από μια συγκεκριμένη γραμματική, τη γραμματική προτύπων (pattern grammar) [2], [67], [68] και [69]. Η γραμματική αυτή περιγράφει τα προς αναγνώριση πρότυπα με ένα συμβατικό τρόπο και ο σχηματισμός και συμπλήρωση του συντακτικού πίνακα PT είναι το χρονοβόρο υποπρόβλημα στις εφαρμογές συντακτικής αναγνώρισης προτύπων. Στη εργασία [2] παρουσιάζεται μια γραμματική χωρίς συμφραζόμενα που περιγράφει τη μορφολογία των χρωματοσωμάτων. Οι δύο τύποι χρωματοσωμάτων - submedian και telocentric (βλέπε σχήμα 5.7a) - μπορούν να χωριστούν σε πέντε θεμελιώδη τμήματα (βλέπε 5.7b). Με τη χρήση της γραμματικής G_{chrom} που παρουσιάζεται ακολούθως, και οι δύο τύποι χρωματοσωμάτων μπορούν να αναγνωριστούν. Η G_{chrom} επιλέγει ως μια απλή εφαρμογή



Σχήμα 5.7: a) Submedian και telocentric χρωματοσώματα, b) Αναπαράσταση τερματικών συμβόλων.

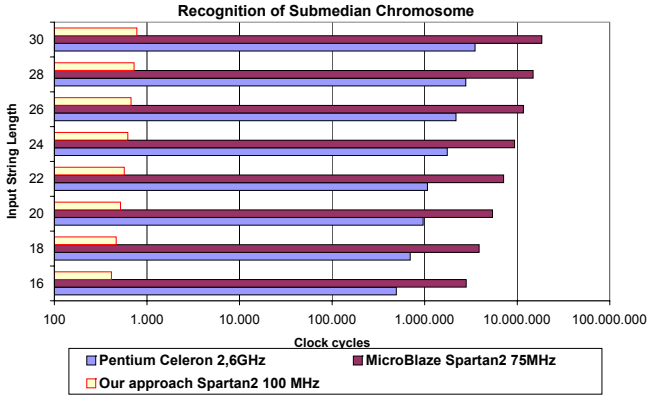
(toy-scale) για να καταφανεί η απόδοση της υλοποίησης. Οι συντακτικοί κανόνες της συγκεκριμένης γραμματικής είναι 18 με μέγιστο μήκος 3, τα μη τερματικά σύμβολα είναι 8, τα τερματικά 5 και $|G_{chrom}|$=32. $G_{chrom} = \{N, T, P, S\}$ όπου:
$N : \{submedian\ chromosome, telocentric\ chromosome, arm\ pair, left\ part, right\ part, arm, side, bottom\}$,
$T : \{a, b, c, d, e\}$,
$P : \{submedian\ chromosome \rightarrow arm\_pair\ arm\_pair, telocentric\ chromosome \rightarrow bottom\ arm\_pair, arm\_pair \rightarrow side\ arm\_pair, arm\_pair \rightarrow arm\_pair\ side, arm\_pair \rightarrow arm\ right\_part, arm\_pair \rightarrow left\_part\ arm, left\_part \rightarrow arm\ c, right\ part \rightarrow c\ arm, bottom \rightarrow bottom\ b, bottom \rightarrow b\ bottom, bottom \rightarrow e, side \rightarrow side\ b, side \rightarrow b\ side, side \rightarrow b, side \rightarrow d, arm \rightarrow arm\ b, arm \rightarrow b\ arm, arm \rightarrow a\}$,
$S : \{submedian\ chromosome, telocentric\ chromosome\}$
Οι κύκλοι μηχανής (βλέπε πίνακα 5.2) που απαιτούνται από την προτεινόμενη αρχιτεκτονική είναι δύο ή τρεις τάξεις μεγέθους (ανάλογα με το μήκος της συμβολοσειράς εισόδου) μικρότεροι από εκείνους που χρειάζεται η υλοποίηση στον Pentium. Σε σχέση με την υλοποίηση στον MicroBlaze, οι αναγκαίοι κύκλοι της προτεινόμενης αρχιτεκτονικής είναι τρεις με τέσσερις τάξεις μεγέθους μικρότεροι. Στη συγκεκρι-

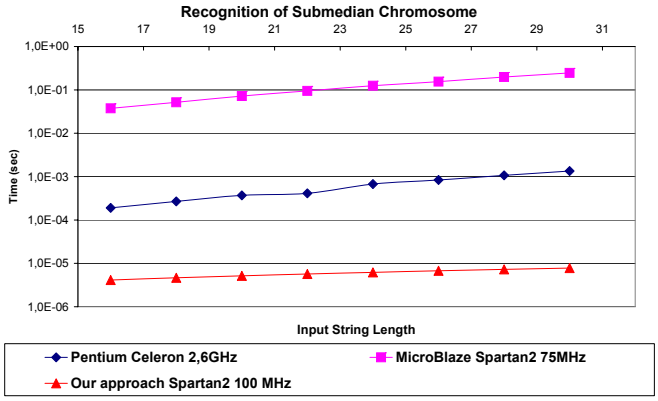
Μήκος Συμβολοσειράς εισόδου	Pentium		MicroBlaze		Υλοποίηση Υλικού
	Submedian	Telocentric	Submedian	Telocentric	Submedian και Telocentric
8	-	107.990	-	774.061	206
10	-	218.945	-	1.523.797	258
12	-	294.957	-	2.046.715	310
14	-	504.281	-	3.384.725	362
16	494.054	623.401	3.328.042	3.557.164	414
18	697.663	913.254	4.607.270	5.280.141	466
20	963.853	1.084.378	6.451.483	6.302.780	518
22	1.071.780	1.284.629	8.506.593	8.746.667	570
24	1.756.993	-	11.227.798	-	622
26	2.181.170	-	14.014.825	-	674
28	2.780.609	-	17.867.276	-	726
30	3.506.258	-	22.332.350	-	778

Πίνακας 5.2: Κύκλοι Ρολογιού για διάφορα μήκη συμβολοσειρών εισόδου της γραμματικής G_{chrom}.

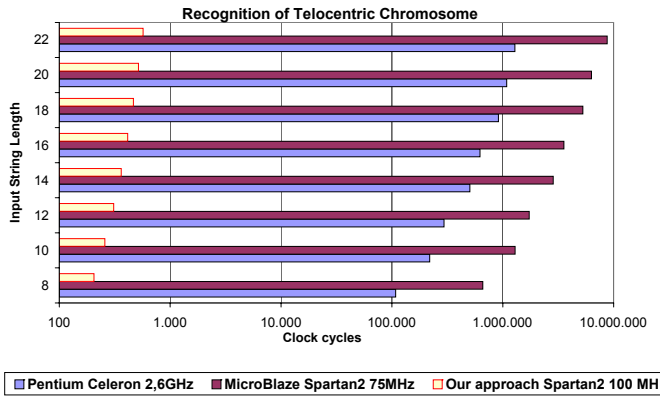
Μήκος Συμβολοσειράς Εισόδου	Pentium		MicroBlaze		Υλοποίηση Υλικού
	Submedian	Telocentric	Submedian	Telocentric	Submedian και Telocentric
8	-	4.153E-05	-	8.812E-03	2.06E-06
10	-	8.421E-05	-	1.723E-02	2.58E-06
12	-	1.134E-04	-	2.311E-02	3.10E-06
14	-	1.940E-04	-	3.802E-02	3.62E-06
16	1.900E-04	2.398E-04	3.748E-02	4.743E-02	4.14E-06
18	2.683E-04	3.513E-04	5.176E-02	7.040E-02	4.66E-06
20	3.707E-04	4.171E-04	7.219E-02	8.404E-02	5.18E-06
22	4.122E-04	4.941E-04	9.495E-02	1.166E-01	5.70E-06
24	6.758E-04	-	1.247E-01	-	6.22E-06
26	8.389E-04	-	1.554E-01	-	6.74E-06
28	1.069E-03	-	1.977E-01	-	7.26E-06
30	1.349E-03	-	2.462E-01	-	7.78E-06

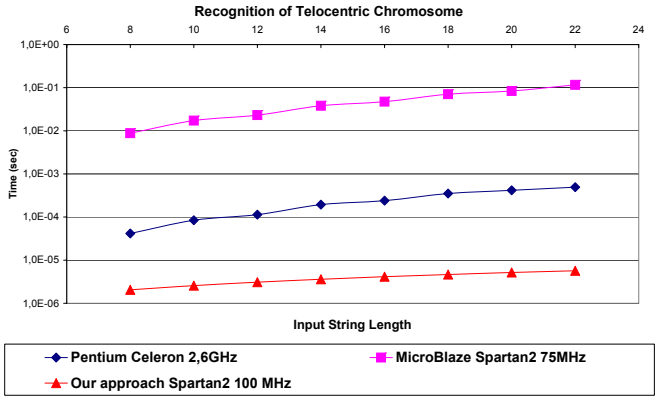
Πίνακας 5.3: Χρόνος σε δευτερόλεπτα για διαφορετικά μήκη συμβολοσειρών εισόδου της G_{chrom}.





Σχήμα 5.8: Κύκλοι Μηχανής και χρόνοι αναλυτών για τη G_{chrom} και την αναγνώριση Submedian χρωματοσωμάτων για διαφορετικό μήκος συμβολοσειρών εισόδου.

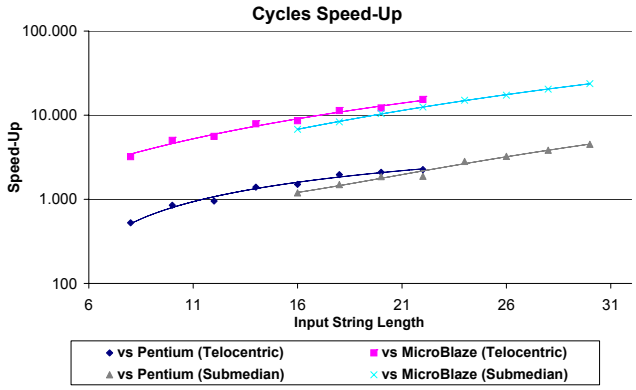


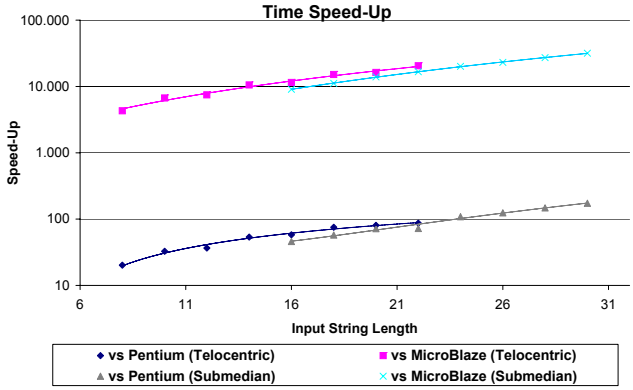


Σχήμα 5.9: Κύκλοι Μηχανής και χρόνοι αναλυτών για τη G_{chrom} και την αναγνώριση Telocentric χρωματοσωμάτων για διαφορετικό μήκος συμβολοσειρών εισόδου.

μένη γραμματική G_{chrom}, το κύκλωμα $C_{\otimes}$ κάνει τους υπολογισμούς σε ένα κύκλο ρολογιού.

Επιπλέον, όπως φαίνεται στον πίνακα 5.3, ο απαιτούμενος χρόνος από την προτεινόμενη αρχιτεκτονική είναι μια με δύο τάξεις μεγέθους μικρότερος (ανάλογα με το μήκος της συμβολοσειράς εισόδου) από εκείνον που χρειάζεται η υλοποίηση στον Pentium. Σε σχέση με την υλοποίηση στον MicroBlaze, ο απαιτούμενος χρόνος από την προτεινόμενη αρχιτεκτονική είναι τρεις με τέσσερις τάξεις μεγέθους μικρότερος (ανάλογα με το μήκος της συμβολοσειράς εισόδου). Όλα τα παραπάνω, απεικονίζονται στα σχήματα 5.8, 5.9 και 5.10.





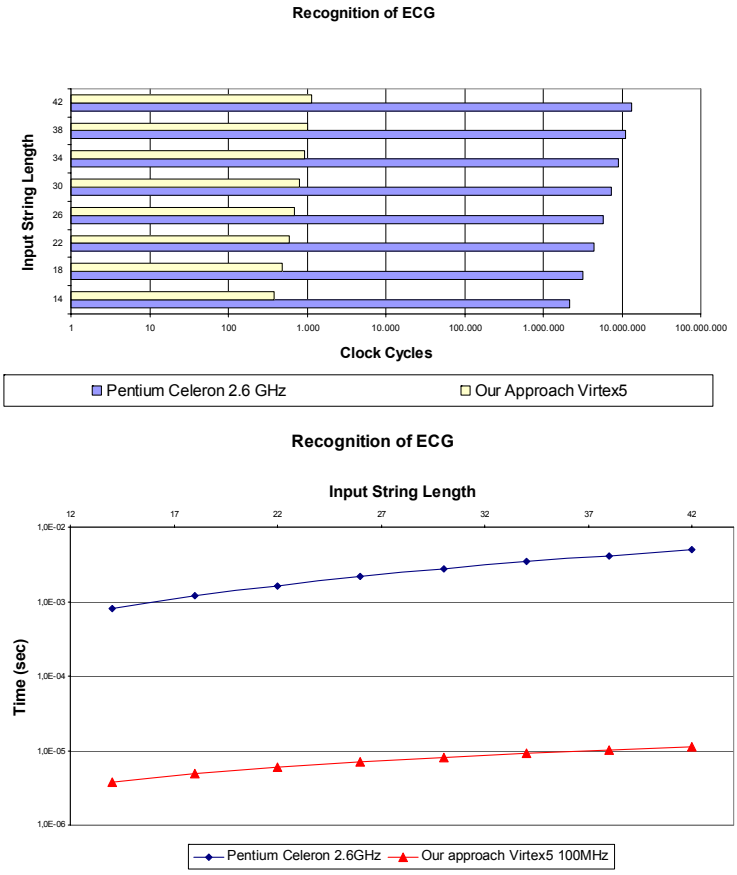
Σχήμα 5.10: Επιτάχυνση αναφορικά με τους κύκλους μηχανής και τους χρόνους των αναλυτών για τη G_{chrom} και για τα δύο είδη χρωματοσωμάτων.

5.4.2 Αναγνώριση Ηλεκτροκαρδιογραφήματος

Μια σύνθετη γραμματική, που αντιστοιχεί σε πραγματικές εφαρμογές είναι η γραμματική G_{ECG} που παρουσιάστηκε στην ενότητα 3.3. Συγκεκριμένα, χρησιμοποιήθηκε

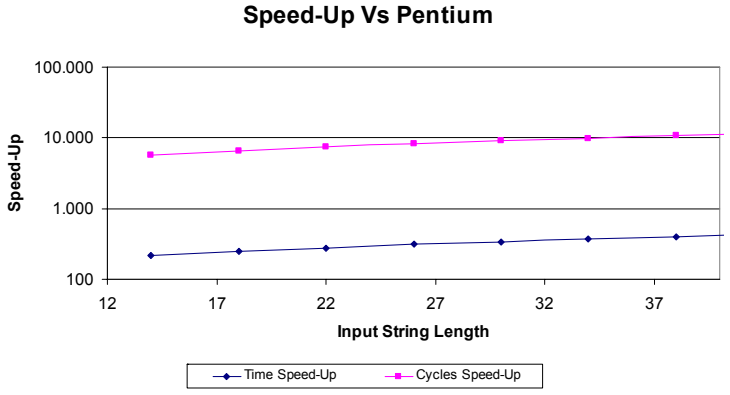
Μήκος Συμβολοσειράς Εισόδου	Pentium		Υλοποίηση Υλικού	
	Κύκλοι Ρολογιού	Χρόνος (sec)	Κύκλοι Ρολογιού	Χρόνος (sec) (sec)
14	2.148.774	8,08E-04	376	3,76E-06
18	3.162.831	1,19E-03	484	4,84E-06
22	4.367.553	1,64E-03	592	5,92E-06
26	5.763.966	2,17E-03	700	7,00E-06
30	7.329.414	2,76E-03	808	8,08E-06
34	9.082.960	3,41E-03	916	9,16E-06
38	11.018.864	4,14E-03	1.024	1,02E-05
42	13.151.534	4,94E-03	1.132	1,13E-05

Πίνακας 5.4: Κύκλοι Μηχανής και χρόνοι αναλυτών για τη G_{ECG} και για διαφορετικό μήκος συμβολοσειρών εισόδου.



Σχήμα 5.11: Κύκλοι ρολογιού και χρόνοι συντακτικής ανάλυσης για τη G_{ECG} και για διαφορετικό μήκος συμβολοσειρών εισόδου.

μόνο το συντακτικό κομμάτι της κατηγορικής γραμματικής. Οι συντακτικοί κανόνες της G_{ECG} είναι 64 με μέγιστο μήκος 8, τα μη τερματικά σύμβολα είναι 35, τα τερματικά 4 και $|G_{chrom}|$=170.



Σχήμα 5.12: Επιτάχυνση αναφορικά με τους κύκλους μηχανής και τους χρόνους των αναλυτών για τη G_{ECG} και για διαφορετικό μήκος συμβολοσειρών εισόδου.

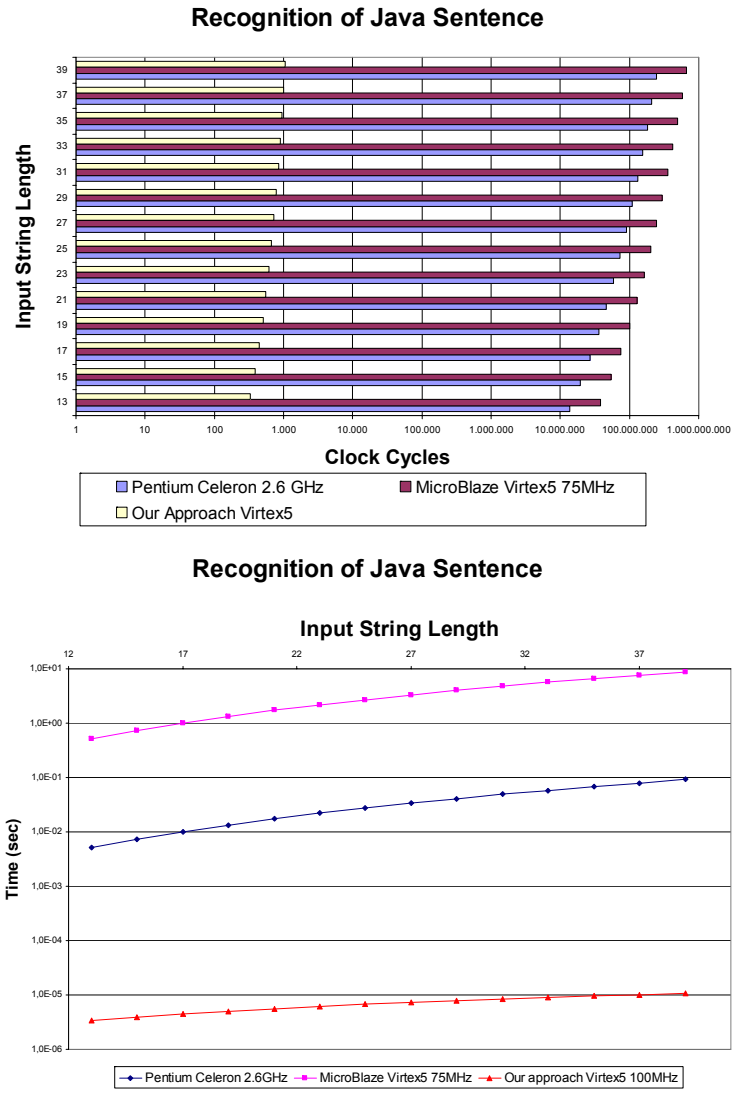
Οι κύκλοι μηχανής (βλέπε πίνακα 5.4) που απαιτούνται από την προτεινόμενη αρχιτεκτονική είναι τέσσερις τάξεις μεγέθους μικρότεροι από εκείνους που χρειάζεται η υλοποίηση στον Pentium. Σε αυτή τη γραμματική G_{ECG}, το κύκλωμα $C_{\otimes}$ βγάζει τελικό αποτέλεσμα σε δύο κύκλους ρολογιού, γι' αυτό εμφανίζονται επιπλέον κύκλοι ρολογιού στον πίνακα 5.4.
Τέλος, όπως φαίνεται και στον πίνακα 5.4, ο αναγκαίος χρόνος από την προτεινόμενη αρχιτεκτονική είναι δύο με τρεις τάξεις μεγέθους μικρότερος (ανάλογα με το μήκος της συμβολοσειράς εισόδου) από τον αντίστοιχο που χρειάζεται η υλοποίηση στον Pentium. Όλα τα παραπάνω, απεικονίζονται στα σχήματα 5.11 και 5.12.

5.4.3 Αναγνώρισης της γλώσσας Java

Άλλη μια πραγματική γραμματική, ακόμη μεγαλύτερη, παρουσιάζεται σε αυτή την υποενότητα. Η λεκτική και συντακτική δομή ενός προγράμματος σε γλώσσα Java [70] μπορεί να περιγραφεί από την γραμματική G_{Java} που παρουσιάζεται στο [71]. Οι συντακτικοί κανόνες της G_{Java} είναι 210 με μέγιστο μήκος 13, τα μη τερματικά σύμβολα είναι 50, τα τερματικά 104 και $|G_{Java}|$=1112.
Οι κύκλοι ρολογιού (βλέπε πίνακα 5.5) που απαιτούνται από την προτεινόμενη αρχιτεκτονική είναι πέντε με έξη τάξεις μεγέθους μικρότεροι από εκείνους που χρειάζεται η υλοποίηση στον Pentium. Σε σχέση με την υλοποίηση στον MicroBlaze, οι κύκλοι είναι πέντε με έξη τάξεις μεγέθους μικρότεροι.
Επιπλέον, όπως φαίνεται στον πίνακα 5.5, ο αναγκαίος χρόνος από την προτεινόμενη αρχιτεκτονική είναι τρεις με τέσσερις τάξεις μεγέθους μικρότερος (ανάλογα με το μήκος της συμβολοσειράς εισόδου) από τον αντίστοιχο που χρειάζεται η υλοποίηση στον Pentium. Συγκριτικά με την υλοποίηση στον MicroBlaze, οι αντίστοιχοι χρόνοι είναι πέντε με έξη τάξεις μεγέθους μικρότεροι στην προτεινόμενη αρχιτεκτονική.

Μήκος Συμβολοσειράς Εισόδου	Pentium		Hardware Implementation		MicroBlaze	
	Κύκλοι Ρολογιού	Χρόνος (sec)	Κύκλοι Ρολογιού	Χρόνος (sec)	Κύκλοι Ρολογιού	Χρόνος (sec)
13	13.525.073	5,085E-03	347	3,47E-06	38.433.054	0,512
15	19.444.973	7,310E-03	406	4,06E-06	54.860.287	0,731
17	26.824.933	1,008E-02	464	4,64E-06	75.162.510	1,002
19	35.736.302	1,343E-02	522	5,22E-06	99.717.261	1,330
21	46.458.946	1,747E-02	581	5,81E-06	128.903.662	1,719
23	59.010.655	2,218E-02	639	6,39E-06	163.102.419	2,175
25	73.520.863	2,764E-02	697	6,97E-06	202.695.822	2,703
27	90.203.243	3,391E-02	756	7,56E-06	248.067.745	3,308
29	109.161.208	4,104E-02	814	8,14E-06	299.603.646	3,995
31	130.649.765	4,912E-02	872	8,72E-06	357.690.567	4,769
33	154.695.761	5,816E-02	931	9,31E-06	422.717.134	5,636
35	181.439.394	6,821E-02	989	9,89E-06	495.073.557	6,601
37	211.321.186	7,944E-02	1047	1,04E-05	575.151.630	7,669
39	243.870.849	9,168E-02	1106	1,10E-05	663.344.731	8,845

Πίνακας 5.5: Κύκλοι Μηχανής και χρόνοι αναλυτών για τη G_{Java} και για διαφορετικό μήκος συμβολοσειρών εισόδου.

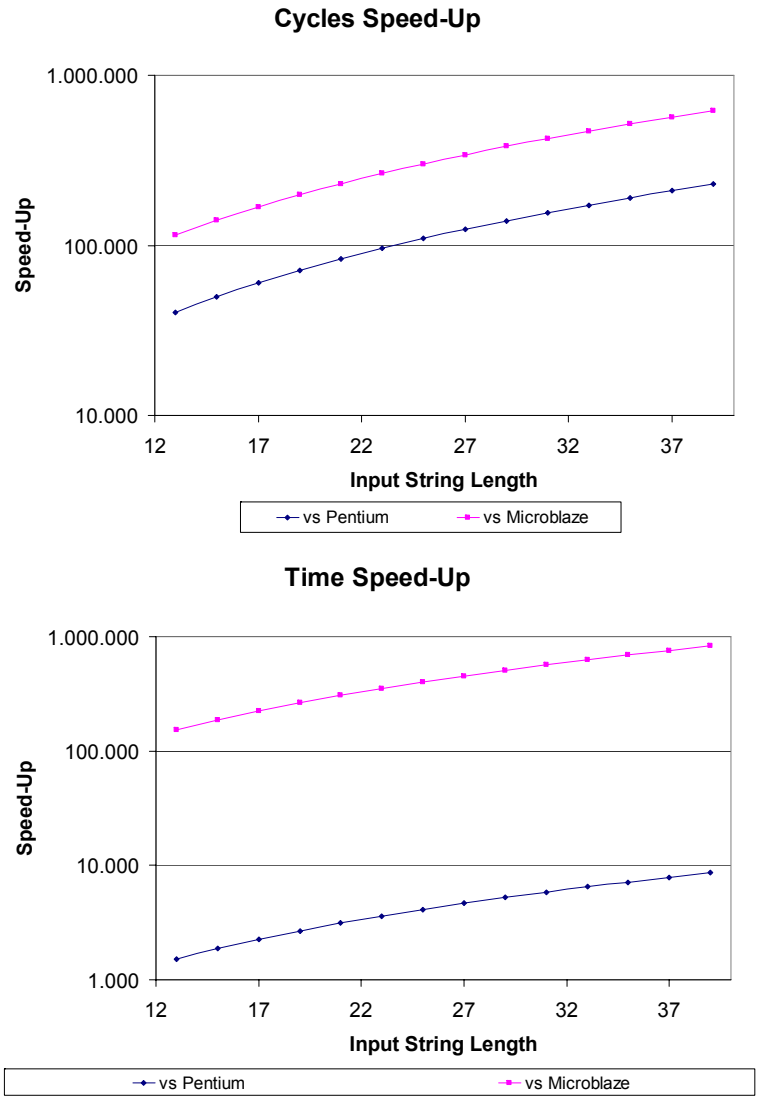


Σχήμα 5.13: Κύκλοι ρολογιού και χρόνοι αναλυτών για τη G_{Java} και για διαφορετικό μήκος συμβολοσειρών εισόδου.

Όλα τα παραπάνω, απεικονίζονται στα σχήματα 5.13 και 5.14.

5.4.4 Σχολιασμός πειραματικών αποτελεσμάτων

Ο λόγος που η απόδοση της προτεινόμενη υλοποίησης παραμένει σχεδόν η ίδια για όλες τις υπό εξέταση γραμματικές (και για το ίδιο μήκος συμβολοσειράς εισόδου), οφείλεται στο γεγονός ότι η καθυστέρηση για το κύκλωμα είναι μικρότερη από έναν κύκλο ρολογιού. Σε περίπτωση που το βάθος των πυλών του κυκλώματος $C_{\otimes}$ επιβάλει καθυστέρηση μεγαλύτερη του ενός κύκλου ρολογιού (όπως στην περίπτωση των υποενοτήτων 5.4.2 και 5.4.3), η συνολική απόδοση ελαττώνεται κατά ένα μικρό παράγοντα για κάθε επιπλέον κύκλο ρολογιού που χρειάζεται. Η μικρή ελάττωση της απόδοσης, οφείλεται στην αύξηση του αριθμού των πυλών που απαρτίζουν το κύκλωμα $C_{\otimes}$, λόγω

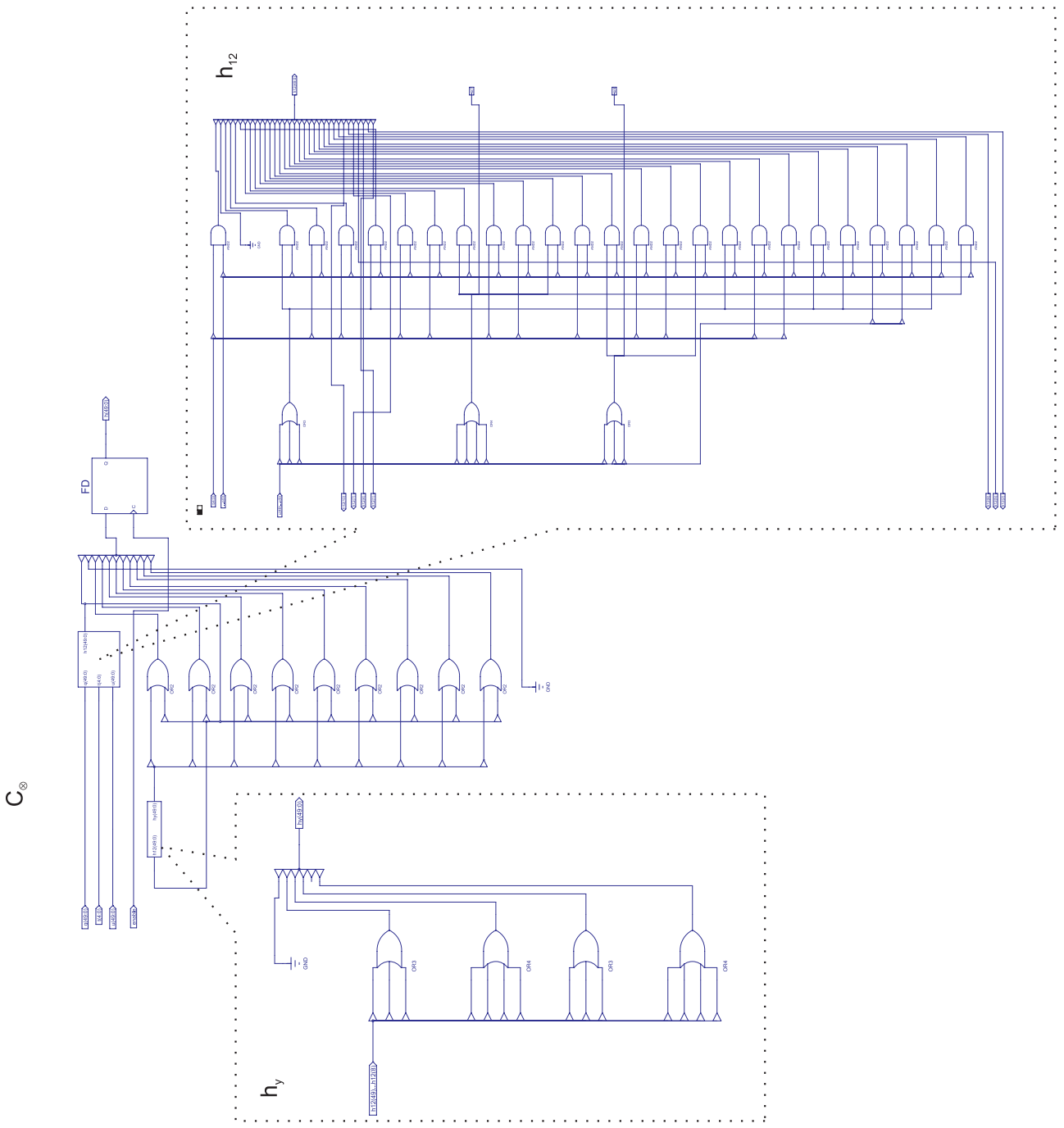


Σχήμα 5.14: Επιτάχυνση αναφορικά με τους κύκλους μηχανής και τους χρόνους των αναλυτών για τη G_{Java} και για διαφορετικό μήκος συμβολοσειρών εισόδου.

του μεγάλου μεγέθους της γραμματικής. Επομένως, το μόνο τμήμα της συνολικής αρχιτεκτονικής που καθυστερεί είναι το $C_{\otimes}$. Οι αναγκαίοι κύκλοι ρολογιού για τον υπολογισμό ενός κελιού, είναι ανεξάρτητοι του μήκους της συμβολοσειράς εισόδου αφού όλα τα απαραίτητα κυκλώματα $C_{\otimes}$ εργάζονται παράλληλα. Εντούτοις, το μήκος της συμβολοσειράς εισόδου επηρεάζει το χρόνο εκτέλεσης, αφού αυξάνει τον αριθμό των βημάτων εκτέλεσης και επικοινωνίας. Επομένως, για δύο διαφορετικές γραμματικές με αντίστοιχο μέγεθος, τέτοιο ώστε να επιτρέπει την εκτέλεση του κυκλώματος $C_{\otimes}$ στον ίδιο χρόνο, ο συνολικός χρόνος εκτέλεσης για συμβολοσειρές εισόδου ίδιου μεγέθους θα είναι ίσος, ανεξαρτήτως από το πλήθος των κανόνων που δημιουργούνται στα κελιά του πίνακα PT.

Λαμβάνοντας υπόψη τα πειραματικά αποτελέσματα, εξάγεται το συμπέρασμα ότι η

επιτάχυνση, σε σχέση με την υλοποίηση σε λογισμικό, ποικίλει από δύο τάξεις μεγέθους για απλές εφαρμογές μέχρι έξη τάξεις μεγέθους για πραγματικές εφαρμογές. Όλες οι μετρήσεις ελήφθησαν με μέγιστο μήκος συμβολοσειράς 42 και πέτυχαν μια μέση ελάττωση του χρόνου εκτέλεσης από ms μέχρι μs. Πρέπει να τονιστεί ότι αν και το κέρδος σε χρόνο μοιάζει ασήμαντο για μια μόνο συμβολοσειρά εισόδου, στις πραγματικές εφαρμογές που μεγάλα μεγέθη συμβολοσειρών πρέπει να αναγνωριστούν σε πραγματικό χρόνο, αυτό το κέρδος συσσωρεύεται σε μια σημαντική ελάττωση του χρόνου εκτέλεσης. Παράδειγμα τέτοιων εφαρμογών είναι η G_{ECG}, όπου διαδοχικοί καρδιακοί κύκλοι πρέπει να αναλύονται σε πραγματικό χρόνο και η G_{Java}, όπου η μεταγλώττιση διαδοχικών εντολών μπορεί να απαιτείται.



Σχήμα 5.15: Αναλυτική Απεικόνιση του κυκλώματος $C_{\otimes}$ για τη G_{chrom}.

Κεφάλαιο 6

Αποτιμητής S-Attributed Κατηγορικών Γραμματικών Με Χρήση Μόνο Εξειδικευμένου Υλικού

Όπως αναφέρθηκε και στο κεφάλαιο 4, η αποτίμηση των κατηγορημάτων είναι μια διαδικασία χρονοβόρως, που προκαλεί την καθυστέρηση του συνολικού συστήματος. Ενώ η συντακτική αναγνώριση γίνεται ταχύτατα, η επικοινωνία μεταξύ του συντακτικού αναλυτή και του αποτιμητή, καθώς και ο χρόνος εκτέλεσης του αποτιμητή, ρίχνουν την απόδοση των υλοποιήσεων. Για το λόγο αυτό, γίνεται μια προσπάθεια για απαλοιφή της ανάγκης ύπαρξης επεξεργαστή γενικής χρήσης για την αποτίμηση και αντικατάστασής του από υλικό ειδικής χρήσης. Δηλαδή, να περιγραφεί με Verilog ένα υποσύστημα που θα συνεργάζεται με τον αναλυτή και θα είναι σε θέση να εκτελέσει τις εργασίες του επεξεργαστή. Αυτή η προσπάθεια, η οποία υποστηρίζει μόνο S-Attributed Γραμματικές, θα αναλυθεί στα ακόλουθα ενώ στο επόμενο κεφάλαιο παρουσιάζεται μια ακόμη πιο πλήρης και αποδοτική προσπάθεια.

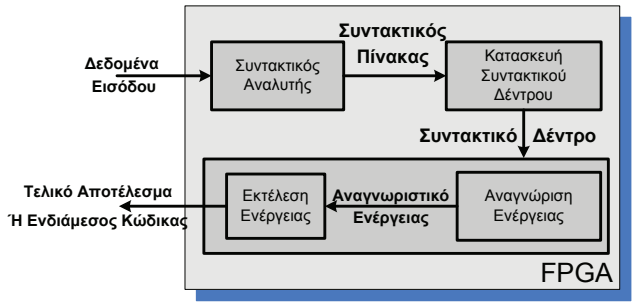
6.1 Υλοποίηση για S-Attributed Γραμματικές

Μια πλατφόρμα γενικής χρήσης υλοποιήθηκε για εφαρμογές που βασίζονται σε κατηγορικές γραμματικές. Ο χρήστης δίνει την κατηγορική γραμματική που περιγράφει την εφαρμογή του - συντακτικώς και σημασιολογικώς - και η πλατφόρμα παράγει τις κατάλληλες υπομονάδες για την αναγνώριση κάθε συμβολοσειράς εισόδου που ανήκει στην γραμματική, καθώς και εκείνες που είναι υπεύθυνες για την αποτίμηση των κατηγορημάτων. Η διαδικασία της αναγνώρισης βασίζεται στην υλοποίηση σε υλικό που

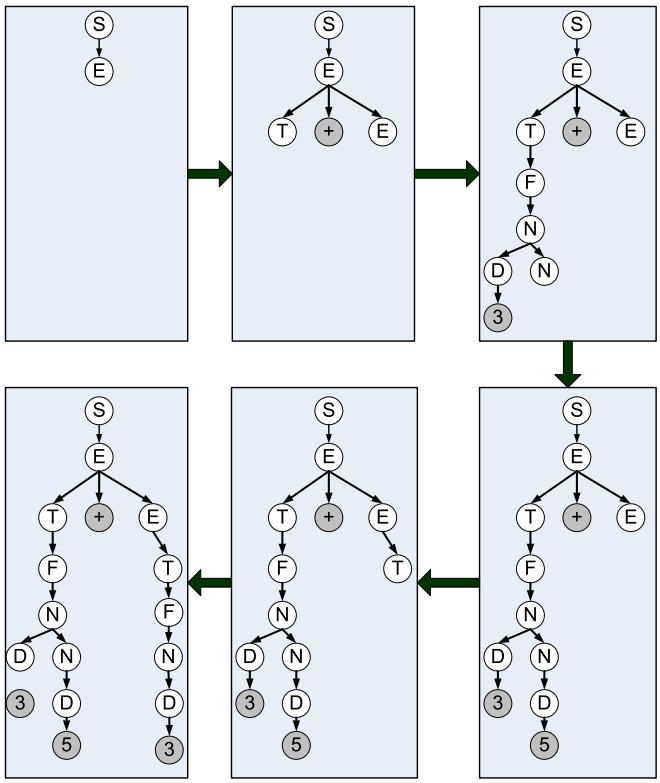
παρουσιάστηκε στο κεφάλαιο 3. Μέχρι τώρα γινόταν χρήση εξωτερικού επεξεργαστή (RISC) ή εσωτερικού μικροελεγκτή (PicoBlaze) ή και ακόμη εσωτερικού επεξεργαστή (MicroBlaze). Εδώ, η αποτίμηση των κατηγορημάτων θα γίνει σε μία ειδικής χρήσης μονάδα (για τη συγκεκριμένη εφαρμογή) στο ίδιο τσιπ που βρίσκεται και ο αναλυτής, άρα θα παραχθεί ένα System on a Chip (Soc). Η έξοδος του συστήματος, μετά την αποτίμηση των κατηγορημάτων μπορεί να εκτελεστεί από κάποιο άλλο κομμάτι υλικού ειδικού σκοπού ή ακόμη και από κάποιο απομακρυσμένο σύστημα. Επιπλέον, μια νέα μεθοδολογία αναπτύχθηκε για τον υπολογισμό των τιμών των κατηγορημάτων. Η υλοποίηση στοχεύει σε εφαρμογές υψηλού υπολογιστικού κόστους, που ζητούν την ταυτόχρονη επεξεργασία μεγάλου πλήθους προτάσεων, γι' αυτό ακολουθήθηκε η σε υλικό υλοποίηση. Η υλοποίηση έγινε σε Verilog με χρήση του περιβάλλοντος Xilinx ISE 9.1 ενώ δοκιμάστηκε σε ένα FPGA Xilinx Virtex-5 ML506.

Για την συγκεκριμένη υλοποίηση, οι χρησιμοποιούμενες κατηγορικές γραμματικές έχουν μόνο συντιθέμενα κατηγορήματα (S-attributed grammars) [7]. Μια υποκατηγορία αυτών των γραμματικών είναι οι γραμματικές ενεργειών (action grammars), οι οποίες από εδώ και κάτω θα θεωρούνται ως γραμματικές χωρίς συμφραζόμενα, στις οποίες όμως έχουν προστεθεί κάποιες ενέργειες στο τέλος των κανόνων. Αυτές οι ενέργειες (σημασιολογία) υπολογίζονται για κάθε κόμβο του συντακτικού δέντρου και όταν αυτό διασχίζεται από πάνω προς τα κάτω, εκτελούνται από αριστερά προς τα δεξιά μετά την συντακτική ανάλυση.

Η προτεινόμενη υλοποίηση ακολουθεί την αρχιτεκτονική που φαίνεται στο σχήμα 6.1, δηλαδή είναι χωρισμένη σε τρία μεγάλα τμήματα, υπεύθυνα για τη συντακτική αναγνώριση, την κατασκευή του συντακτικού δέντρου και την παραγωγή και εκτέλεση των αντιστοίχων ενεργειών. Ο συντακτικός αναλυτής αναλαμβάνει την αναγνώριση και την κατασκευή του συντακτικού πίνακα PT. Όταν τελειώσει η διαδικασία της συντακτικής ανάλυσης, κατασκευάζεται το συντακτικό δέντρο και καθώς διασχίζεται, οι αντίστοιχες ενέργειες στέλνονται προς εκτέλεση. Οι ενέργειες αυτές μπορεί να είναι ενδιάμεσος κώδικας που εκτελείται σε κάποια αφηρημένη μηχανή π.χ διακομιστής SQL, Java VM ή μπορεί να εκτελείται από κάποια εξειδικευμένη μονάδα που βρίσκε-



Σχήμα 6.1: Προτεινόμενη Αρχιτεκτονική.



Σχήμα 6.2: Κατασκευή Συντακτικού Δέντρου.

ται στο ίδιο FPGA με τις υπόλοιπες υπομονάδες της υλοποίησης. Στο παράδειγμα της υποενότητα 6.1.1, μια εξειδικευμένη μονάδα κάνει αριθμητικές πράξεις, ενώ σε εκείνο της υποενότητα 6.1.2 οι ενέργειες μετασχηματίζονται σε ερωτήματα SQL που εκτελούνται σε έναν SQL διακομιστή.

Για κάθε κελί του PT, ο αναλυτής έχει επεκταθεί ώστε να κρατάει επιπλέον πληροφορίες για την προέλευσή του, πληροφορίες που χρησιμοποιούνται για την επόμενη διαδικασία της κατασκευής του συντακτικού δέντρου. Βάσει του συντακτικού πίνακα PT, δημιουργείται το συντακτικό δέντρο. Αρχικά, εντοπίζεται η ρίζα του και μετά διαδοχικά ο επόμενος κόμβος, μέχρι να δημιουργηθεί το πλήρες συντακτικό δέντρο. Η αναζήτηση του επόμενου στοιχείου βασίζεται στα στοιχεία προέλευσης του τρέχοντος στοιχείου - πού πρωτοδημιουργήθηκε ή από πού ήρθε - έτσι ώστε να μην χρειάζεται πάλι "συντακτική αναγνώριση" ο πίνακας PT. Επομένως, βάσει των στοιχείων προέλευσης μπορεί να κατασκευαστεί το δέντρο, χωρίς χρονοβόρες αναζητήσεις στον πίνακα PT. Σημειώνεται, ότι το δέντρο σχηματίζεται από τα πάνω προς τα κάτω και από αριστερά προς τα δεξιά, όπως απεικονίζεται και στο σχήμα 6.2.

Ο τελικός στόχος, αυτός της αποτίμησης των κατηγορημάτων, εκτελείται μετά την πλήρη κατασκευή του συντακτικού δέντρου. Υπάρχει μια ένα προς ένα αντιστοιχία μεταξύ συντακτικών και σημασιολογικών κανόνων. Όπως έχει τονιστεί, οι συμβατικές μέθοδοι αποτίμησης κατηγορημάτων, βασίζονται στην κατασκευή του συντακτικού

δέντρου και τη διάσχισή του μία ή περισσότερες φορές, ανάλογα με τη μορφή της γραμματικής, για την αποτίμηση των κατηγορημάτων. Στην προτεινόμενη προσέγγιση, το συντακτικό δέντρο διατρέχεται από κάτω προς τα επάνω και από δεξιά προς τα αριστερά, και για κάθε κόμβο οι αντίστοιχοι σημασιολογικοί κανόνες αποτιμόνται. Η βασική ιδέα του αλγορίθμου είναι να αποτιμηθούν πρώτα όλοι οι υποκόμβοι ενός κόμβου και κατόπιν ο ίδιος ο κόμβος. Επιπλέον, για κάθε σημασιολογικό κανόνα, κατά τη διαδικασία της αποτίμησης καμία πραγματική λειτουργία δεν γίνεται αλλά σήματα ελέγχου - ενέργειες - στέλνονται για να εκτελεστούν αργότερα. Συνεπώς, κάθε σημασιολογικός κανόνας αντιστοιχεί σε μια ενέργεια και μετά την κατασκευή του συντακτικού δέντρου, μια υπομονάδα παράγει τις ενέργειες αυτές, διασχίζοντας μια μόνο φορά το δέντρο. Στη συνέχεια, οι ενέργειες αυτές εκτελούνται από μια άλλη υπομονάδα, η οποία χρησιμοποιεί μια μεθοδολογία με στοίβες (stack-based approach) για να εκτελέσει τους αναγκαίους υπολογισμούς. Ο μετασχηματισμός από το συμβολισμό των κατηγορικών γραμματικών, σε εκείνο των Γραμματικών Ενεργειών είναι μια απλή εργασία, όπως εξηγείται παρακάτω και μπορεί εύκολα να αυτοματοποιηθεί με τη χρήση ενός προεπεξεργαστή (preprocessor) που αποτελεί άλλωστε μέρος μελλοντικής εργασίας. Οι απαιτούμενες ενέργειες της μεθοδολογίας περιγράφονται ακολούθως:

- Για κάθε συντιθέμενο κατηγόρημα ορίζεται μια στοίβα.
- Καθώς το δέντρο διατρέχεται, όταν βρεθεί το τελευταίο σύμβολο ενός κανόνα, μπορεί να αποτιμηθεί η συντιθέμενη τιμή του κατηγορήματος του συμβόλου που βρίσκεται στο αριστερό μέλος του κανόνα, εξάγοντας από την στοίβα (unstacking - pop up) τις τιμές των αντίστοιχων κατηγορημάτων των συμβόλων που απαρτίζουν το δεξιό μέλος του κανόνα. Το αποτέλεσμα που προκύπτει βάσει των σημασιολογικών κανόνων, αποθηκεύεται (pushed) στην κατάλληλη στοίβα που αντιστοιχεί στο κατηγόρημα.
- Εάν από το σημασιολογικό κανόνα ορίζεται απλώς μια ανάθεση, τότε καμία ενέργεια δεν γίνεται.

Η υπολογιστική ισχύς του προτεινόμενου συστήματος είναι αυτή των γραμματικών S-attributed [7]. Το Yacc [29] έχει την ίδια υπολογιστική ισχύ, οπότε μπορεί να λεχθεί ότι το προτεινόμενο σύστημα αποτελεί μια υλοποίηση σε υλικό ενός μέτα-αναλυτή Yacc (Yacc metacompiler). Ωστόσο, ο αναλυτής που χρησιμοποιείται από την υλοποίηση, είναι πολύ πιο ισχυρός από εκείνον του Yacc. Είναι μη ντετερμινιστικός και βρίσκει όλες τις διφορούμενες (ambiguous) λύσεις. Τόσο ο αναλυτής, όσο και οι δυο υπομονάδες που αναφέρθηκαν παραπάνω, μπορούν να παραχθούν αυτόματα για οποιαδήποτε κατηγορική γραμματική, αφού προηγουμένως μετατραπεί σε Γραμματική Ενεργειών.

Ακολούθως, δύο παραδείγματα με δύο διαφορετικές γραμματικές θα δοθούν. Στο

πρώτο, αριθμητικές εκφράσεις περιγράφονται μέσω μιας κατηγορικής γραμματικής. Οι σημασιολογικοί κανόνες μετατρέπονται σε απλές ενέργειες push και pop, οι οποίες εκτελούνται από μια ξεχωριστή υπομονάδα στο ίδιο FPGA. Στο δεύτερο παράδειγμα, μια διεπαφή για Φυσικές Γλώσσες (Natural Language interface) αναπτύσσεται, η οποίο μετασχηματίζει ερωτήσεις από τα Αγγλικά σε ερωτήματα SQL. Οι σημασιολογικοί κανόνες έχουν μετατραπεί σε ενέργειες, που κατασκευάζουν ερωτήματα SQL, τα οποία εκτελούνται σε έναν απομακρυσμένο SQL διακομιστή.

6.1.1 Υπολογισμός Αριθμητικών Εκφράσεων

Σε μια πρώτη προσέγγιση, ένα ενδεικτικό παράδειγμα δίνεται, βασισμένο στην κατηγορική γραμματική G_{op} (βλέπε πίνακα 6.1), η οποία περιγράφει τις βασικές αριθμητικές πράξεις της πρόσθεσης και του πολλαπλασιασμού μεταξύ δύο ή περισσότερων τελεστέων. Το παραγόμενο σύστημα μπορεί να αναγνωρίσει συμβολοσειρές που περιγράφουν αριθμητικές πράξεις και επιπλέον να υπολογίσει το τελικό αποτέλεσμα, θυμίζοντας ένα κοινό κομπιουτεράκι. Σύμφωνα με τη μεθοδολογία που παρουσιάστηκε στην προηγούμενη ενότητα, μόνο μια στοίβα θα χρειαστεί αφού στην γραμματική G_{op} υπάρχει μόνο ένα συντιθέμενο κατηγόρημα. Επιπλέον, για την περίπτωση κανόνων που γίνεται μια απλή ανάθεση, καμία ενέργεια δεν γίνεται. Για τους υπόλοιπους κα-

Συντακτικός Κανόνας	Σημασιολογικός Κανόνας σε συμβολισμό Κατηγορικών Γραμματικών	Αντίστοιχη Ενέργεια σε συμβολισμό Γραμματικών Ενεργειών
$S \rightarrow E$	$S_s = E_s$	[pop result]
$E_1 \rightarrow T + E_2$	$E_{1s} = T_s + E_{2s}$	[pop x] [pop y] [evaluate x+y][push result]
$E \rightarrow T$	$E_s = T_s$	-
$T \rightarrow F * T$	$T_s = F_s * T_s$	[pop x] [pop y] [evaluate x*y][push result]
$T \rightarrow F$	$T_s = F_s$	-
$F \rightarrow (E)$	$F_s = E_s$	-
$F \rightarrow N$	$F_s = N_s$	-
$N \rightarrow DN$	$N_s = 10 * D_s + N_s$	[pop x] [pop y] [evaluate 10y+x][push result]
$N \rightarrow D$	$N_s = D_s$	-
$D \rightarrow 0$	$D_s = 0$	[push 0]
...	...	...
$D \rightarrow 9$	$D_s = 9$	[push 9]

Πίνακας 6.1: Γραμματική Αριθμητικών Πράξεων G_{op}.

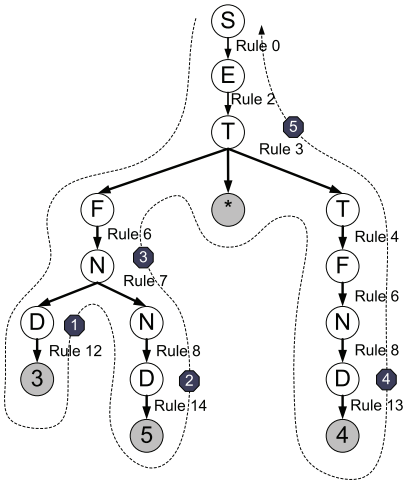
Κόμβος	Ενέργεια	Στοίβα
1	push 3;	3
2	push 5;	3, 5
3	pop x; pop y; push 10y+x;	35
4	push 3 ;	35, 3
5	push 4;	35, 3, 4
6	push 2;	35, 3, 4, 2
7	push 0;	35, 3, 4, 2, 0
8	pop x; pop y; push 10y+x;	35, 3, 4, 20
9	pop x; pop y; push x+y;	35, 3, 24
10	pop x; pop y; push x*y;	35, 72
11	pop x; pop y; push x+y;	107
12	pop result;	-

Πίνακας 6.2: Εκτέλεση Ενεργειών G_{op}.

νόνες, όλοι οι σημασιολογικοί κανόνες έχουν μετασχηματιστεί σε απλές ενέργειες push και pop.

Συμβατικές μέθοδοι, θα αναγνώριζαν εάν η συμβολοσειρά εισόδου $35 + 3 * (4 + 20)$ ανήκει στην G_{op}, βάσει των συντακτικών κανόνων. Και σε περίπτωση επιτυχίας, το συντακτικό δέντρο θα δημιουργείτο (σχήμα 6.3). Σε επόμενο βήμα, το συντακτικό δέντρο θα διασχιζόταν και κάθε φορά που ένας κόμβος του θα διασχιζόταν, οι σημασιολογικοί κανόνες θα υπολογίζονταν.

Στην προτεινόμενη προσέγγιση, οι σημασιολογικοί κανόνες έχουν μετασχηματιστεί σε ενέργειες, όπως φαίνεται στην $3^η$ στήλη του πίνακα 6.1. Για κάθε κόμβο του δέντρου, έχει παραχθεί το αντίστοιχο αναγνωριστικό ενέργειας. Η τελευταία υπόμονάδα, λαμβάνει αυτές τις απλές ενέργειες και τις εκτελεί. Με αυτό τον τρόπο,



Σχήμα 6.3: Συντακτικό Δέντρο για Είσοδο 35+3*(4+20).

το τελικό αποτέλεσμα (ο αριθμός 107) εμφανίζεται στην έξοδο του συστήματος. Οι παραγόμενες ενέργειες, καθώς και τα περιεχόμενα της στοίβας για την συμβολοσειρά εισόδου $35+3*(4+20)$ παρουσιάζονται στον πίνακα 6.2. Η 1^{η} στήλη του πίνακα 6.2 αναφέρεται στους αριθμημένους κόμβους του συντακτικού δέντρου που παρουσιάζεται στο σχήμα 6.3.

6.1.2 Αναγνώριση Φυσικών Γλωσσών

Η επεξεργασία φυσικών γλωσσών, είναι μια ιδιαίτερα ελκυστική μέθοδος αλληλεπίδρασης ανθρώπου-υπολογιστή και μπορεί να εφαρμοστεί σε μεγάλο αριθμό τομέων όπως τα ευφυή ενσωματωμένα συστήματα, ευφυείς διεπαφές, συστήματα γνώσης (learning systems) κ.α. [72], [73]. Είναι σαφές, ότι η αυτόματη εξόρυξη γλωσσικών πληροφοριών από ένα κείμενο είναι μια εξαιρετικά ισχυρή μέθοδος για συστήματα επεξεργασίας φυσικών γλωσσών.

Προκειμένου να φανεί πώς μπορεί να δημιουργηθεί μια διεπαφή φυσικής γλώσσας με χρήση του προτεινόμενου συστήματος, δίνεται ένα παράδειγμα ερώτησης-απόκρισης από την περιοχή των αεροπορικών πτήσεων. Το σύστημα μπορεί να λάβει προτάσεις που ανήκουν σε ένα υποσύνολο της Αγγλικής γλώσσας. Όταν τελειώσει η συντακτική αναγνώριση της πρότασης, με χρήση του κατασκευασθέντος συντακτικού δέντρου, οι σημασιολογικοί κανόνες αποτιμώνται και το FPGA στέλνει τα ερωτήματα SQL που δημιουργήθηκαν σε μια μηχανή διαχείρισης δεδομένων, προκειμένου να πα-

Συντακτικοί κανόνες	
$PR \rightarrow NP\ VP$	$SB \rightarrow OB$
$NP \rightarrow QS\ W\ VP$	$SB \rightarrow OB\ P\ OB$
$NP \rightarrow QS$	$QF \rightarrow A\ SET$
$NP \rightarrow OB$	$QF \rightarrow E\ SET$
$QS \rightarrow A\ SET$	$OB \rightarrow Object\ Names$
$QS \rightarrow E\ SET$	$SET \rightarrow Class\ Names$
$VP \rightarrow VP_1\ VP_2\ SP$	$REL \rightarrow Relation\ Phrases$
$VP \rightarrow SP$	$AT \rightarrow Property\ Names$
$VP_1 \rightarrow IP\ VP_1$	$NRL \rightarrow Numerical\ Relations$
$VP_2 \rightarrow SP\ CNJ\ VP_2$	$N \rightarrow Numbers$
$IP \rightarrow REL\ QF\ W$	$W \rightarrow Relative\ Pronouns$
$SP \rightarrow REL\ SB$	$A \rightarrow Determiner$ "A"
$SP \rightarrow AT\ NRL\ N$	$Each \rightarrow Determiner$ "Each"
$SB \rightarrow QF$	$CNJ \rightarrow Conjuction\ Words$

Πίνακας 6.3: Γραμματική Φυσικής Γλώσσας G_{NL}.

Συντακτικός Κανόνας	Ενέργειες
$PR \rightarrow NP\ VP$	pop x; pop y; push conc(y,x,;)
$QS \rightarrow A\ SET$	pop x; pop y; push conc(y,x)
$VP \rightarrow SP$	
$SP \rightarrow REL\ SB$	pop x; pop y; push conc(y,x)
$SB \rightarrow OB$	
$OB \rightarrow Athens$	push ""ATH""
$SET \rightarrow Flights$	push "from Flights"
$REL \rightarrow Departs\ From$	push "where DepartsFrom="
$A \rightarrow A$	push "select FlightNo"

Πίνακας 6.4: Αντιστοιχία Ενεργειών για G_{NL}.

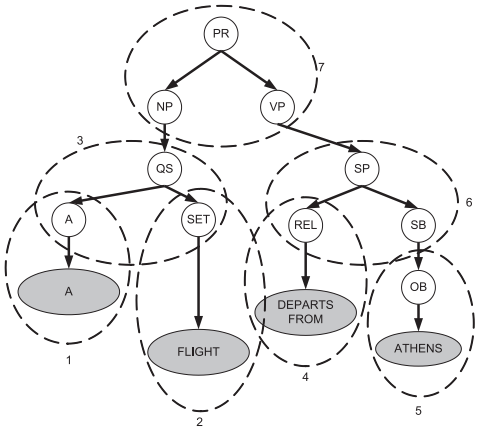
ραχθεί το τελικό αποτέλεσμα - η απάντηση. Το σύστημα μπορεί να χρησιμοποιηθεί σε εφαρμογές όπου πολλές προτάσεις υποβάλλονται για επεξεργασία ταυτόχρονα και πληροφορίες για τα ερωτήματα εξάγονται. Τέτοιες πληροφορίες, μπορούν να είναι στατιστικά στοιχεία για τις προτιμήσεις των πελατών, το προφίλ των χρηστών, ευφυής εξόρυξη λέξεων κλειδιών για Web browsers κλπ.

Στον πίνακα 6.3 παρουσιάζονται οι συντακτικοί κανόνες της κατηγορικής γραμματικής G_{NL} που αφορούν στις αεροπορικές πτήσεις. Η συνολική γραμματική μπορεί να βρεθεί στο [74], όπου ένα υποσύνολο της Αγγλικής γίνεται δεκτό από το σύστημα. Οι προτάσεις που γίνονται δεκτές, είναι ερωτήσεις αναφορικά με αεροπορικές πτήσεις και η απάντηση μετά την επεξεργασία του ενδιάμεσου κώδικα είναι NAI ή OXI. Στο [74] ο ενδιάμεσος κώδικας εκτελείτο από ένα μηχάνημα που είχε πρόσβαση σε μια στοιχειώδη βάση δεδομένων. Αντιθέτως, εδώ ο ενδιάμεσος κώδικας είναι ερωτήματα SQL που μπορεί να διαχειριστεί ένας πραγματικός διακομιστής. Ένα απλό παράδειγμα είναι η ακόλουθη ερώτηση:

A FLIGHT DEPARTS FROM ATHENS?

Η συγκεκριμένη ερώτηση μπορεί να αναλυθεί συντακτικά σε μια "noun phrase", αποτελούμενη από έναν "determiner" και ένα "common noun" και μια "verb phrase" αποτελούμενη από ένα "verb", μια "preposition" και ένα "proper noun". Ο "determiner" δίνει ποσοτικά στοιχεία, το "common noun" το όνομα της κλάσης, το "verb" και η "preposition" μια σχέση και το "proper noun" ένα αντικείμενο. Τα "noun" και "verb" θα χρησιμοποιηθούν σαν παράμετροι από τις εντολές που θα παραχθούν.

Ακολουθώντας την ίδια πορεία με την προηγούμενη ενότητα, οι σημασιολογικοί κανόνες μετατρέπονται σε κατάλληλες ενέργειες και μια στοίβα χρησιμοποιείται. Για την ερώτηση "A FLIGHT DEPARTS FROM ATHENS?" το συντακτικό δέντρο παρουσιάζεται στο σχήμα 6.4, ενώ στον πίνακα 6.4 παρουσιάζονται οι μετασχηματισμένοι σημασιολογικοί κανόνες, μόνο για τους απαραίτητους κόμβους, Σημειώνεται ότι conc $(par_1, \ldots, par_n)$ είναι η συνένωση των περιεχομένων των $par_1, \ldots, par_n$. Για μία βάση



Σχήμα 6.4: Συντακτικό Δέντρο Για το Παράδειγμα Φυσικής Γλώσσας.

δεδομένων που αποτελείται από μια σχέση Flights: Flights(FlightNo id, DepartsFrom char[3], ArrivesAt char[3], Airline char[5], ...), το παραγόμενο σύστημα εξάγει το SQL ερώτημα: Select FlightNo From Flights Where DepartsFrom="ATH";, όπως φαίνεται και στον πίνακα 6.5, το οποίο μπορεί να σταλεί προς εκτέλεση σε ένα διακομιστή SQL.

Κόμβος	Ενέργεια	Στοίβα
1	push "select FlightNo"	"select FlightNo"
2	push "from Flights"	"select FlightNo", "from Flights"
3	pop x; pop y; push conc(y,x)	"select FlightNo from Flights"
4	push "where DepartsFrom="	"select FlightNo from Flights", "where DepartsFrom="
5	push ""ATH""	"select FlightNo from Flights", "where DepartsFrom=", ""ATH""
6	pop x; pop y; push conc(y,x)	"select FlightNo from Flights", "where DepartsFrom="ATH""
7	pop x; pop y; push conc(y,x,;)	"select FlightNo from Flights where DepartsFrom="ATH";"

Πίνακας 6.5: Αποτελέσματα Ενεργειών G_{NL}.

Κεφάλαιο 7

Αποτιμητής L-Attributed Κατηγορικών Γραμματικών Με Χρήση Μόνο Εξειδικευμένου Υλικού

Στο κεφάλαιο 6 παρουσιάστηκε μια μεθοδολογία για τη δημιουργία αποτιμητών κατηγορημάτων με τη χρήση μόνο εξειδικευμένου υλικού, με τη βοήθεια του συντακτικού αναλυτή που παρουσιάστηκε στο κεφάλαιο 3. Στο κεφάλαιο αυτό, θα παρουσιαστεί ένα πλήρες σύστημα για την αποτίμηση των κατηγορημάτων, το οποίο για το συντακτικό μέρος βασίζεται στον αναλυτή που παρουσιάστηκε στο κεφάλαιο 5 και στηρίζεται στην υλοποίηση του αλγορίθμου του Earley με συνδυαστική λογική. Το σημασιολογικό μέρος το χειρίζεται ένα ειδικής χρήσης λειτουργικό τμήμα (module), το οποίο διασχίζει το συντακτικό δέντρο και υπολογίζει τα κατηγορήματα (συντιθέμενα και κληρονομούμενα), βάσει μιας προτεινόμενης προσέγγισης με χρήση στοιβών (stack-based approach). Και εδώ, το συνολικό σύστημα έχει περιγραφεί στην HDL γλώσσα Verilog σε τέτοια μορφή ώστε με τη χρήση αυτοματοποιημένου εργαλείου να παράγεται το σύστημα για κάθε δοσμένη κατηγορική γραμματική.

7.1 Γενικά

Προκειμένου να επιτευχθεί όσο το δυνατόν μεγαλύτερη αποδοτικότητα, ειδική προσοχή έχει δοθεί σε κάθε ένα από τα δύο υποτμήματα που αποτελούν το προτεινόμενο σύστημα, το τμήμα που αναλαμβάνει το συντακτικό έλεγχο και εκείνο που αναλαμβάνει το σημασιολογικό.

Για το συντακτικό μέρος, ένας αποδοτικός αλγόριθμος έπρεπε να επιλεγεί, ικανός όμως και για τη δημιουργία του συντακτικού δέντρου που είναι αναγκαίο για την

επόμενη λειτουργία της αποτίμησης των κατηγορημάτων. Επιπλέον, στην περίπτωση διφορούμενων γραμματικών (ambiguous grammar) όλα τα δυνατά συντακτικά δέντρα πρέπει να δημιουργούνται, προκειμένου η υλοποίηση να μπορεί να εφαρμοστεί στο πεδίο της τεχνητής νοημοσύνης. Ο επιλεγμένος συντακτικός αναλυτής [75] μπορεί να χειριστεί οποιαδήποτε γραμματική, όμως δε δημιουργεί το συντακτικό δέντρο αλλά έναν πίνακα, στον οποίο αποθηκεύονται σε κάθε βήμα εκτέλεσης δεδομένα για όλα τα πιθανά δέντρα που μπορούν να παραχθούν για τη συμβολοσειρά εισόδου που έχει εξετασθεί μέχρι στιγμής. Συνεπώς, η αρχική υλοποίηση πρέπει να επεκταθεί αφενός μεν ώστε να παράγει όλα τα δυνατά συντακτικά δέντρα, διατηρώντας όμως την συνδυαστική της φύση, χάρη στην οποία επιτυγχάνεται η μεγάλη απόδοση και αφετέρου ώστε να υπολογίζει τα κατηγορήματα.

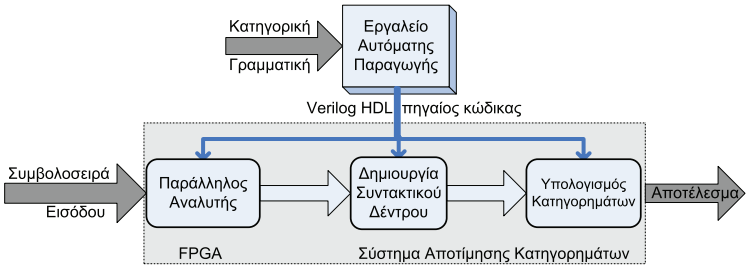
Όσον αφορά στο σημασιολογικό μέρος, η αποτίμηση δε λαμβάνει χώρα σε εξωτερικό επεξεργαστή ή σε εσωτερικό soft core, αλλά σε εξειδικευμένα λειτουργικά τμήματα. Τα τμήματα αυτά έχουν σχεδιαστεί ειδικά για την εκτέλεση των αναγκαίων πράξεων για τον υπολογισμό των κατηγορημάτων - συντιθέμενων και κληρονομούμενων - και συνεπώς είναι εξαιρετικά αποδοτικά. Τα τμήματα αυτά βασίζονται σε μια προσέγγιση με χρήση στοιβών, η οποία επιτρέπει τον υπολογισμό των κατηγορημάτων με χρήση κυρίως απλών εντολών push και pop σε συγκεκριμένες στοίβες. Συνεπώς, το προτεινόμενο σύστημα υλοποιείται σε ειδικής χρήσης υλικό και κατεβαίνει στο ίδιο FPGA, απαλείφοντας την ανάγκη για χρονοβόρες επικοινωνίες μεταξύ απομακρυσμένων υπομονάδων. Η φύση της υλοποίησης (υλικό) επιτρέπει την ταχεία επεξεργασία πλήθους εισόδων που πρέπει να εξυπηρετηθούν ταυτόχρονα.

Μάλιστα, η αρχιτεκτονική του συστήματος βασίζεται σε ένα πρότυπο, το οποίο επιτρέπει την αυτόματη παραγωγή για κάθε L-attributed γραμματική [7]. Τα αποτελέσματα για διάφορες γραμματικές ήταν πολύ ενθαρρυντικά, αφού η επιτάχυνση[1] συγκριτικά με προσεγγίσεις σε λογισμικό είναι δύο με τρεις τάξεις μεγέθους, ενώ ο χρόνος εκτέλεσης της προτεινόμενης υλοποίησης μία τάξη μεγέθους μικρότερος, όταν το λογισμικό εκτελείται σε σύγχρονους επεξεργαστές, που τρέχουν σε εξαιρετικά μεγαλύτερες συχνότητες. Ενώ συγκριτικά με υλοποιήσεις σε υλικό, η επιτάχυνση είναι μια με τρεις τάξεις μεγέθους ανάλογα με την υλοποίηση, το μέγεθος της γραμματικής και το μήκος της συμβολοσειράς εισόδου.

7.2 Περιγραφή του Συστήματος

Το προτεινόμενο σύστημα, όπως φαίνεται και στο σχήμα 7.1, αποτελείται από τρία κυρίως τμήματα: τον επεκταμένο συντακτικό αναλυτή, ένα τμήμα υπεύθυνο για την κατασκευή του συντακτικού δέντρου και ένα για την αποτίμηση των κατηγορημάτων.

[1] Η σύγκριση των αποδόσεων βασίζεται στους αναγκαίους κύκλους ρολογιού για την εκτέλεση, συνεπώς η επιτάχυνση αναφέρεται στο λόγο των καταναλισκόμενων κύκλων ρολογιού.



Σχήμα 7.1: Επισκόπηση προτεινόμενης υλοποίησης.

Ο αναλυτής αναλαμβάνει το έργο της συντακτικής αναγνώρισης και κατασκευάζει το συντακτικό πίνακα, βάσει της συμβολοσειράς εισόδου. Μόλις τελειώσει η διαδικασία αυτή, κατασκευάζεται το συντακτικό δέντρο και ακολούθως διασχίζεται και υπολογίζονται οι τιμές των κατηγορημάτων. Πρέπει να διευκρινισθεί ότι τα δύο πρώτα λειτουργικά τμήματα παράγονται αυτόματα βάσει της κατηγορικής γραμματικής, καθώς και μια γενική αρχιτεκτονική για τον υπολογισμό των κατηγορημάτων, η οποία είναι ανεξάρτητη από την γραμματική. Εάν η γραμματική επιβάλει σύνθετες πράξεις, τότε ειδικά λειτουργικά τμήματα χρειάζονται, τα οποία όμως ακόμη δε μπορούν να παραχθούν αυτόματα. Αντιθέτως πρέπει να δοθούν από το χρήστη, ανάλογα με τις απαιτήσεις της γραμματικής. Στην περίπτωση που οι σημασιολογικοί κανόνες αποτελούνται μόνο από απλές αριθμητικές πράξεις, κάτι αρκετά σύνηθες, το συνολικό σύστημα μπορεί να παραχθεί αυτόματα. Συνήθως προκειμένου να αναλυθούν μεγάλες συμβολοσειρές εισόδου, χωρίζονται σε τμήματα μικρότερου μεγέθους. Επομένως, τεχνικές pipeline θα μπορούσαν να εφαρμοστούν ανάμεσα στα τρία κυρίως τμήματα για τις περιπτώσεις μεγάλων συμβολοσειρών εισόδου, δίνοντας ακόμη καλύτερα αποτελέσματα. Προφανώς μια τέτοια υλοποίηση μπορεί να χρησιμοποιηθεί και στην περίπτωση που η συμβολοσειρά εισόδου αποτελείται από πολλές διαδοχικές αλλά ανεξάρτητες συμβολοσειρές μικρού μεγέθους.

7.2.1 Επεκτείνοντας το Συντακτικό Αναλυτή

Έχοντας επιλέξει τον αναλυτή που παρουσιάστηκε στο κεφάλαιο 5, ορισμένες τροποποιήσεις έπρεπε να γίνουν προκειμένου να μετατραπεί σε έναν αναλυτή κατάλληλο για κατηγορικές γραμματικές. Το συντακτικό δέντρο είναι αναγκαίο για την αποτίμηση των κατηγορημάτων, επομένως ο συνδυαστικός αναλυτής πρέπει να τροποποιηθεί ώστε να το κατασκευάζει. Στην περίπτωση μάλιστα διφορούμενων γραμματικών, όλα τα πιθανά δέντρα θα πρέπει να δημιουργούνται, προκειμένου να μπορεί το σύστημα να εφαρμοστεί στο πεδίο της τεχνητής νοημοσύνης. Τονίζεται ότι όλες οι επεκτάσεις γίνονται έτσι ώστε ο αναλυτής να διατηρεί τη συνδυαστική του φύση, η οποία αποφέρει τη μεγάλη απόδοση.

Η πιο ουσιώδης επέκταση είναι η αποθήκευση για κάθε δημιουργούμενο κανόνα με

τελεία της προέλευσής του. Λόγω της φύσης του τελεστή $\otimes$, ένας κανόνας με τελεία μπορεί να προκύψει από δύο κανόνες με τελεία ή από έναν κανόνα με τελεία και ένα τερματικό σύμβολο της συμβολοσειράς εισόδου. Επομένως για κάθε κανόνα με τελεία το πολύ δύο θέσεις προέλευσης αποθηκεύονται, δηλαδή τα κελιά που περιέχουν τους κανόνες βάσει των οποίο δημιουργήθηκε ο καινούργιος, σε διανύσματα δυαδικών ψηφίων. Η προέλευση κάθε κανόνα χρησιμοποιείται κατά τη διαδικασία κατασκευής του συντακτικού δέντρου, για τη σωστή τοποθέτηση του κανόνα στο δέντρο.

Σε κάθε βήμα εκτέλεσης, ο αναλυτής υπολογίζει τους κανόνες με τελεία για τα κελιά που ανήκουν στην ίδια διαγώνιο, όπως φαίνεται και στο σχήμα 5.2(a). Μετά το τέλος κάθε βήματος εκτέλεσης, μια στήλη του συντακτικού πίνακα έχει συμπληρωθεί πλήρως και συνεπώς τα περιεχόμενά της μπορούν να αποσταλούν στο επόμενο τμήμα, το οποίο είναι υπεύθυνο για τη δημιουργία του συντακτικού δέντρου, όπως φαίνεται και στο σχήμα 7.1. Λόγω της φύσης του παράλληλου αναλυτή, σε κάθε βήμα εκτέλεσης όλοι οι πιθανοί κανόνες δημιουργούνται και αποθηκεύονται στο συντακτικό πίνακα. Συνεπώς ο συντακτικός πίνακας περιέχει μεγάλο αριθμό κανόνων, κάποιοι εκ των οποίων δεν είναι καν χρήσιμοι για την αναγνώριση της δοσμένης συμβολοσειράς εισόδου. Επιπροσθέτως, για τη δημιουργία του συντακτικού δέντρου μόνο κανόνες με την τελεία στο τέλος χρησιμοποιούνται και επομένως μόνο τέτοιοι μεταδίδονται από τον αναλυτή στο επόμενο λειτουργικό τμήμα. Ο διαχωρισμός των κανόνων με την τελεία στο τέλος γίνεται εξαιρετικά απλά και γρήγορα με τη χρήση μασκών διανυσμάτων δυαδικών ψηφίων, που έχουν τιμή 1 μόνο στα ψηφία που αντιστοιχούν σε κανόνες με τελεία στο τέλος.

Μόλις διαβαστεί ολόκληρη η συμβολοσειρά εισόδου από τον αναλυτή, δηλαδή συμπληρωθεί και η τελική στήλη του συντακτικού πίνακα, αποστέλλεται στο επόμενο τμήμα και μπορεί να ξεκινήσει η διαδικασία κατασκευής του συντακτικού δέντρου. Η διαδικασία ξεκινά από το πάνω δεξιά κελί του συντακτικού πίνακα, στο οποίο αναζητείται ο αρχικός κανόνας της γραμματικής. Μόλις βρεθεί, τοποθετείται στη ρίζα του συντακτικού δέντρου και αναζητούνται οι κανόνες από τους οποίους προήλθε, με τη βοήθεια των διανυσμάτων δυαδικών ψηφίων που έχουν αποθηκευτεί για αυτό το λόγο. Με τον τρόπο αυτό, κλαδί-κλαδί κατασκευάζεται ολόκληρο το συντακτικό δέντρο. Εάν ένας κανόνας δημιουργήθηκε από περισσότερες από μια πράξεις $\otimes$, δηλαδή έχει περισσότερες από μια προελεύσεις, τότε πρόκειται για διφορούμενη γραμματική και περισσότερα του ενός συντακτικά δέντρα δημιουργούνται και η ίδια διαδικασία ακολουθείται - ξεχωριστά πια - για κάθε μια προέλευση. Για κάθε κανόνα που τοποθετείται στο δέντρο, η ίδια διαδικασία ακολουθείται αναδρομικά μέχρι να οδηγηθούμε σε κανόνα που ανήκει στην κύρια διαγώνιο του συντακτικού πίνακα, γεγονός που σηματοδοτεί το τέλος του συγκεκριμένου κλάδου του δέντρου και συνεπώς τη δημιουργία κόμβου φύλλου.

Άλλη μια λεπτομέρεια που πρέπει να ξεκαθαριστεί είναι ο τρόπος με τον οποίο από

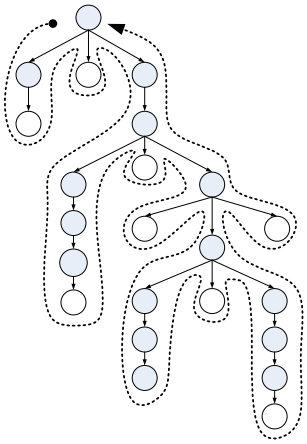
τους κανόνες με την τελεία στο τέλος διαχωρίζονται οι χρήσιμοι από τους υπολοίπους που βρίσκονται στο ίδιο κελί και άρα και στο ίδιο διάνυσμα δυαδικών ψηφίων. Προκειμένου να διατηρηθεί η συνδυαστική φύση του αναλυτή, μάσκες διανυσμάτων δυαδικών ψηφίων χρησιμοποιούνται για κάθε κανόνα. Αυτές οι μάσκες στηρίζονται στη σκεπτικό ότι κάθε κόμβος- γονέας στο συντακτικό δέντρο έχει κόμβους-παιδιά που έχουν ως σύμβολο στο αριστερό μέλος του κανόνα παραγωγής τους (left hand side symbol - lhss) μη τερματικά σύμβολα που βρίσκονται στο δεξιό μέλος του κανόνα παραγωγής (right hand side symbol - rhss) που αντιστοιχεί στον κόμβο-πατέρα. Προκειμένου να αποσαφηνιστεί αυτό, έστω ότι ο κανόνας $A \rightarrow BC$ αντιστοιχεί στον κόμβο-πατέρα. Οι κανόνες των κόμβων-παιδιών μπορούν να έχουν lhss μόνο τα μη τερματικά B ή C και κανένα άλλο. Επομένως, η κατάλληλη μάσκα για το συγκεκριμένο κανόνα επιτρέπει να περάσουν μόνο οι κανόνες με τα συγκεκριμένα lhss. Λόγω της κωδικοποίησης σε διανύσματα δυαδικών ψηφίων, οι μάσκες αυτές μπορούν να παραχθούν πολύ απλά και να εφαρμοστούν με τη βοήθεια λογικών πυλών KAI. Στην περίπτωση που υπάρχουν περισσότερα από δύο μη τερματικά σύμβολα στο δεξιό μέλος του κανόνα, πάλι δύο μάσκες χρησιμοποιούνται: μια για το τελευταίο, το πιο δεξιά σύμβολο και μια για όλα τα υπόλοιπα.

Η διαδικασία κατασκευής όλων των μασκών είναι απλή και υλοποιείται αυτόματα μαζί με την κατασκευή ολόκληρου του συστήματος. Πρέπει να σημειωθεί ότι η χρήση των μασκών δεν αυξάνει την πολυπλοκότητα του συντακτικού αναλυτή, αφού προστίθενται απλώς μερικές πύλες KAI και επομένως διατηρείται η συνδυαστική φύση και απόδοση. Περισσότερες λεπτομέρειες θα δοθούν στην παράγραφο 7.2.3 με χρήση παραδειγμάτων.

7.2.2 Προσέγγιση με χρήση Στοιβών

Με το πέρας των δύο τμημάτων (βλέπε 7.1) που αναλύθηκαν στην προηγούμενη παράγραφο, το συντακτικό δέντρο έχει κατασκευαστεί αλλά δεν έχει κατηγορήματα στους κόμβους του. Ο υπολογισμός των τιμών των κατηγορημάτων γίνεται από το τελευταίο λειτουργικό τμήμα, με χρήση της προτεινόμενης προσέγγισης μέσω στοιβών.

Για κάθε κατηγόρημα της γραμματικής, συντιθέμενο ή κληρονομούμενο, μια στοίβα ορίζεται με το ίδιο όνομα με το κατηγόρημα. Ο υπολογισμός των κατηγορημάτων του συντακτικού δέντρου ξεκινά από τη ρίζα του. Αρχικά, οι αρχικές τιμές εισάγονται στις αντίστοιχες στοίβες των κληρονομούμενων κατηγορημάτων. Η διαδικασία που ακολουθείται για κάθε κόμβο, συμπεριλαμβανομένου και της ρίζας, είναι ο υπολογισμός των τιμών των κατηγορημάτων για όλους τους κόμβους παιδία και κατόπιν ο υπολογισμός των κατηγορημάτων του τρέχοντος κόμβου. Ο υπολογισμός για τους κόμβους-παιδιά ξεκινάει από το πιο αριστερό και κατευθύνεται στο πιο δεξιό. Όλοι οι σημασιολογικοί κανόνες της γραμματικής μετατρέπονται σε ενέργειες εισαγωγής



Σχήμα 7.2: Διάσχιση συντακτικού δέντρου.

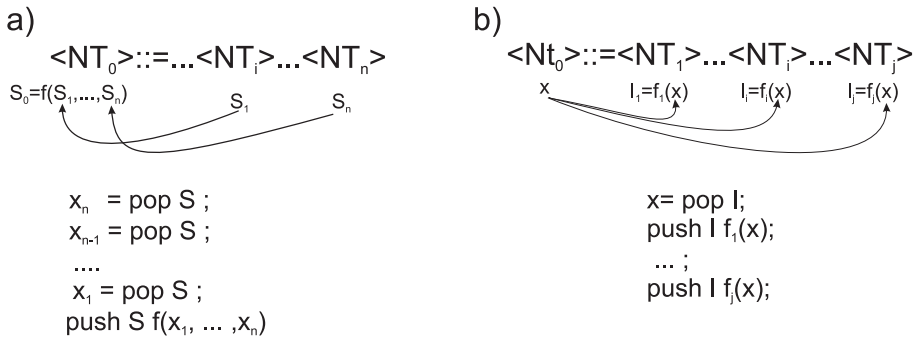
(push) και εξαγωγής (pop) δεδομένων στις στοίβες, ακολουθώντας μια μεθοδολογία που αναπτύσσεται παρακάτω και στηρίζεται στην αντίστοιχη της ενότητας 6.1. Σε κάθε κόμβο, οι ενέργειες εισαγωγής και εξαγωγής εκτελούνται ξεκινώντας από τον υπολογισμό των κληρονομούμενων κατηγορημάτων και ακολούθως υπολογίζονται οι τιμές των συντιθέμενων. Εξαιτίας της φύσης του αλγορίθμου, από πάνω προς τα κάτω και από αριστερά προς τα δεξιά, καθώς εισάγονται συντιθέμενες τιμές στις στοίβες, το δέντρο διατρέχεται προς το πιο αριστερό φύλλο του. Μόλις βρεθεί, υπολογίζονται τα συντιθέμενα κατηγορήματα και ο κόμβος-γονέας (ή ο δεξιός του κόμβος-αδερφός) μπορεί να αρχίσει τον υπολογισμό των δικών του συντιθέμενων κατηγορημάτων. Για πράξεις μεταξύ κατηγορημάτων αποθηκευμένων στην ίδια ή σε διαφορετική στοίβα, εξειδικευμένα κομμάτια υλικού χρησιμοποιούνται και το εξαγόμενο αποτέλεσμα εισάγεται πάλι στην κατάλληλη στοίβα. Προφανώς, το τέλος της συνολικής διαδικασίας σηματοδοτείται από τον υπολογισμό όλων των κατηγορημάτων του κόμβου ρίζα. Η διαδρομή που ακολουθείται για τον υπολογισμό των κατηγορημάτων ενός απλού συντακτικού δέντρου φαίνεται στο σχήμα 7.2.
Οι σημασιολογικοί κανόνες της γραμματικής μετατρέπονται σε ενέργειες εισαγωγής (push) και εξαγωγής (pop) δεδομένων στις στοίβες, ακολουθώντας την εξής μεθοδολογία:

- Αναφορικά με τα συντιθέμενα κατηγορήματα, κάθε φορά που η διάσχιση του δέντρου φθάνει σε φύλλο ή σε κόμβο του οποίου οι κόμβοι-παιδιά έχουν υπολογισμένα τα κατηγορήματα, οι συντιθέμενες τιμές των rhss εξάγονται από τη στοίβα. Αυτές οι συντιθέμενες τιμές βρίσκονται στην κορυφή των συγκεκριμένων στοιβών. Τότε το συντιθέμενο κατηγόρημα του κόμβου-γονέα υπολογίζεται σύμφωνα με τον αντίστοιχο σημασιολογικό κανόνα και εισάγεται στην κατάλληλη στοίβα. Με αυτό τον τρόπο εξασφαλίζεται ότι στην κορυφή των συντιθέμενων στοιβών, τα κατηγορήματα των κόμβων-παιδιών (μέχρι τον

τρέχοντα κόμβο-παιδί) είναι τοποθετημένα διαδοχικά. Στο σχήμα 7.3(a) παρουσιάζεται η συγκεκριμένη διαδικασία για ένα τυχαίο κανόνα με n μη τερματικά σύμβολα στο δεξιό μέλος, όπου γίνονται n διαδοχικές εξαγωγές από τη στοίβα S προκειμένου να αποκτηθούν όλα τα απαραίτητα κατηγορήματα για τον υπολογισμό του συντιθέμενου κατηγορήματος του lhss και κατόπιν η τιμή αυτή εισάγεται στην κορυφή της στοίβας S.

- Αναφορικά με τα κληρονομούμενα κατηγορήματα, μια εξαγωγή από τη στοίβα γίνεται την πρώτη φορά που ένας κόμβος διασχίζεται και χρειάζεται η τιμή του κληρονομούμενου κατηγορήματος I. Εάν σε ένα κανόνα η τιμή ενός κληρονομούμενου κατηγορήματος χρησιμοποιείται σε περισσότερα από ένα rhss, τότε το πλήθος εισαγωγών στην αντίστοιχη στοίβα πρέπει να γίνει. Αυτό έγκειται στο ότι κάθε φορά που μια τιμή από τη στοίβα χρειάζεται, εξάγεται και άρα παύει να υπάρχει στη στοίβα. Όλα τα παραπάνω παρουσιάζονται στο σχήμα 7.3(b), όπου η τιμή εξόδου των κατάλληλων σημασιολογικών κανόνων $(f_1, \ldots, f_j)$ που παίρνουν ως παράμετρο την τιμή του κληρονομούμενου κατηγορήματος I του lhss, εισάγεται στην στοίβα I.

- Για τους σημασιολογικούς κανόνες που επιβάλουν απλώς μεταφορά της τιμής (για το ίδιο κατηγόρημα), καμία ενέργεια δεν χρειάζεται - τόσο για τα συντιθέμενα όσο και για τα κληρονομούμενα κατηγορήματα - αφού οι ισοδύναμες ενέργειες θα ήταν μια διαδοχική εξαγωγή και εισαγωγή στην ίδια στοίβα.



Σχήμα 7.3: a) Συντιθέμενο κατηγόρημα S, b) Κληρονομούμενο κατηγόρημα I.

7.2.3 Διευκρινιστικό Παράδειγμα

Προκειμένου να επεξηγηθεί αναλυτικότερα η μεθοδολογία, παρουσιάζεται ένα παράδειγμα βασισμένο στην κατηγορική γραμματική G_{op}, η οποία παρουσιάστηκε στο προηγούμενο κεφάλαιο και για λόγους ευκολία ξαναγράφεται στον πίνακα 7.1 και περιγράφει τις βασικές αριθμητικές πράξεις της πρόσθεσης και του πολλαπλασιασμού

α/α κανόνα	Συντακτικός Κανόνας	Σημασιολογικός Κανόνας στη σημειογραφία των AG	Αντίστοιχες Ενέργειες Στοίβας
0	$S \to E$	$S_s = E_s$	pop result;
1	$E_1 \to T + E_2$	$E_{1s} = T_s + E_{2s}$	pop x; pop y; evaluate x+y; push result
2	$E \to T$	$E_s = T_s$	-
3	$T \to F * T$	$T_s = F_s * T_s$	pop x; pop y; evaluate x*y; push result;
4	$T \to F$	$T_s = F_s$	-
5	$F \to (E)$	$F_s = E_s$	-
6	$F \to N$	$F_s = N_s$	-
7	$N \to DN$	$N_s = 10 * D_s + N_s$	pop x; pop y; evaluate 10y+x; push result;
8	$N \to D$	$N_s = D_s$	-
9	$D \to 0$	$D_s = 0$	push 0;
	...	...	...
18	$D \to 9$	$D_s = 9$	push 9;

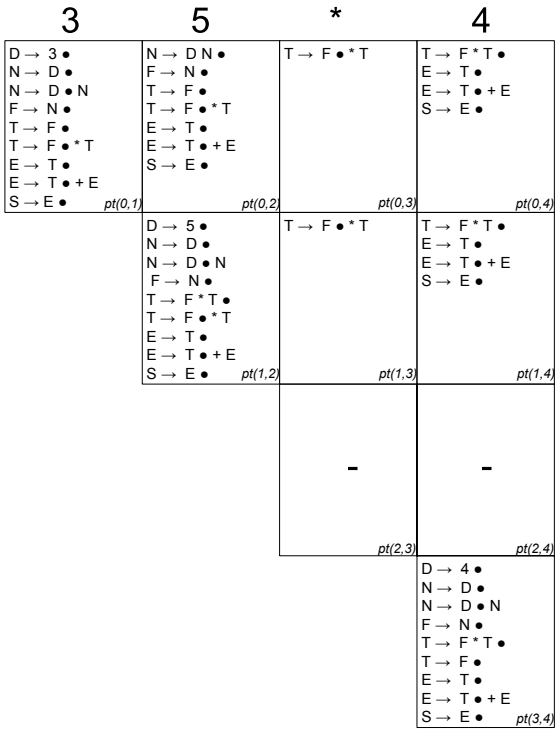
Πίνακας 7.1: Γραμματική Αριθμητικών Πράξεων G_{op}.

μεταξύ δύο ή περισσοτέρων τελεστέων. Το παραγόμενο σύστημα μπορεί να αναγνωρίσει συμβολοσειρές εισόδου που περιγράφουν αριθμητικές πράξεις και επιπλέον να υπολογίσει το τελικό αποτέλεσμα, θυμίζοντας ένα κλασικό κομπιουτεράκι. Σύμφωνα με τη μεθοδολογία που παρουσιάστηκε στην προηγούμενη παράγραφο, μόνο μια στοίβα χρειάζεται αφού μόνο ένα συντιθέμενο κατηγόρημα S ορίζεται από τη γραμματική G_{op}. Επιπλέον, για την περίπτωση των κανόνων που μόνο μια απλή ανάθεση υπαγορεύεται, καμία ενέργεια δεν χρειάζεται. Όλοι οι άλλοι σημασιολογικοί κανόνες έχουν μετασχηματισθεί σε απλές ενέργειες εισαγωγής και εξαγωγής στη στοίβα S και σε απλές αριθμητικές πράξεις, όπως φαίνεται και στον πίνακα 7.1.

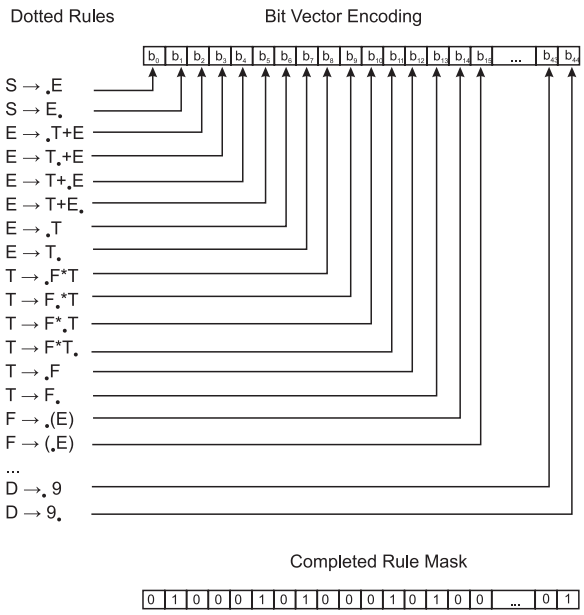
Μόλις εισαχθεί η έκφραση $35 * 4$ στο σύστημα, η πρώτη ενέργεια είναι η κατασκευή του συντακτικού πίνακα από τον επεκταμένο συντακτικό αναλυτή, ο οποίος φαίνεται στο σχήμα 7.4, χωρίς την κύρια διαγώνιο αφού περιέχει αρχικούς κανόνες που έχουν προϋπολογιστεί. Η αναπαράσταση στο σχήμα δεν είναι σε διάνυσμα δυαδικών ψηφίων για λόγους απλότητας. Στην πραγματικότητα, η αναπαράσταση γίνεται σε διανυσμάτων δυαδικών ψηφίων μήκους 45, όπως επιβάλλεται από το δυνατό αριθμό

των κανόνων με τελεία που μπορούν να προκύψουν από τους 19 συντακτικούς κανόνες της γραμματικής (βλέπε σχήμα 7.5). Ο συντακτικός πίνακας περιέχει κανόνες με την τέλεια όχι μόνο στο τέλος και επομένως μόνο οι κανόνες με την τελεία στο τέλος πρέπει να διατηρηθούν. Το τελευταίο επιτυγχάνεται με τη χρήση της μάσκας που έχει τιμή 1 μόνο στα ψηφία που αντιστοιχούν στη θέση κανόνων με την τελεία στο τέλος. Η μάσκα αυτή παράγεται αυτόματα και για τη γραμματική G_{op} έχει τιμή 10101010101010101010100101000101000101000010100010.

Μετά την εφαρμογή της μάσκα, ο συντακτικός πίνακας μετασχηματίζεται σε εκείνο του σχήματος 7.6, όπου χρησιμοποιείται η πραγματική απεικόνιση σε διανύσματα δυαδικών ψηφίων. Προφανώς το αναγκαίο μέγεθος των διανυσμάτων δυαδικών ψηφίων είναι μόνο 19, όσο αρκεί για την κωδικοποίηση των 19 κανόνων. Το επόμενο βήμα είναι η δημιουργία του συντακτικού δέντρου, δηλαδή η διατήρηση μόνο των χρήσιμων κανόνων. Για τη διαδικασία αυτή δύο είδη μασκών έχουν δημιουργηθεί για κάθε συντακτικό κανόνα που περιέχει τουλάχιστον ένα μη τερματικό σύμβολο στο δεξίο του μέλος, δηλαδή για τους πρώτους 9 κανόνες. Οι μάσκες παράγονται αυτόματα βάσει της μεθοδολογίας του εξηγήθηκε στην παράγραφο 7.2.1 και παρουσιάζονται στον πίνακα 7.2. Οι κανόνες που εντέλει χρησιμοποιούνται για τη δημιουργία του συντακτικού δέντρου (βλέπε σχήμα 7.7) αναπαρίστανται στα διανύσματα του σχήματος 7.6 με γραμματοσειρά μεγαλύτερου μεγέθους από τους υπολοίπους. Διατηρώντας μόνο τους χρήσιμους κανόνες, το συντακτικό δέντρο αποθηκεύεται στη μορφή που φαίνε-



Σχήμα 7.4: Συντακτικός Πίνακας για συμβολοσειρά εισόδου 35*4.



Σχήμα 7.5: Αναπαράσταση σε διάνυσμα δυαδικών ψηφίων της γραμματικής G_{op}.

α/α Κανόνα	Κανόνας	Αριστερή Μάσκα	Δεξιά Μάσκα
0	S→E	E 0000000000000000110	- -
1	E→T+E	T 0000000000000011000	E 0000000000000000110
2	E→T	T 0000000000000011000	- -
3	T→F*T	F 0000000000001100000	T 0000000000000011000
4	T→F	F 0000000000001100000	- -
5	F →(E)	E 0000000000000000110	- -
6	F →N	N 0000000000110000000	- -
7	N →DN	D 1111111111000000000	N 0000000000110000000
8	N →D	D 1111111111000000000	- -

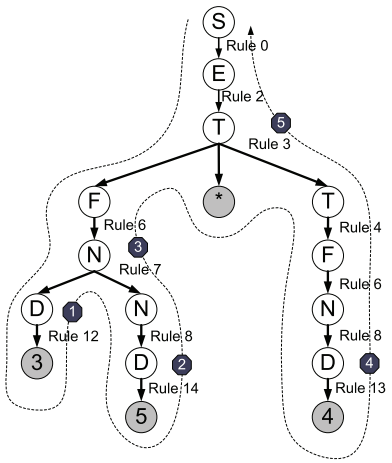
Πίνακας 7.2: Μάσκες για το διαχωρισμό των χρήσιμων κανόνων της γραμματικής G_{op}.

3	5	*	4
000000**1**000101010101	00000000000**11**010101	0000000000000000000	000000000000000**11**0**1**
	0000**1**00000**1**01010101	0000000000000000000	0000000000000001101
		0000000000000000000	0000000000000000000
			00000**1**0000**1**0**1**0**1**0101

Σχήμα 7.6: Συντακτικός Πίνακας για συμβολοσειρά εισόδου 35*4 σε αναπαράσταση διανυσμάτων δυαδικών ψηφίων.

ται στον πίνακα 7.3(a). Η πρώτη στήλη περιέχει το δείκτη της θέσης του πίνακα, η δεύτερη τους κανόνες και η τρίτη το υποδέντρο των απογόνων (με χρήση των δεικτών της πρώτης στήλης). Για παράδειγμα, οι απόγονοι του κανόνα 3 στο δέντρο του σχήματος 7.7 είναι αποθηκευμένοι στις θέσεις 1 (κανόνες 6 και 7) και 2 (κανόνες 4, 6, 8 και 13), για τον αριστερό και δεξιό κλάδο αντίστοιχα.

Για το τελικό στάδιο, εκείνο του υπολογισμού των κατηγορημάτων, το δέντρο διασχίζεται και για κάθε κανόνα η αντίστοιχη εισαγωγή/εξαγωγή στη στοίβα εκτελείται (βλέπε πίνακα 7.3(b)). Το δέντρο διατρέχεται από πάνω προς τα κάτω και από αριστερά προς τα δεξιά, όπως φαίνεται στο σχήμα 7.7. Η εκτέλεση των ενεργειών καθώς και τα περιεχόμενα της στοίβας φαίνονται στον πίνακα 7.3(b). Τέλος, σημειώνεται ότι τα αριθμημένα οκτάγωνα του σχήματος 7.7 αναφέρονται στους κόμβους του πίνακα 7.3(b). Πράγματι, στο τέλος της διαδικασία στην κορυφή της στοίβας παράγεται το αποτέλεσμα, δηλαδή η τιμή 140.



Σχήμα 7.7: Συντακτικό δέντρο για τη συμβολοσειρά εισόδου 35*4.

(a)

Δείκτης	Κανόνες	Απόγονοι
0	0,2,3	1,2
1	6,7	3,4
2	4,6,8,13	-
3	12	-
4	8,14	-

(b)

Κόμβος	Ενέργεια	Στοίβα
1	push 3;	3
2	push 5;	3, 5
3	pop x; pop y; push 10y+x	35
4	push 4;	35, 4
5	pop x; pop y; push x*y ; pop result;	140 -

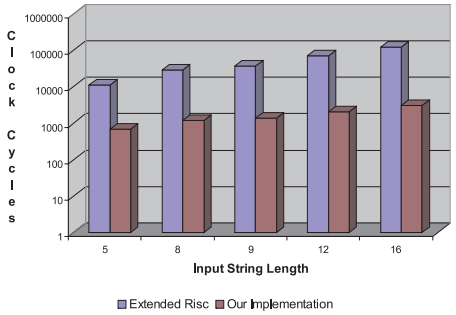
Πίνακας 7.3: a) Μορφή αποθήκευσης συντακτικού δέντρου, b) Εκτέλεση ενεργειών.

7.3 Πειραματικά Αποτελέσματα

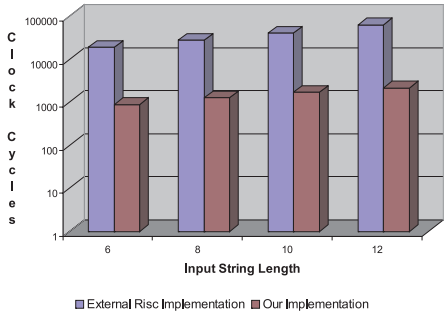
Προκειμένου να αξιολογηθεί η απόδοση της προτεινόμενης αρχιτεκτονικής, τέσσερα συστήματα αποτίμησης κατηγορημάτων δημιουργήθηκαν με τη βοήθεια του αυτοματοποιημένου εργαλείου. Τα τρία πρώτα χρησιμοποιήθηκαν για τη σύγκριση με προηγούμενες υλοποιήσεις σε υλικό ενώ το τέταρτο για σύγκριση με μια υλοποίηση σε λογισμικό. Όλες οι υλοποιήσεις έγιναν στο ίδιο FPGA, το Xilinx Virtex-5 ML506. Για κάθε μια υλοποίηση, ελήφθησαν μετρήσεις για το πλήθος των κύκλων ρολογιού που χρειάστηκαν για συμβολοσειρές εισόδου ποικίλου μεγέθους.
Το συγκεκριμένο είδος μετρήσεων (κύκλοι ρολογιού) προτιμήθηκε προκειμένου να συγκριθούν οι αρχιτεκτονικές των προσεγγίσεων, ανεξαρτήτως της χρησιμοποιούμενης τεχνολογίας. Ειδικά για την υλοποίηση σε λογισμικό, ελήφθη και ένας ακόμη τύπος μετρήσεων, ο πραγματικός χρόνος εκτέλεσης.

7.3.1 Σύγκριση με Υλοποιήσεις σε Υλικό

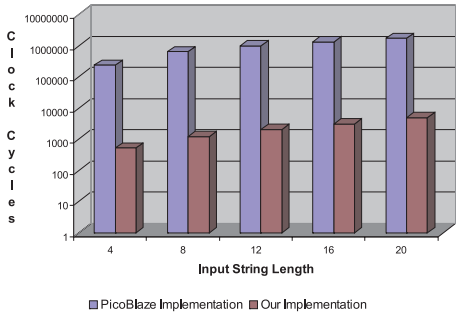
Το 2005 οι Panagopoulos et al [36] παρουσίασαν ένα επεκταμένο μικροεπεξεργαστή RISC για την αποτίμηση κατηγορημάτων, ο οποίος είχε υλοποιηθεί σε FPGA. Αποτελεί την πρώτη προσπάθεια για σχεδίαση ενός εξειδικευμένου μικροεπεξεργαστή για το



(a) Σύγκριση με την προσέγγιση του επεκταμένου επεξεργαστή Risc



(b) Σύγκριση με την προσέγγιση του εξωτερικού επεξεργαστή Risc



(c) Σύγκριση με την προσέγγιση των μικροελεκτών PicoBlaze

Σχήμα 7.8: Σύγκριση της προτεινόμενης υλοποίησης με άλλες προσεγγίσεις σε υλικό.

σκοπό της αποτίμησης κατηγορημάτων και έκανε χρήση του αλγορίθμου συντακτικής αναγνώρισης του Floyd [37]. Η αποτίμηση των επιδόσεων στο [36] βασίστηκε σε μια κατηγορική γραμματική που περιγράφει το λογικό πρόγραμμα/βάση γνώσης (Logic Program/Knowledge Base) για το παράδειγμα του "Απογόνου" ("Successor"), κάνοντας χρήση συντιθέμενων και κληρονομούμενων κατηγορημάτων. Η ίδια γραμματική χρησιμοποιήθηκε και για την αυτόματη παραγωγή του προτεινόμενου συστήματος, οδηγώντας όμως σε καλύτερες επιδόσεις, δύο τάξεων μεγέθους όπως φαίνεται και στο σχήμα 7.8(a).

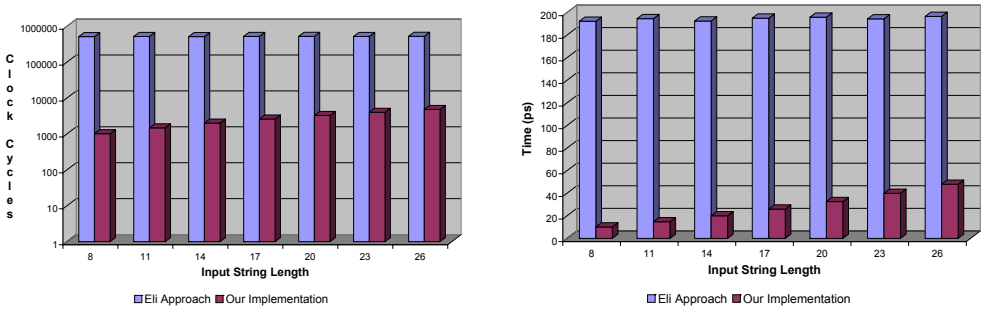
Επίσης το 2005, παρουσιάστηκε η διαφορετική προσέγγιση που αναλύθηκε στο κεφάλαιο 3 και έκανε χρήση του αλγορίθμου του Earley υλοποιημένου σε FPGA [28] και ενός εξωτερικού μικροεπεξεργαστή RISC υπεύθυνου για την αποτίμηση των κατηγορημάτων. Σε εκείνη την προσέγγιση, η αποτίμηση των επιδόσεων βασίστηκε στο γνωστό παράδειγμα του 'Wumpus" World game [76], μετασχηματισμένο στην αντίστοιχη S-Attributed κατηγορική γραμματική. Η απόδοση της προτεινόμενης υλοποίησης για τη συγκεκριμένη γραμματική είναι συγκριτικά επίσης μια με δύο τάξεις μεγέθους καλύτερη, όπως φαίνεται και στο σχήμα 7.8(b).

Το 2006 στην εργασία [47] η ίδια υλοποίηση του συντακτικού αναλυτή [28] χρησιμοποιήθηκε για τη δημιουργία του αποτιμητή που παρουσιάστηκε στην ενότητα 4.1.1,

στον οποίο η σημασιολογική ανάλυση γινόταν σε δύο soft core μικροελεγκτές (PicoBlaze) ενσωματωμένους στο ίδιο FPGA με τον συντακτικό αναλυτή. Η αρχιτεκτονική αυτή αύξησε μεν την μεταφερσιμότητα εν συγκρίσει με εκείνη του κεφαλαίου 3 αλλά εξαιτίας της χρήσης των μικροελεγκτών η απόδοση ελαττώθηκε. Η κατηγορική γραμματική που χρησιμοποιήθηκε για την αξιολόγηση της απόδοσης περιέγραφε το λογικό πρόγραμμα αναζήτησης συνδέσεων σε ένα διατεταγμένο ακυκλικό γράφο, με χρήση συντιθέμενων και κληρονομούμενων κατηγορημάτων. Η απόδοση της προτεινόμενης αρχιτεκτονικής για τη συγκεκριμένη γραμματική είναι δύο με τρεις τάξεις μεγέθους καλύτερη, όπως φαίνεται και στο σχήμα 7.8(c).

7.3.2 Σύγκριση με Υλοποιήσεις σε Λογισμικό

Αναφορικά με τις υλοποιήσεις σε λογισμικό, ένας συνδυασμός εργαλείων [32] που υλοποιούν διάφορες στρατηγικές κατασκευής μεταγλωττιστών σε ένα συγκεκριμένο προγραμματιστικό περιβάλλον είναι το Eli [31]. Για την αποτίμηση των κατηγορημάτων, το Eli κάνει χρήση της LIDO [33], μιας γλώσσας για τον προσδιορισμό των υπολογισμών σε συντακτικά δέντρα, που υποστηρίζει τόσο συντιθέμενα όσο και κληρονομούμενα κατηγορήματα. Η κατηγορική γραμματική που χρησιμοποιήθηκε για την αξιολόγηση της επίδοσης ήταν η G_{op}, που έχει παρουσιαστεί τον πίνακα 7.1. Ελήφθησαν μετρήσεις για τους κύκλους ρολογιού που χρειάστηκαν για συμβολοσειρές εισόδου διαφορετικού μεγέθους. Η προτεινόμενη προσέγγιση επιτυγχάνει επιτάχυνση δύο με τρεις τάξεις μεγέθους, συγκριτικά με τη υλοποίηση στο λογισμικό, όπως παρουσιάζεται και στο λογαριθμικό γράφημα του σχήματος 7.9(a).
Επιπλέον μετρήσεις ελήφθησαν για το χρόνο εκτέλεσης, προκειμένου να παρουσιασθεί μια ακόμη πιο απτή ένδειξη της επιτάχυνσης. Λόγω της χαμηλής συχνότητας του ρολογιού του FPGA (100 MHz), ο απαιτούμενος χρόνος θα μπορούσε να είναι συγκρίσιμος με εκείνον που χρειάζεται η υλοποίηση σε λογισμικό που εκτελείται σε ένα πολύ ταχύτερο επεξεργαστή (Pentium @ 2.6 GHz). Παρόλα αυτά, αν και η συ-



(a) Κύκλοι Ρολογιού (b) Χρόνος

Σχήμα 7.9: Σύγκριση της προτεινόμενης υλοποίησης με την προσέγγιση σε λογισμικό του Eli.

χνότητα του επεξεργαστή είναι μια τάξη μεγέθους μεγαλύτερη από εκείνη του FPGA, ο χρόνος εκτέλεσης της προτεινόμενης προσέγγισης είναι, για λογικά μήκη συμβολοσειρών εισόδου, μια τάξη μεγέθους μικρότερος από τον αντίστοιχο του Pentium. Οι συγκεκριμένες μετρήσεις παρουσιάζονται στο σχήμα 7.9(b).

Βάσει των παραπάνω σχημάτων, μπορεί να προβλεφθεί ότι η επιτάχυνση της προτεινόμενης προσέγγισης ελαττώνεται όσο αυξάνει το μήκος της συμβολοσειράς εισόδου. Όμως πρέπει να τονισθεί ότι ο παραγόμενος από το Eli συντακτικός αναλυτής είναι ντετερμινιστικός και συνεπώς ταχύτατος αλλά εν αντιθέσει με την προτεινόμενη προσέγγιση δεν παράγει όλες τις δυνατές λύσεις για την περίπτωση διφορούμενων κατηγορικών γραμματικών. Εξαιτίας λοιπόν της ντετερμινιστικής φύσης του αναλυτή, η αύξηση του μήκους της συμβολοσειράς εισόδου δεν οδηγεί σε ανάλογη αύξηση του χρόνου εκτέλεσης. Ασφαλώς, όπως έχει ήδη εξηγηθεί, στόχος είναι οι προτεινόμενες υλοποιήσεις να βρίσκουν εφαρμογή και στο πεδίο της λογικής και της τεχνητής νοημοσύνης, επομένως για αυτό το λόγο προτιμήθηκε ο μη ντετερμινιστικός συντακτικός αναλυτής. Επιπλέον, κατά την διαδικασία αναγνώρισης μεγάλων συμβολοσειρών εισόδου, οι συμβολοσειρές χωρίζονται κατά βάση σε μικρότερου μεγέθους συμβολοσειρές. Ένα τυπικό παράδειγμα αποτελεί η αναγνώριση του ηλεκτροκαρδιογραφήματος (ECG), όπου διαδοχικοί καρδιακοί κύκλοι πρέπει να αναλυθούν σε πραγματικό χρόνο. Όλες οι μετρήσεις ελήφθησαν για συμβολοσειρές εισόδου με μέγιστο μέγεθος 26 και επιτυγχάνουν μια μέση ελάττωση του χρόνου εκτέλεσης κατά μια τάξη μεγέθους. Για μεγαλύτερες γραμματικές η επιτάχυνση ενισχύεται περαιτέρω. Είναι σημαντικό να τονιστεί αν και ο χρόνος που εξοικονομείται μπορεί να θεωρηθεί μικρής σημασίας (από 200 psec σε 20 psec), σε πραγματικής εφαρμογές όπου μεγάλα σύνολα συμβολοσειρών εισόδου πρέπει να αναγνωριστούν σε πραγματικό χρόνο, η εξοικονόμηση αυτή συσσωρεύεται σε μια σημαντική μείωση του χρόνου εκτέλεσης.

7.4 Διεπαφή Φυσικών Γλωσσών σε Βάση Δεδομένων

Ένα αναλυτικό παράδειγμα δίνεται ακολούθως από το πεδίο της επεξεργασίας Φυσικών Γλωσσών (Natural Language). Ένα σύστημα που επιτρέπει σε χρήστες να έχουν πρόσβαση σε πληροφορίες αποθηκευμένες σε μια βάση δεδομένων μέσω ερωτημάτων εκφρασμένων σε φυσική γλώσσα, ονομάζεται διεπαφή φυσικής γλώσσας σε βάση δεδομένων (natural language interface to a database - N_{LIDB}) [77]. Η ιδέα για ένα τέτοιο σύστημα μοιάζει γοητευτική αφού απλοποιεί τον τρόπο επικοινωνίας για τον τελικό χρήστη αλλά το επεξεργαστικό κόστος μπορεί να είναι πολύ μεγάλο, ιδιαίτερα στην περίπτωση πολλών ταυτόχρονων χρηστών που υποβάλλουν ερωτήματα. Υπάρχουν προσεγγίσεις σε λογισμικό [78] οι οποίες είναι δημοφιλείς και εμπορικά χρησιμοποιούμενες σε εφαρμογές. Στη συγκεκριμένη ενότητα, το προτεινόμενο σύστημα δημιουργεί μια διεπαφή σε υλικό ικανή να μετατρέπει ερωτήματα αναφορικά με

αεροπορικές πτήσεις από γλώσσα που μοιάζει με Αγγλικά σε ερωτήματα SQL [74]. Το σύστημα μπορεί να δεχτεί προτάσεις (ερωτήματα) που ανήκουν σε ένα υποσύνολο της Αγγλικής γλώσσας μέσω της διεπαφής δικτύου (NIC). Μόλις ολοκληρωθεί η συντακτική αναγνώριση του ερωτήματος, χρησιμοποιώντας το κατασκευασθέν συντακτικό δέντρο, οι τιμές των κατηγορημάτων υπολογίζονται και το FPGA παράγει και αποστέλλει ερωτήματα SQL σε μια βάση δεδομένων προκειμένου να παραχθεί το τελικό αποτέλεσμα (απάντηση) ή το εκτελεί απευθείας σε ένα ενσωματωμένο σύστημα (π.χ. κινητό τηλέφωνο). Η υλοποίηση σε υλικό επιτρέπει στο σύστημα να χρησιμοποιηθεί σε εφαρμογές όπου πολλές προτάσεις εισόδου πρέπει να επεξεργαστούν ταυτόχρονα και να εξαχθούν πληροφορίες.

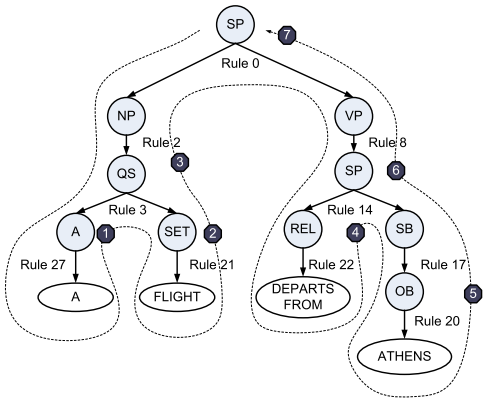
Στη δεύτερη στήλη του πίνακα 7.4 φαίνονται οι συντακτικοί κανόνες της κατηγορικής γραμματικής G_{NL} [74] που αναλύθηκε και στο κεφάλαιο 6, η οποία αποτελείται από 29 κανόνες και μπορεί να δεχθεί ένα υποσύνολο των Αγγλικών. Υπενθυμίζεται ότι στη συγκεκριμένη γραμματική, η σημασιολογία περιγράφεται με τη χρήση ενός μόνο συντιθέμενου κατηγορήματος για κάθε μη τερματικό σύμβολο. Οι προτάσεις είναι ερωτήσεις αναφορικά με αεροπορικές πτήσεις και στο κατηγόρημα αποθηκεύεται ενδιάμεσος κώδικας προς εκτέλεση από μια αφηρημένη μηχανή διαχείρισης πληροφοριών. Η μόνη πράξη που χρειάζεται ανάμεσα στα κατηγορήματα των μη τερματικών συμβόλων είναι η conc (par_1, ..., par_n), που εκτελεί την συνένωση των περιεχομένων των par_1, ..., par_n. Μια απλή ερώτηση είναι :"A flight departs from Athens?"

Με χρήση της προτεινόμενης μεθοδολογίας, μια νέα κατηγορική γραμματική G_{SQL} προκύπτει από την G_{NL}, στην οποία οι συντακτικοί κανόνες είναι πανομοιότυποι. Μόνο ένα συντιθέμενο κατηγόρημα υπολογίζεται αλλά τώρα το αποτέλεσμα είναι ένα SQL ερώτημα το οποίο μπορεί να επεξεργαστεί οποιοσδήποτε εξυπηρετητής SQL. Οι σημασιολογικοί κανόνες έχουν αντιστοιχιθεί σε απλές ενέργειες εισαγωγής/εξαγωγής δεδομένων σε μια στοίβα. Οι αντίστοιχες αυτές ενέργειες παρουσιάζονται στην τελευταία στήλη του πίνακα 7.4.

Ο τρόπος με τον οποίο μετασχηματίζεται η παραπάνω ερώτηση σε ερώτημα SQL παρουσιάζεται στο σχήμα 7.10 και στον πίνακα 7.4. Στο σχήμα 7.10 το αντίστοιχο συντακτικό δέντρο δίνεται, το οποίο διασχίζεται και για κάθε κανόνα η αντίστοιχη ενέργεια εισαγωγής/εξαγωγής δεδομένων στη στοίβα εκτελείται. Το δέντρο διασχίζεται από πάνω προς τα κάτω και από αριστερά προς τα δεξιά, όπως υποδεικνύεται από τη διακεκομμένη γραμμή. Η εκτέλεση των ενεργειών καθώς και τα περιεχόμενα της στοίβας παρουσιάζονται στον πίνακα 7.4. Πρέπει να σημειωθεί ότι τα αριθμημένα οκτάγωνα του σχήματος 7.10 αναφέρονται στους κόμβους του πίνακα 7.4. Στο τέλος της διαδικασίας εκτέλεσης των ενεργειών, στην κορυφή της στοίβας βρίσκεται το ερώτημα "Select FlightNo From Flights Where DepartsFrom= "ATH";". Δεδομένου ότι η βάση δεδομένων αποτελείται από σχεσιακό μοντέλο Flights: Flights(FlightNo id, DepartsFrom char[3], ArrivesAt char[3], Airline char[5], ...), το προτεινόμενο

Αριθμός Κανόνα	**Συντακτικός Κανόνας**	**Αντίστοιχη Ενέργεια**
0	PR → NP VP	pop x; pop y; push conc(y,x,";")
1	NP → QS W VP	pop x; pop y; push conc("select",y,"where",x,"AND")
2	NP → QS	pop x;push conc("select",x,"where")
3	QS → A SET	pop x; pop y; push conc(y,"from",x)
4	QS → E SET	pop x; pop y; push conc("*","from",x)
5	VP → VP1 VP2 SP	pop x; pop y; pop z; pop w; push con(z,y,x,w)
6	VP → VP2 SP	pop x; pop y; push conc(y,x);
7	VP → VP1 SP	pop x; pop y; pop z; push conc(y,x,z)
8	VP → SP	-
9	VP1 → IP VP1	pop x; pop y; pop z; pop w; push conc(w,z); push conc(y,x)
10	VP1 → IP	-
11	VP2 → SP CNJ VP2	pop x; pop y; pop z; push conc(z,y,x)
12	VP2 → SP CNJ	pop x; pop y; push conc(y,x)
13	IP → REL QF W	pop x; pop y; push ")"; push conc(y,"IN (select ",x,"where")
14	SP → REL SB	pop x; pop y; push conc(y,"=",x)
15	SP → AT NRL N	pop x; pop y; pop z; push conc(z,y,x,")")
16	SB → QF	-
17	SB → OB	-
18	QF → A SET	pop x; pop y; push conc(y,"from",x)
19	QF → E SET	pop x; pop y; push conc("*","from",x)
20	OB → Object names e.g. : Athens	push "Athens"
21	SET → Class Names e.g.: Flights City	push "FlightNo"; push "Flights" push "CityName"; push "Cities"
22	REL → Relation Phrases e.g.: Departs from	push "DepartsFrom"
23	AT → Property Names e.g.: Has population	push "Population"
24	NRL → Numerical Relations e.g.: Larger than	push ">"
25	N → Numbers e.g. 55	push "55"
26	W → Relative pronouns e.g.: which	-
27	A → Determiners e.g.: A	-
28	E → Determiners e.g.: Each	-
29	CNJ → Conjunction words or symbols e.g.: And	push "AND"

Πίνακας 7.4: Η κατηγορική γραμματική G_{SQL}.



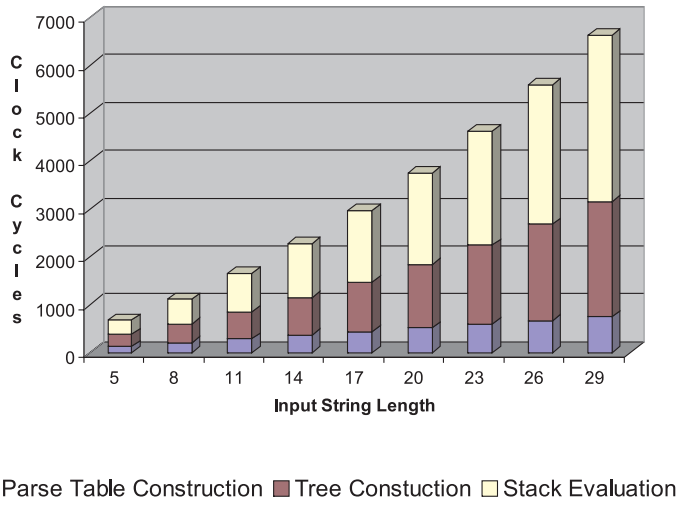
Σχήμα 7.10: Αντίστοιχο Συντακτικό Δέντρο για τη συμβολοσειρά εισόδου "A flight departs from Athens".

σύστημα παράγει ένα ερώτημα SQL που μπορεί να απαντηθεί από οποιοδήποτε εξυπηρετητή SQL.

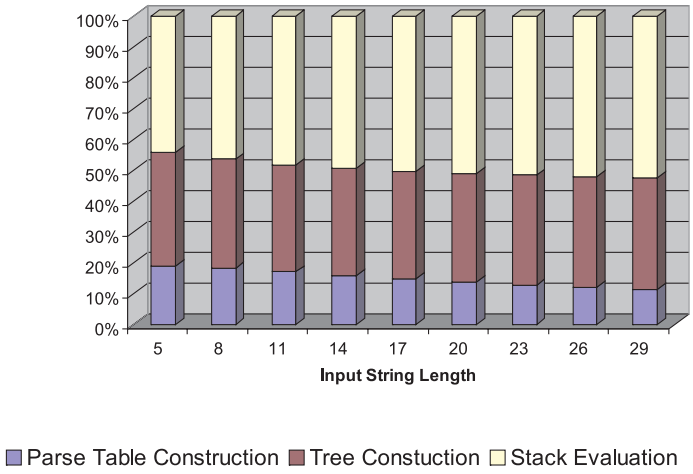
Το προτεινόμενο σύστημα δοκιμάστηκε με ποικίλα μήκη συμβολοσειρών εισόδου για πιο σύνθετες ερωτήσεις όπως "TWA flies from Athens to New York", "Each flight which is connected to a flight which belongs to Swissair departs from a city which is linked to each city which belongs to France" κτλ. Μάλιστα, στην επόμενη παράγραφο δίνεται η ανάλυση για τη σύνθετη ερώτηση "Each flight which is connected to a flight which belongs to TWA departs from a city which is linked to each city which belongs to Greece". Η απόδοση του συστήματος μετρήθηκε σε κύκλους ρολογιού και παρουσιάζεται στο σχήμα 7.11(a). Επιπλέον, στο σχήμα 7.11(b) παρουσιάζεται η

Κόμβος	**Ενέργεια**	**Περιεχόμενα Στοίβας**
1	push "select FlightNo"	"select FlightNo"
2	push "from Flights"	"select FlightNo", "from Flights"
3	pop x; pop y; push conc(y,x)	"select FlightNo from Flights"
4	push "where DepartsFrom="	"select FlightNo from Flights", "where DepartsFrom="
5	push ""ATH""	"select FlightNo from Flights", "where DepartsFrom=", ""ATH""
6	pop x; pop y; push conc(y,x)	"select FlightNo from Flights", "where DepartsFrom="ATH""
7	pop x; pop y; push conc(y,x,;)	"select FlightNo from Flights where DepartsFrom="ATH";"

Πίνακας 7.5: Αποτέλεσμα Ενεργειών των κανόνων της G_{SQL} στη στοίβα.



(a)



(b)

Σχήμα 7.11: a) Απαραίτητοι κύκλοι ρολογιού για τη G_{SQL} για διαφορετικά μήκη συμβολοσειρών εισόδου b) Ποσοστιαία κατανάλωση του συνολικού χρόνου εκτέλεσης από κάθε ένα από τα τρία λειτουργικά τμήματα.

ποσοστιαία κατανάλωση του συνολικού χρόνου εκτέλεσης από κάθε ένα από τα τρία λειτουργικά τμήματα για διάφορα μεγέθη συμβολοσειρών εισόδου. Όπως φαίνεται στο συγκεκριμένο σχήμα, το πιο γρήγορο τμήμα είναι εκείνο της συντακτικής ανάλυσης και κατασκευής του συντακτικού πίνακα, όπως αναμενόταν άλλωστε λόγω της συνδυαστική λογικής της υλοποίησης. Εντούτοις, λόγω της προτεινόμενης μεθοδολογίας, η απόδοση και των υπολοίπων δύο τμημάτων οδηγεί σε μια συνολική αποδοτική επίδοση.

7.4.1 Ένα σύνθετο SQL ερώτημα

Στο σχήμα 7.12 παρουσιάζεται το αντίστοιχο συντακτικό δέντρο για τη σύνθετη ερώτηση "Each flight which is connected to a flight which belongs to TWA departs from a city which is linked to each city which belongs to Greece". Η συγκεκριμένη ερώτηση οδηγεί στην κατασκευή του SQL ερωτήματος "select * from Flights where ConnectedTo IN (select FlightNo from Flights where BelongTo=TWA) AND DepartFrom IN (select CityName from Cities where LinkedTo IN (select * from 'Cities where BelongsTo=Greece))". Όπως και στην ενότητα 7.4, τα αριθμημένα οκτάγωνα του σχήματος αντιστοιχούν στου κόμβους του πίνακα 7.6, στον οποίο παρουσιάζονται οι αντίστοιχες ενέργειες καθώς και τα περιεχόμενα της στοίβας.

Πίνακας 7.6: Εκτέλεση Ενεργειών και περιεχόμενα της στοίβας για το συντακτικό δέντρο του σχήματος 7.12.

Κόμβος	Ενέργεια	Περιεχόμενα Στοίβας
1	push "FlightNo";push "Flights"	"FlightNo" "Flights"
2	pop x;pop y; push ("*","from",x)	"* from Flights"
3	push "ConnectedTo"	"ConnectedTo" "* from Flights"
4	push "FlightNo";push "Flights"	"Flights" "FlightNo" "ConnectedTo" "* from Flights"
5	pop x;pop y; push (y,"from",x)	"FlightNo from Flights" "ConnectedTo" "* from Flights"
6	pop x; pop y; push ")";push conc(y,"IN (select ",x,"where")	"ConnectedTo IN (select FlightNo from Flights where" ")" "* from Flights"
7	push "BelongTo"	"BelongTo" "ConnectedTo IN (select FlightNo from Flights where" ")" "* from Flights"
8	push "TWA"	"TWA" "BelongTo" "ConnectedTo IN (select FlightNo from Flights where" ")" "* from Flights"
9	pop x; pop y; push conc(y,"=",x)	"BelongTo=TWA" "ConnectedTo IN (select FlightNo from Flights where" ")" "* from Flights"
10	pop x; pop y; pop z; push conc(y,x,z)	"ConnectedTo IN (select FlightNo from Flights where BelongTo=TWA) "* from Flights"
11	pop x; pop y; push conc("select",y,"where",x,"AND")	"select * from Flights where ConnectedTo IN (select FlightNo from Flights where BelongTo=TWA) AND"
12	push "DepartFrom"	"DepartFrom" "select * from Flights where ConnectedTo IN (select FlightNo from Flights where BelongTo=TWA) AND"
13	push "CityName";push "Cities"	"Cities" "CityName"

Συνεχίζεται στην επόμενη σελίδα

Πίνακας 7.6 – συνέχεια από την προηγούμενη σελίδα

Κόμβος	Ενέργεια	Περιεχόμενα Στοίβας
		"DepartFrom" "select * from Flights where ConnectedTo IN (select FlightNo from Flights where BelongTo=TWA) AND"
14	pop x; pop y; push conc(y,"from",x)"	"CityName from Cities" "DepartFrom" "select * from Flights where ConnectedTo IN (select FlightNo from Flights where BelongTo=TWA) AND"
15	pop x; pop y; push ")"; push conc(y,"IN (select ",x,"where")	"DepartFrom IN (select CityName from Cities where" ")" "select * from Flights where ConnectedTo IN (select FlightNo from Flights where BelongTo=TWA) AND"
16	push "LinkedTo"	"LinkedTo" "DepartFrom IN (select CityName from Cities where" ")" "select * from Flights where ConnectedTo IN (select FlightNo from Flights where BelongTo=TWA) AND"
17	push "CityName";push "Cities"	"Cities" "CityName" "LinkedTo" "DepartFrom IN (select CityName from Cities where" ")" "select * from Flights where ConnectedTo IN (select FlightNo from Flights where BelongTo=TWA) AND"
18	pop x;pop y; push ("*","from",x)	"* from 'Cities" "LinkedTo" "DepartFrom IN (select CityName from Cities where" ")" "select * from Flights where ConnectedTo IN (select FlightNo from Flights where BelongTo=TWA) AND"
19	pop x; pop y; push ")"; push conc(y,"IN (select ",x,"where")	" LinkedTo IN (select * from 'Cities where" "DepartFrom IN (select CityName from Cities where" ")" ")" "select * from Flights where ConnectedTo IN (select FlightNo from Flights where BelongTo=TWA) AND"
20	pop x; pop y; pop z; pop w; push conc(w,z); push conc(y,x,")")	"DepartFrom IN (select CityName from Cities where LinkedTo IN (select * from 'Cities where" "))" "select * from Flights where ConnectedTo IN (select FlightNo from Flights where BelongTo=TWA) AND"
21	push "BelongsTo"	"BelongsTo" "DepartFrom IN (select CityName from Cities where LinkedTo IN (select * from 'Cities where" "))" "select * from Flights where ConnectedTo IN (select FlightNo

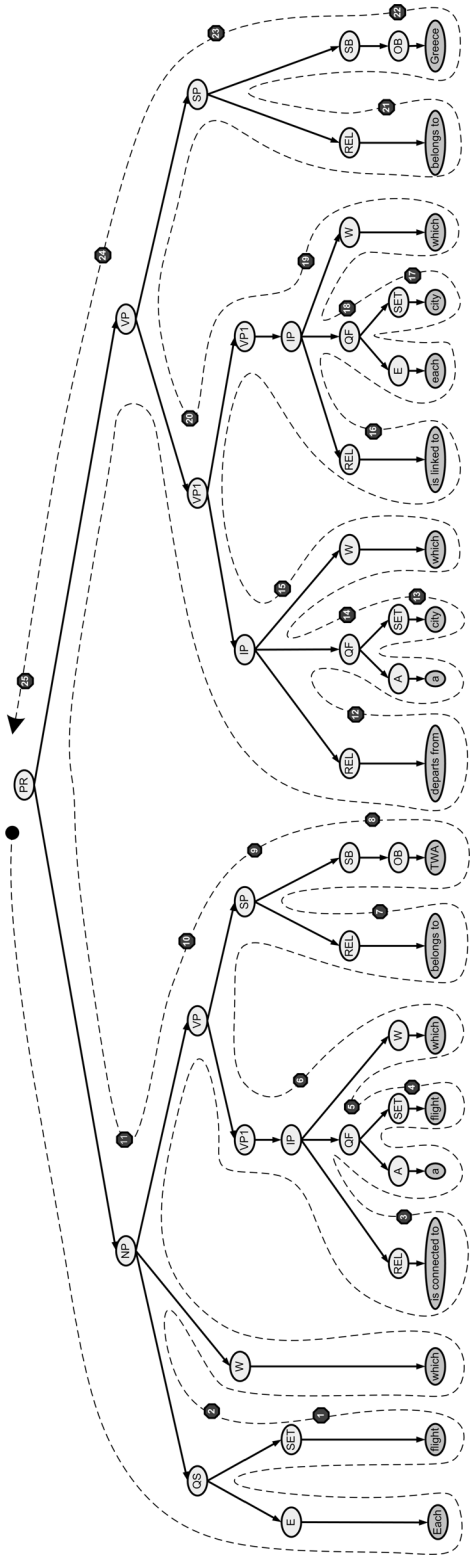
Συνεχίζεται στην επόμενη σελίδα

Πίνακας 7.6 – συνέχεια από την προηγούμενη σελίδα

Κόμβος	Ενέργεια	Περιεχόμενα Στοίβας
		from Flights where BelongTo=TWA) AND"
22	push "Greece"	"Greece" "BelongsTo" "DepartFrom IN (select CityName from Cities where LinkedTo IN (select * from 'Cities where" "))" "select * from Flights where ConnectedTo IN (select FlightNo from Flights where BelongTo=TWA) AND"
23	pop x; pop y; push conc(y,"=",x)	"BelongsTo=Greece" "DepartFrom IN (select CityName from Cities where LinkedTo IN (select * from 'Cities where" "))" "select * from Flights where ConnectedTo IN (select FlightNo from Flights where BelongTo=TWA) AND"
24	pop x; pop y; pop z;push conc(y,x,z)	"DepartFrom IN (select CityName from Cities where LinkedTo IN (select * from 'Cities where BelongsTo=Greece))" "select * from Flights where ConnectedTo IN (select FlightNo from Flights where BelongTo=TWA) AND"
25	pop x; pop y; push conc(y,x,";")	"select * from Flights where ConnectedTo IN (select FlightNo from Flights where BelongTo=TWA) AND DepartFrom IN (select CityName from Cities where LinkedTo IN (select * from 'Cities where BelongsTo=Greece))"

7.5 Εφαρμογές και Μελλοντικές Επεκτάσεις

Στο κεφάλαιο αυτό παρουσιάστηκε μια αρχιτεκτονική για αποτιμητές κατηγορικών γραμματικών, η οποία στηρίζεται σε καθαρά συνδυαστική λογική για τη συντακτική αναγνώριση ενώ για το σημασιολογικό κομμάτι κάνει χρήση μιας μεθοδολογίας βασισμένης στη χρήση στοιβών. Όπως έχει παρουσιαστεί νωρίτερα και φαίνεται και στο σχήμα 7.1, το σύστημα αποτελείται από τρία βασικά λειτουργικά τμήματα. Μια μελλοντική κατεύθυνση, στην οποία υπάρχει ήδη πρόοδος, είναι η εφαρμογή τεχνικών pipeline μεταξύ των τριών αυτών τμημάτων, που θα οδηγήσει σε δραστική ελάττωση του συνολικού χρόνου εκτέλεσης, ιδίως όταν μεγάλες συμβολοσείρες εισόδου προς αναγνώριση χωρίζονται σε μικρότερες και αναγνωρίζονται διαδοχικά. Μια μελλοντική πτυχή είναι ο συνδυασμός με άλλα ενσωματωμένα συστήματα που επιταχύνουν την εκτέλεση εφαρμογών Java, όπως το JOP [79], προκειμένου να παραχθεί ένα σύστημα πλήρως σε υλικό, το οποίο θα λαμβάνει τον πηγαίο κώδικα Java, θα τον μετατρέπει σε Java Bytecode και θα εκτελεί τον τελευταίο σε εικονική μηχανή Java (Java



Σχήμα 7.12: Αντίστοιχο Συντακτικό Δέντρο για τη σύνθετη ερώτηση.

virtual Machine) σε υλικό (JOP). Τέλος, δεδομένη της στενής σχέσης μεταξύ κατηγορικών γραμματικών και λογικών προγραμμάτων [10], το προτεινόμενο σύστημα μπορεί να επεκταθεί εύκολα ώστε να βρει εφαρμογή σε ευφυή ενσωματωμένα συστή-

ματα με εφαρμογές λογικού προγραμματισμού καθώς και να προστεθεί η υποστήριξη αβεβαιότητας (uncertainty) και ασάφειας (fuzziness). Το τελευταίο έχει μάλιστα ήδη αρχίσει να εξετάζεται και κάποια αποτελέσματα έχουν ήδη προκύψει, όπως θα φανεί στην επόμενη παράγραφο.

7.5.1 Το Μοντέλο Συντελεστών Βεβαιότητας

Η αντίληψη που έχει ο άνθρωπος για την γύρω του πραγματικότητα βασίζεται κυρίως σε ανακριβή γνώση. Παρά την ανακρίβεια, η γνώση αυτή είναι πολύ χρήσιμη για τον ίδιο προκειμένου να λύσει απλά αλλά και σύνθετα προβλήματα. Προφανώς, προβλήματα που απαιτούν μεγάλη ακρίβεια για να λυθούν όπως η στόχευση από απόσταση ενός τηλεσκοπίου ή εστίαση ενός ηλεκτρονικού μικροσκοπίου δεν εμπίπτουν σε αυτή την κατηγορία. Όμως, λίγα είναι τα προβλήματα που απαιτούν τόση ακρίβεια αφού καθημερινά προβλήματα όπως η οδήγηση, το παρκάρισμα, η επιλογή γεύματος σε ένα εστιατόριο δεν απαιτούν τόσο μεγάλη ακρίβεια.
Μερικά από τα πιο γνωστά μοντέλα ανακρίβειας (inexact models) είναι το πιθανοτητικό (probabilistic) μοντέλο, το μοντέλο της βεβαιότητας (certainty), το μοντέλο δυνατότητας-αναγκαιότητας (possibility-necessity) και το ασαφές (fuzzy) μοντέλο. Στα ακόλουθα θα γίνει αναφορά στο μοντέλο της βεβαιότητας, αλλά μπορούν εύκολα να βρουν εφαρμογή και στα υπόλοιπα μοντέλα.
Η θεωρία της βεβαιότητας αναπτύχθηκε για τη χρήση αβέβαιης γνώσης σε ιατρικά έμπειρα συστήματα (expert systems). Ενσωματώθηκε στο MYCIN [80] προκειμένου να ξεπεραστούν πολλά προβλήματα που προκύπτουν όταν χρησιμοποιείται η θεωρία πιθανοτήτων για ανακριβείς συλλογισμούς. Το μέτρο της βεβαιότητας των συμβόλων είναι οι συντελεστές βεβαιότητας. Υπάρχει ακόμη μέτρο βεβαιότητας για τους κανόνες, το οποίο ονομάζεται συντελεστής εξασθένισης (attenuation coefficients) του κανόνα. Ο συντελεστής εξασθένισης είναι ένα μέτρο βεβαιότητας της κατάληξης του κανόνα, δεδομένης της ύπαρξής του.
Σε κάθε δεδομένο (fact) X αντιστοιχεί ένας συντελεστής βεβαιότητας C_X, που λαμβάνει τιμές από -1 μέχρι 1 με την ακόλουθη λογική:

- $C_X = +1$ εάν το X είναι γνωστό ότι είναι αληθές.
- $C_X =$ -1 εάν το X είναι γνωστό ότι είναι ψευδές.
- $C_X = 0$ εάν τίποτε δεν είναι γνωστό για το X.

Οι τιμές μεταξύ 0 και 1 υποδεικνύουν ένα μέτρο της πίστης στο γεγονός ενώ οι τιμές μεταξύ -1 και 0 υποδεικνύουν ένα μέτρο της δυσπιστίας στο γεγονός. Σε κάθε κανόνα, έστω R, πρέπει να ανατεθεί ένας συντελεστής εξασθένισης C_R, που είναι ένας αριθμός μεταξύ 0 και 1 και αντιπροσωπεύει την αξιοπιστία του κανόνα. Έστω ο κανόνας $R : X_0$ εάν $X_1, X_2, \ldots, X_n$, με συντελεστή εξασθένισης C_R και $C_{X1}, C_{X2}, \ldots, C_{Xn}$ οι

συντελεστές βεβαιότητας για τα σύμβολα (συνθήκες) του κανόνα. Η συνδυαστική συνάρτηση (combination function) fC_M είναι το συνδυασμένο μέτρο των βεβαιοτήτων των $X_1, X_2, \ldots, X_n$:

$$fC_M(C_{X1}, C_{X2}, \ldots, C_{Xn}) = min(C_{X1}, C_{X2}, \ldots, C_{Xn}) \tag{7.1}$$

Κάθε φορά που το συμπέρασμα ενός κανόνα αποδεικνύεται από τις συνθήκες του κανόνα, πρέπει να υπολογίζεται η βεβαιότητα του συμπεράσματος:

$$C_{X0} = C_R * fC_M(C_{X1}, C_{X2}, \ldots, C_{Xn}) \tag{7.2}$$

Η συσσωρευτική συνάρτηση (cumulation function) fC_U(C$_1$,C$_2$) δίνει το αποτέλεσμα της συσσώρευσης δύο συντελεστών βεβαιότητας C$_1$, C$_2$:

- Εάν $C_1 \geq 0$ και $C_2 \geq 0$
  τότε fC_U(C$_1$,C$_2$) = C$_1$ + C$_2$ - C$_1$*C$_2$
- Εάν $C_1 < 0$ και $C_2 < 0$
  τότε fC_U(C$_1$,C$_2$) = C$_1$ + C$_2$ + C$_1$*C$_2$
- Εάν $C_1 * C_2 < 0$ τότε
  εάν $|C_1| * |C_2| = 1$ τότε fC_U(C$_1$,C$_2$) = 1 αλλιώς

$$fC_U(C_1, C_2) = \frac{C_1 + C_2}{1 - min(|C_1|, |C_2|)} \tag{7.3}$$

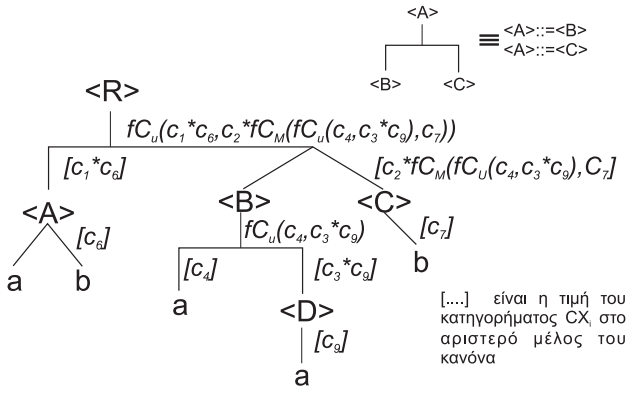
Εάν κατά τη διάρκεια της διαδικασίας απόδειξης, χρησιμοποιηθούν περισσότεροι του ενός κανόνες R$_i$ που οδηγούν στο ίδιο συμπέρασμα X, με βεβαιότητα C$_i$, τότε ο συντελεστής βεβαιότητας του συμπεράσματος C$_X$ πρέπει να είναι ένας "συσσωρευμένος συντελεστής βεβαιότητας" και υπολογίζεται ως εξής:

$$C_X \;=\; fC_U\,(C_m,\; fC_U\,(C_m - 1),\;\; \ldots,\; fC_U\,(C_3,\; fC_U\,(C_2,\; C_1)\ldots) \tag{7.4}$$

Παράδειγμα 10 *Έστω η ακόλουθη γραμματική σε μορφή BNF, εμπλουτισμένη με συντελεστές βεβαιότητας:*

$<R> ::= <A>\ ,$ $\quad cf = c_1$
$<R> ::= <B><C>,$ $\quad cf = c_2$
$<B> ::= <D>\ ,$ $\quad cf = c_3$
$<B> ::= a\ ,$ $\quad cf = c_4$
$<A> ::= a\ ,$ $\quad cf = c_5$
$<A> ::= ab\ ,$ $\quad cf = c_6$
$<C> ::= b\ ,$ $\quad cf = c_7$
$<C> ::= c\ ,$ $\quad cf = c_8$
$<D> ::= a\ ,$ $\quad cf = c_9$

A, B, C, D και R είναι μη τερματικά σύμβολα, a, b, c τερματικά σύμβολα και cf ο συντελεστής βεβαιότητας κάθε κανόνα. Η συγκεκριμένη γραμματική είναι διφορούμενη, δηλαδή μια συμβολοσειρά εισόδου μπορεί να παραχθεί με περισσότερους συνδυασμούς κανόνων και όχι μόνον ένα. Για την απλή συμβολοσειρά εισόδου "ab", υπάρχουν τρία πιθανά συντακτικά δέντρα, τα οποία είναι όλα σωστά. Στο σχήμα 7.13 δίνεται το αντίστοιχο συντακτικό δέντρο για τη συμβολοσειρά εισόδου "ab", το συντακτικό δέντρο έχει εμπλουτιστεί προκειμένου να φαίνονται οι υπολογισμοί των συντελεστών βεβαιότητας καθώς και όλες οι εναλλακτικές λύσεις για κάθε μη τερματικό σύμβολο (χρήση οριζόντιας γραμμής).



Σχήμα 7.13: Συντακτικό δέντρο με τα αντίστοιχα κατηγορήματα για τη συμβολοσειρά εισόδου "ab".

Όπως φάνηκε και από το σχήμα 7.13, για τον τελικό υπολογισμό των κατηγορημάτων είναι απαραίτητο να βρεθούν όλες οι δυνατές λύσεις και να συνδυαστούν κατάλληλα οι αντίστοιχοι συντελεστές βεβαιότητας. Το τελευταίο είναι εφικτό με την προτεινόμενη υλοποίηση του παραλλήλου αλγορίθμου του Earley, αφού όπως έχει φανεί σε κάθε υπολογιστικό βήμα όλες οι εναλλακτικές ενός μη τερματικού συμβόλου εξετάζονται παράλληλα. Συνεπώς, βάσει των εξισώσεων (7.1) - (7.4), μπορούν να υπολογιστούν οι συντελεστές βεβαιότητας των μη τερματικών συμβόλων. Σε κάθε κανόνα έχει ανατεθεί ένας συντελεστής εξασθένισης ενώ σε κάθε μη τερματικό σύμβολο ένα συντιθέμενο κατηγόρημα C_{Xi}. Η τιμή του κατηγορήματος αυτού δε θα είναι παρά ο συντελεστής βεβαιότητας. Με αυτό τον τρόπο, θέτοντας κατάλληλες συναρτήσεις χειρισμού των κατηγορημάτων, μπορεί εύκολα να επεκταθεί η προτεινόμενη υλοποίηση προκειμένου να υποστηρίξει το μοντέλο των συντελεστών βεβαιότητας. Μάλιστα, απλώς αλλάζοντας τις συγκεκριμένες συναρτήσεις χειρισμού του μοντέλου των συντελεστών βεβαιότητας με άλλες διαφορετικών μοντέλων, μπορεί η ίδια υλοποίηση να βρει εφαρμογή και στα υπόλοιπα μοντέλα (στοχαστικό, ασάφειας κ.λ.π.). Τέλος, πρέπει να τονισθεί ότι στην περίπτωση που γίνει χρήση κάποιου μοντέλου αβεβαιότητας, δεν χάνεται η δυνατότητα χρήσης και επιπλέον συντιθέμενων και κληρονομούμενων κατηγορημάτων.

Κεφάλαιο 8

Συμπεράσματα και Μελλοντικές Επεκτάσεις

Έχοντας παρουσιάσει στα προηγούμενα κεφάλαια την μέχρι τώρα πρόοδο, ακολουθεί μια συνολική ανασκόπηση και αποτίμηση της εργασίας. Επιπλέον, παρουσιάζονται σκέψεις και ιδέες για εφαρμογές και μελλοντικές επεκτάσεις.

Η διδακτορική διατριβή είχε ως αντικείμενο την υλοποίηση σε υλικό (FPGA) κατάλληλων αλγορίθμων που προσθέτουν "ευφυΐα" σε ένα Ενσωματωμένο Σύστημα, μέσω της αναγνώρισης προτάσεων που ανήκουν σε γραμματικές χωρίς συμφραζόμενα καθώς και την αναγνώριση και υπολογισμό των αντίστοιχων κατηγορημάτων για προτάσεις που ανήκουν σε κατηγορικές γραμματικές. Τελικό στόχο αποτελεί η δημιουργία ενός συστήματος για την αυτόματη παραγωγή ευφυών ΕΣ, η διαδικασία περιγραφής των οποίων θα διαφέρει από την κλασική μέχρι στιγμής. Πιο συγκεκριμένα, αντί να περιγράφεται το ΕΣ με τις κλασικές γλώσσες Verilog και VHDL περιγράφεται σε υψηλότερο δηλωτικό επίπεδο με τη χρήση του συμβολισμού των κατηγορικών γραμματικών.

Η υλοποίηση των κατηγορικών γραμματικών, βασίζεται στην παράλληλη υλοποίηση του αλγορίθμου του Earley, του ταχύτερου αλγορίθμου συντακτικής αναγνώρισης στη βιβλιογραφία, ο οποίος όμως έχει επεκταθεί κατάλληλα ώστε να είναι εφικτή η υλοποίησή του με μικρότερες απαιτήσεις χώρου αλλά και χρονικές απαιτήσεις. Για το σημασιολογικό μέρος των κατηγορικών γραμματικών, είναι απαραίτητη η ύπαρξη κάποιας μονάδας που να εκτελεί τις αναγκαίες πράξεις. Κατά τη διάρκεια της συγκεκριμένης εργασίας, έγινε χρήση διαφορετικών αρχιτεκτονικών για τη συντακτική ανάλυση και τον υπολογισμό των κατηγορημάτων. Συγκεκριμένα, για τη συντακτική αναγνώριση παρουσιάστηκαν δύο υλοποιήσεις ενώ για τον υπολογισμό των κατηγορημάτων - και ανάλογα με τις απαιτήσεις σε υπολογιστική ισχύ και μέγεθος - επιλέγεται ενίοτε η χρήση μικροελεγκτή, η χρήση εξωτερικού μικροεπεξεργαστή γενικής χρήσης,

η χρήση εσωτερικού μικροεπεξεργαστή γενικής χρήσης και τελικά η χρήση επεξεργαστή εξειδικευμένης χρήσης ειδικά σχεδιασμένου για τις ανάγκες κάθε εφαρμογής:

- Η πρώτη προσέγγιση ήταν η επέκταση της υλοποίησης του αλγορίθμου των Pavlatos et al [28] προκειμένου να φέρει εις πέρας εκτός του μέρους της συντακτικής αναγνώρισης και το σημασιολογικό κομμάτι. Ο αλγόριθμος συντακτικής αναγνώρισης υλοποιήθηκε σε FPGA ενώ η αποτίμηση των κατηγορημάτων γινόταν σε έναν εξωτερικό επεξεργαστή RISC. Παράλληλα αναπτύχθηκε κατάλληλη μεθοδολογία διάσχισης του συντακτικού πίνακα προκειμένου να αποτιμώνται σωστά οι τιμές των κατηγορημάτων.

- Προκειμένου να αποφευχθεί η χρήση εξωτερικού επεξεργαστή, η αρχιτεκτονική τροποποιήθηκε κατάλληλα ώστε η αποτίμηση των κατηγορημάτων να εκτελείται από έναν μικροελεγκτή που υλοποιήθηκε σε FPGA, οδηγώντας στην ύπαρξη και των δύο υποτμημάτων της αρχιτεκτονικής στο ίδιο FPGA. Μάλιστα οι μικρές απαιτήσεις του μικροελεγκτή σε χώρο του FPGA, επέτρεψαν τον πειραματισμό με χρήση δύο μικροελεγκτών που εργάζονταν παράλληλα.

- Οι περιορισμοί στις δυνατότητες των μικροελεγκτών, οδήγησαν προς την αντικατάστασή τους από εσωτερικό επεξεργαστή soft core, οδηγώντας πάλι στην ύπαρξη και των δύο υποτμημάτων της αρχιτεκτονικής στο ίδιο FPGA. Με τη χρήση του soft core MicroBlaze, δόθηκαν δυνατότητες πρακτικά απεριόριστες για τις συναρτήσεις υπολογισμού των κατηγορημάτων που επιβάλλονται από τη σημασιολογία της κατηγορικής γραμματικής.

- Σε όλες τις προηγούμενες υλοποιήσεις, η διαδικασία αποτίμησης των κατηγορημάτων ήταν η πιο χρονοβόρα και μάλιστα σε τέτοιο βαθμό που αναζητήθηκαν λύσεις αντικατάστασης των επεξεργαστών/μικροελεγκτών με μηχανισμό αποτίμησης αποκλειστικά υλοποιημένο σε υλικό. Η πρώτη προσέγγιση επέτρεψε την χρήση S-Attributed κατηγορικών γραμματικών και επομένως τον υπολογισμό μόνο συντιθέμενων κατηγορημάτων, με τη βοήθεια μιας μεθοδολογίας βασισμένη σε στοίβες, η οποία οδηγεί στην αντιστοίχιση των σημασιολογικών κανόνων σε ενέργειες εισαγωγής/εξαγωγής δεδομένων σε κατάλληλες στοίβες.

- Παράλληλα με την αναζήτηση καλύτερων λύσεων για την αποτίμηση των κατηγορημάτων, αναζητήθηκαν και τρόποι περαιτέρω επιτάχυνσης του συντακτικού αναλυτή. Τα αποτελέσματα ήταν παραπάνω από ενθαρρυντικά, οδηγώντας στην υλοποίηση ενός συντακτικού αναλυτή (κατάλληλου για γλώσσες χωρίς συμφραζόμενα) η λειτουργία του οποίου στηριζόταν αποκλειστικά στη χρήση συνδυαστικής λογικής. Μάλιστα η πολυπλοκότητα του συγκεκριμένου αναλυτή είναι μόλις $O(log_2|G|)$, όπου $|G|$ είναι το μέγεθος της γραμματικής χωρίς συμφραζόμενα και συνεπώς δίνει εντυπωσιακά χαμηλότερους χρόνους εκτέλεσης.

- Επόμενο βήμα δε μπορούσε παρά να αποτελέσει η συνεργασία της νέας συνδυαστικής υλοποίησης με τη μεθοδολογία αποτίμησης των κατηγορημάτων που βασίζεται σε στοίβες. Διατηρώντας τη συνδυαστική του φύση, ο αναλυτής επεκτάθηκε ώστε να συνεργάζεται με κατηγορικές γραμματικές και να ακολουθεί την προτεινόμενη μεθοδολογία των στοιβών. Μάλιστα η μεθοδολογία αποτίμησης επεκτάθηκε ώστε να επιτρέπεται ο υπολογισμός και κληρονομούμενων κατηγορημάτων, επιτρέποντας πλήρως την ενσωμάτωση L-Attributed κατηγορικών γραμματικών. Προτάθηκε ουσιαστικά μια υλοποίηση σε υλικό πιο ισχυρή από το YACC που μπορεί να υπολογίσει μόνο συντιθέμενα κατηγορήματα.

- Σε κάθε στάδιο της έρευνας, ανεξαρτήτως του επιλεγμένου συντακτικού αναλυτή αλλά και του τόπου αποτίμησης των κατηγορημάτων, η αρχιτεκτονική έχει σχεδιαστεί πλήρως παραμετροποιήσιμη ώστε με χρήση ενός αυτόματου εργαλείου (πλατφόρμα ανάπτυξης) να παράγεται για κάθε γραμματική ο HDL κώδικας που περιγράφει το ΕΣ. Δίνεται συνεπώς η δυνατότητα στο χρήστη να παράγει άμεσα το ΕΣ περιγράφοντας μόνον την κατάλληλη κατηγορική γραμματική.

Όλες οι προτεινόμενες υλοποιήσεις δοκιμάστηκαν ως προς την απόδοσή τους και συγκρίθηκαν με άλλα παρεμφερή συστήματα, όπου αυτό ήταν εφικτό. Προοδευτικά οι δυνατότητες των προτεινόμενων υλοποιήσεων διευρύνονταν ενώ οι απαιτούμενοι χρόνοι εκτέλεσης ελαττώνονταν, καταλήγοντας στην τελική υλοποίηση, εκείνη που επιτρέπει την αποτίμηση κατηγορημάτων με χρήση συνδυαστικής λογικής και στοιβών.

Τελικό στόχο της προσπάθειας αποτελεί η κατασκευή ενός συστήματος στο ίδιο τσιπ (SoC) που θα είναι ικανό να εκτελέσει τόσο την συντακτική όσο και την σημασιολογική ανάλυση, με τη βοήθεια λογικών συνδυαστικών κυκλωμάτων και απλών δομικών στοιχείων σε όσο το δυνατό μικρότερο χρόνο. Η αρχιτεκτονική θα πρέπει να παραμείνει παραμετροποιήσιμη και να επιτρέπεται η αυτόματη παραγωγή, βάσει της κατηγορικής γραμματικής, τόσο των μονάδων που χρειάζονται για τη συντακτική αναγνώριση όσο και των μονάδων της σημασιολογικής αποτίμησης. Ο στόχος έχει εκπληρωθεί σε μεγάλο βαθμό αλλά δεν παύουν να υπάρχουν δυνατότητες για βελτιώσεις και μελλοντικές επεκτάσεις. Καταρχάς, γίνεται μια προσπάθεια [UR.C14] για υποστήριξη και κατηγορικών γραμματικών που απαιτούν πολλαπλά περάσματα (Multipass AGs) για αποτίμηση των κατηγορημάτων, εν αντιθέσει με τις μέχρι τώρα υποστηριζόμενες. Ένας ακόμη τομέας στον οποίο θα πρέπει να επικεντρωθεί η προσοχή, είναι ο τομέας των πρακτικών εφαρμογών. Ήδη αναζητούνται πέραν των ήδη παρουσιασθέντων και άλλες εφαρμογές στις οποίες η επιτάχυνση της υλοποίησης θα φανεί πραγματικά χρήσιμη. Υποψήφιες εφαρμογές μπορούν να βρεθούν στον τομέα των γλωσσών προγραμματισμού όπως η γλώσσα XML, στην οποία η γρήγορη μεταγλώτ-

τιση και αποτίμηση κατηγορημάτων θα βοηθήσει σημαντικά το έργο των εξυπηρετητών που έχουν να αποκριθούν ταυτόχρονα σε αιτήματα πολυάριθμων χρηστών, αλλά και η γλώσσα Java. Επιπλέον, μπορούν να αναζητηθούν και βιοϊατρικές εφαρμογές όπως η αναγνώριση του ηλεκτροκαρδιογραφήματος [UP.J4], αλλά και του γονιδιώματος RNA.

Τέλος, σημειώνεται ότι επιπλέον εκφραστική δύναμη, και συνεπώς μεγαλύτερο πεδίο εφαρμογών, μπορεί να προσδοθεί με την εισαγωγή αβεβαιότητας και ασάφειας. Δηλαδή, να επεκταθούν οι κατηγορικές γραμματικές, ώστε να υποστηρίζονται και έννοιες τέτοιες όπως ο βαθμός συμμετοχής, η αοριστία κ.α.. Η επέκταση αυτή δεν προϋποθέτει μεγάλες αλλαγές στον τρόπο που δουλεύουν ο προτεινόμενος μεταγλωττιστής και αποτιμητής και είναι υπό εξέλιξη μια προσπάθεια προς τη συγκεκριμένη κατεύθυνση [UP.J3].

Βιβλιογραφία

[1] Alfred V. Aho and Jeffrey D. Ullman. *The Theory of Parsing, Translation, and Compiling*, volume I: Parsing of *Series in Automatic Computation*. Prentice Hall, Englewood Cliffs, New Jersey, 1972.

[2] K. S. Fu. *Syntactic Pattern Recognition and Applications*. Prentice-Hall, December 1981.

[3] Elisabeth Andrι. The generation of multimedia presentations. In R. Dale, H. Moisl, and H. Somers, editors, *A Handbook of Natural Language Processing: techniques and applications for the processing of language as text*, pages 305–327. Marcel Dekker Inc., 2000.

[4] Janusz Jurek. Recent developments of the syntactic pattern recognition model based on quasi-context sensitive languages. *Pattern Recognition Letters*, 26(7), May 2005.

[5] G. Papakonstantinou and F. Gritzali. Syntactic filtering of ECG waveforms. *Computers and Biomedical research*, 14(2):158–167, 1981.

[6] G Papakonstantinou, C Moraitis, and T Panayiotopoulos. An attribute grammar interpreter as a knowledge engineering tool. *Angewandte Informatik*, 28(9):382–388, 1986.

[7] Jukka Paakki. Attribute grammar paradigms a high-level methodology in language implementation. *ACM Computing Surveys (CSUR)*, 27(2):196–255, 1995.

[8] Pierre Deransart and Jan Maluszynski. *A grammatical view of logic programming*. MIT Press, Cambridge, MA, USA, 1993.

[9] Donald E. Knuth. Semantics of context-free languages. *Mathematical Systems TheoryTHEORY*, 2:127–145, 1968.

[10] G. Papakonstantinou and J. Kontos. Knowledge representation with attribute grammars. *The Computer Journal*, 29(3):241–245, 1986.

[11] C. Voliotis, N. M. Sgouros, and G. Papakonstantinou. Attribute gramamr based modeling of concurrent constraint logic programming. *International Journal on Artificial Intelligence Tools (IJAIT)*, 4(3):383–411, 1995.

[12] Wen Hsiang Tsai and King Sun Fu. Attributed grammar –a tool for combining syntactic and statistical approaches to pattern recognition. *IEEE Transactions On Systems, Man and Cybernetics*, 10(12):873–885, 1980.

[13] P. Trahanias and E. Skordalakis. Syntactic pattern recognition of the ECG. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):648–657, 1990.

[14] Susan L. Graham, Michael Harrison, and Walter L. Ruzzo. An improved context-free recognizer. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 2(3):415–462, 1980.

[15] Walter Lawrence Ruzzo. *General context-free language recognition.* PhD thesis, Univ. of California, Berkeley, 1978.

[16] Y T Chiang and King-Sun Fu. Parallel parsing algorithms and VLSI implementations for syntactic pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:302–314, 1984.

[17] T. Kasami. An efficient recognition and syntax analysis for context-free languages. *Science Report AF CRL-65-758*, Air Force Cambridge Research Laboratory Bedford,Mass., 1965.

[18] D. H. Younger. Recognition and parsing of context-free languages in n^3. *Information and Control*, 10:189–208, 1967.

[19] Jay Earley. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102, 1970.

[20] M. A. Harrison. *Introduction to Formal Language Theory.* Addison-Wesley, Reading, Mass., 1978.

[21] Alica Kelemenová. Complexity of normal form grammars. *Theoretical Computer Science*, 28(3):299–314, February 1984.

[22] Norbert Blum and Robert Koch. Greibach normal form transformation revisited. *Information and Computation*, 150(1):112–118, 1999.

[23] H. D. Cheng and K.S. Fu. Algorithm partition and parallel recognition of general context-free languages using fixed-size vlsi architecture. *Pattern Recognition*, 19(5):361–372, 1986.

[24] Oscar H. Ibarra, Ting-Chuen Pong, and Stephen M. Sohn. Parallel recognition and parsing on the hypercube. *IEEE Transactions On Computers*, 40(6):764–770, 1991.

[25] Dong-Yul Ra and Jong-Hyun Kim. A parallel parsing algorithm for arbitrary context-free grammars. *Information Processing Letters*, 58(2):87–96, 1996.

[26] Cristian Ciressan, Eduardo Sanchez, Martin Rajman, and Jean-Cedric Chappelier. An fpga-based coprocessor for the parsing of context-free grammars. In *FCCM '00: Proceedings of the 2000 IEEE Symposium on Field-Programmable Custom Computing Machines*, page 236, Washington, DC, USA, 2000. IEEE Computer Society.

[27] Jacir L. Bordim, Yasuaki Ito, and Koji Nakano. Accelerating the cky parsing using fpgas. In *HiPC '02: Proceedings of the 9th International Conference on High Performance Computing*, pages 41–51, London, UK, 2002. Springer-Verlag.

[28] Christos Pavlatos, Ioannis Panagopoulos, and George Papakonstantinou. A programmable pipelined coprocessor for parsing applications. In *Workshop on Application Specific Processors (WASP) CODES*, Stockholm, September 2004.

[29] S. C. Johnson. *Yacc̊yet another compiler compiler.* Computing Science Technical Report 32. AT&T Bell Laboratories, Murray Hill, N.J., 1975.

[30] Bison - gnu parser generator. http://www.gnu.org/software/bison/.

[31] Robert W. Gray, Steven P. Levi, Vincent P. Heuring, Anthony M. Sloane, and William M. Waite. Eli: a complete, flexible compiler construction system. *Communications of the ACM*, 35(2):121–130, 1992.

[32] Eli: An integrated toolset for compiler construction. http://eli-project.sourceforge.net.

[33] U. Kastens. LIDO – a specification language for attribute grammars. *Betriebsdatenerfassung, Fachbereich*, October 1989.

[34] Utrecht university attribute grammar. http://www.cs.uu.nl/wiki/bin/view/HUT/Attribute GrammarSystem.

[35] H. D. Cheng and X. Cheng. Shape recognition using a fixed-size VLSI architecture. *International Journal of Pattern Recognition and Artificial Intelligence*, 9(1), 1995.

[36] Ioannis Panagopoulos, Christos Pavlatos, and George Papakonstantinou. An embedded microprocessor for intelligent control. *Journal of Intelligent and Robotic Systems*, 42(2):179–211, 2005.

[37] R. W. Floyd. The syntax of programming languages - a survey. *IEEE Transactions On Electronic Computers*, 13(4), 1964.

[38] Noam Chomsky. Three models for the description of language. In *IRE Transactions on Information Theory (2)*, pages 113–124, 1956.

[39] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition, 2003.

[40] Edited by S.G Tzafestas. *Knowledge-based system diagnosis, supervision, and control.* Plenum Press, New York, 1989.

[41] Hassan Aït-Kaci. *Warren's abstract machine : a tutorial reconstruction.* MIT Press, Cambridge, Mass., 1991.

[42] Gopal Gupta, Enrico Pontelli, Khayri A.M. Ali, Mats Carlsson, and Manuel V. Hermenegildo. Parallel execution of prolog programs: a survey. *ACM Transactions on Programming Languages and Systems*, 23(4):472–602, 2001.

[43] *Communications of the ACM*, 36(3), 1993.

[44] Roland Karlsson. *A High Performance OR-parallel Prolog System.* PhD thesis, The Royal Institute of Technology, Stockholm, March 1992.

[45] Christos Pavlatos, Alexandros Dimopoulos, and George Papakonstantinou. An intelligent embedded system for control applications. In *Workshop on Modeling and Control of Complex Systems*, Cyprus, 2005.

[46] Christos Pavlatos, Alexandros Dimopoulos, and George Papakonstantinou. An embedded system for the electrocardiogram recognition. *EMBEC'05*, November 2005.

[47] Alexandros Dimopoulos, Christos Pavlatos, Ioannis Panagopoulos, and George Papakonstantinou. An efficient hardware implementation for AI applications. *Lecture Notes in Computer Science*, 3955:35–45, April 2006.

[48] Themis Panayiotopoulos and George Papakonstantinou. A full theorem prover under uncertainty. *Journal of Intelligence and Robotic Systems*, 7:139–149, 1993.

[49] G Papakonstantinou, C Moraitis, and T Panayiotopoulos. An attribute grammar interpreter as a knowledge engineering tool. *Angewandte Informatik*, 28(9):382–388, 1986.

[50] Pierre Deransart, Bernard Lorho, and Jan Maluszynski, editors. *Programming Language Implementation and Logic Programming, 1st International Workshop PLILP'88, Orléans, France, May 16-18, 1988, Proceedings*, volume 348 of *Lecture Notes in Computer Science*. Springer, 1989.

[51] Fernando C. N. Pereira and David H. D. Warren. Definite clause grammars for language analysis - a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, 13(3):231–278, 1980.

[52] Themis Panayiotopoulos, George Papakonstantinou, and G. Sgouros. An attribute grammar interpreter for inexact reasoning. *Information and Software Technology*, 32(7), 1993.

[53] Hans-Juergen Boehm and Willy Zwaenepoel. Parallel Attribute Grammar Evaluation. In *Proceedings of the Seventh International Conference on Distributed Computing Systems*, pages 347–354, Berlin, Germany, 1987.

[54] A. Klaiber and M. Gokhale. Parallel evaluation of attribute grammars. *IEEE Transactions on Parallel and Distributed Systems*, 03(2):206–220, 1992.

[55] Andrew Koulouris Christos Pavlatos and George Papakonstantinou. Hardware Implementation of Syntactic Pattern Recognition Algorithms. In *IASTED International Conference on Signal Processing and Pattern Analysis (SPPRA)*, pages 360–365, Rhodes, Greece, 2003.

[56] Takehiro Tokuda and Yoshimichi Watanabe. An attribute evaluation of context-free languages. *Information Processing Letters*, 52(2):91–98, 1994.

[57] Martha Sideri, Sophocles Efremidis, and G. Papaconstantinou. Semantically driven parsing of context-free languages. *The Computer Journal*, 32(1):91–93, 1989.

[58] Christos C. Pavlatos Ioannis P. Panagopoulos and George K. Papakonstantinou. An embedded system for artificial intelligence applications. *International Journal of Computational Intelligence*, 1(1):10–24, 2004.

[59] G Papakonstantinou, E Skordalakis, and F Gritzali. An attribute grammar for QRS detection. *Pattern Recognition*, 19(4):297–303, 1986.

[60] Picoblaze ^{TM}8-bit embedded microcontroller user guide. `http://www.xilinx.com/support/documentation/ip_documentation/ug129.pdf`.

[61] Microblaze processor reference guide. `http://www.xilinx.com/support/documentation/sw_manuals/edk92i_mb_ref_guide.pdf`.

[62] Embedded linux/microcontroller project. http://www.uclinux.org.

[63] The freertos.org project. http://www.freertos.org.

[64] Xilkernel. http://www.xilinx.com/ise/embedded/edk91i_docs/xilkernel_v3_00_a.pdf.

[65] Xilinx ise 8.2i software manuals and help. `http://www.xilinx.com/support/sw_manuals/xilinx82/index.htm`.

[66] Χαράλαμπος Χ. Νάκος. Το Λειτουργικό Σύστημα MicroEmpix/FPGA για Εφαρμογές Ενσωματωμένων Συστημάτων. In *Διπλωματική Εργασία, Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, ΕΜΠ*, Αθήνα, Δεκέμβριος 2006.

[67] D. Vitulano. A syntactic approach applied to materials degradation. *Pattern Recognition Letters*, 20(1), January 1999.

[68] E. Giakoumakis, G. Papaconstantinou, and E. Skordalakis. Rule-based systems and pattern recognition. *Pattern Recognition Letters*, 5:267–272, 1987.

[69] Colin de la Higuera. A bibliographical study of grammatical inference. *Pattern Recognition*, 38:1332–1348, 2005.

[70] Java technology. `http://java.sun.com`.

[71] *Java Language Specification Second Edition.* Sun Microsystems, Inc., 2000.

[72] Y. Li, H. Yang, and H. V. Jagadish. NaLIX: an interactive natural language interface for querying xml. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 900–902, Baltimore, Maryland, 2005.

[73] A. Yates, O. Etzioni, and D. Weld. A reliable natural language interface to household appliance. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 189 – 196, 2003.

[74] C. Pavlatos, A.Dimopoulos, and G. Papakonstantinou. Hardware natural language interface. In *The 4^{th} IFIP International Conference on Artificial Intelligence Applications & Innovations (AIAI)*, Athens, Greece, September 2007.

[75] Christos Pavlatos, Alexandros C. Dimopoulos, Andrew Koulouris, Theodore Andronikos, Ioannis Panagopoulos, and George Papakonstantinou. Efficient

reconfigurable embedded parsers. *Computer Languages, Systems & Structures*, 35(2):196 – 215, 2009.

[76] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.

[77] Ion Androutsopoulos, Graeme D. Ritchie, and Peter Thanisch. Natural language interfaces to databases - an introduction. *CoRR*, cmp-lg/9503016, 1995.

[78] Linguistic Technology. *English Wizard – Dictionary Administrator's Guide*. Littleton, MA, USA, 1997.

[79] Martin Schoeberl. A java processor architecture for embedded real-time systems. *Journal of Systems Architecture*, 54(1-2):265–286, January-February 2008.

[80] B. G. Buchanan and E. H. Shortliffe. *Rule Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project (The Addison-Wesley series in artificial intelligence)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984.

[81] Νικόλαος Σ. Παπασπύρου και Εμμανουήλ Σ. Σκορδαλάκης. *Μεταγλωττιστές*. Εκδόσεις Συμμετρία, 2002.

[82] John Catsoulis. *Designing Embedded Hardware, Second Edition*. O'Reilly, 2005.

[83] Clive Maxfield. *The Design Warrior's Guide to FPGAs*. Academic Press, Inc., Orlando, FL, USA, 2004.

[84] Mark Balch. *Complete Digital Design: A Comprehensive Guide to Digital Electronics and Computer System Architecture (Professional Engineering)*. McGraw-Hill Professional,Ltd, 2003.

[85] Vojin G. Oklobdzija, editor. *Digital Design and Fabrication (Computer Engineering Handbook)*. CRC Press, 2007.

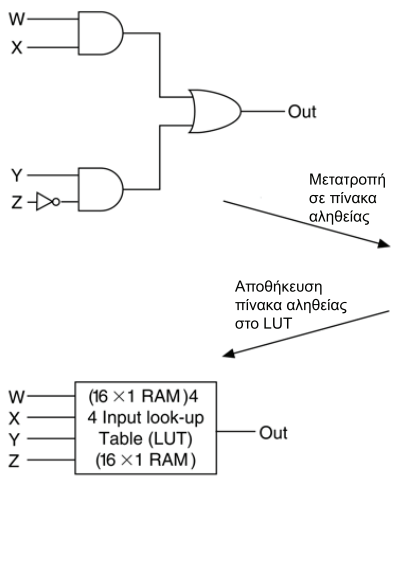
Παράρτημα Α

Field Programmable Gate Array

Τα **F**ield **P**rogrammable **G**ate **A**rray (FPGA) είναι ολοκληρωμένα κυκλώματα, η εσωτερική λειτουργία των οποίων καθορίζεται από το χρήστη. Μέσω προγραμματισμού μπορεί ο χρήστης να υλοποιεί την επιθυμητή συμπεριφορά τους, περιγράφοντας τη συμπεριφορά σε χαμηλό επίπεδο λογικών πυλών ή ακόμη και σε υψηλότερο με χρήση διαδικαστικού προγραμματισμού.

Α.1 Τεχνολογία των FPGA

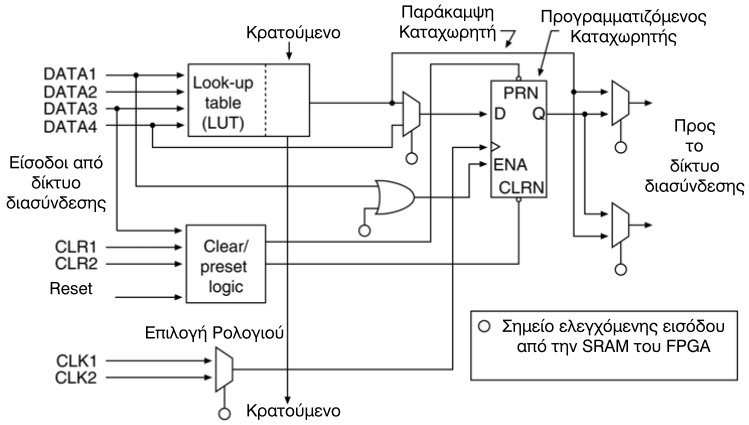
Τα FPGA έχουν τη μεγαλύτερη πυκνότητα και τις πιο ανεπτυγμένες δυνατότητες προγραμματιζόμενης λογικής. Το μέγεθος των FPGA καθορίζεται κυρίως από το πλήθος των χρησιμοποιούμενων ή ισοδύναμων πυλών, δηλαδή του μέγιστου αριθμού των πυλών NAND δύο εισόδων που είναι διαθέσιμες στο FPGA. Ανάλογα με τον κατασκευαστή, διαφορετικά μέτρα χρησιμοποιούνται για τον καθορισμό του μεγέθους, επομένως το μέγεθος δεν είναι απόλυτο αλλά μια χονδρική εκτίμηση. Επιπλέον, η χρησιμοποίηση των πυλών μπορεί να διαφοροποιείται από υλοποίηση σε υλοποίηση.
Τα περισσότερα FPGA χρησιμοποιούν SRAM πίνακες αναζήτησης (LUT) για την υλοποίηση λογικών κυκλωμάτων με πύλες. Ένα παράδειγμα χρήσης ενός LUT για την αναπαράσταση ενός δικτύου πυλών παρουσιάζεται στο σχήμα Α.1. Αρχικά το δίκτυο μετατρέπεται σε πίνακα αληθείας τεσσάρων εισόδων και μιας εξόδου. Ακολούθως ο πίνακας αληθείας φορτώνεται στην SRAM του LUT, όταν προγραμματίζεται το FPGA. Οι τέσσερις είσοδοι χρησιμοποιούνται ως διεύθυνση για τη RAM και η έξοδος του δικτύου (και του πίνακα αληθείας), είναι τα δεδομένα που αποθηκεύονται στην SRAM του LUT. Με αυτή την τεχνική, η SRAM του LUT υλοποιεί το ισοδύναμο κύκλωμα εκτελώντας μια αναζήτηση σε πίνακα RAM, αντί για χρήση πραγματικών λογικών πυλών. Σε ορισμένες συσκευές, οι LUT μπορούν να χρησιμοποιηθούν επιπλέον και για την υλοποίηση RAM ή καταχωρητών ολίσθησης.
Εσωτερικά, τα FPGA περιέχουν πολλαπλά αντίγραφα των στοιχείων της βασικής προγραμματιζόμενης λογικής (Logic Element - LE), η οποία ονομάζεται λογικό κύτταρο



Περιεχόμενα RAM (LUT)				
Διεύθυνση				DATA
W	X	Y	Z	Out
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Σχήμα Α.1: Χρήση LUT του FPGA για την υλοποίηση δικτύου λογικών πυλών.

(Logic Cell - LC) ή προγραμματιζόμενο λογικό μπλοκ (Configurable Logic Block - CLB). Ένα τυπικό LE φαίνεται στο σχήμα Α.2. Χρησιμοποιώντας ένα ή περισσότερα LUT, το LE μπορεί να υλοποιήσει ένα δίκτυο από αρκετές λογικές πύλες, η έξοδος του οποίου μπορεί να τροφοδοτεί ένα προγραμματιζόμενο flip-flop εάν ζητείται ή σε περιπτώσεις συνδυαστικής λογική απλώς να παρακάμπτεται. Ορισμένες συσκευές FPGA περιλαμβάνουν δύο παρόμοια κυκλώματα σε κάθε LE ή CLB. Πολυάριθμα LE ή CLB είναι διατεταγμένα σε ένα δισδιάστατο πίνακα πάνω στο τσιπ. Η τρέχουσα τεχνολογία FPGA επιτρέπει την ύπαρξη από μερικές εκατοντάδες μέχρι πάνω από εκατό χιλιάδες LE. Για την εκτέλεση πολύπλοκων πράξεων, ένα LE μπορεί να συνδεθεί αυτόματα με



Σχήμα Α.2: Ένα τυπικό στοιχείο βασικής προγραμματιζόμενης λογικής.

άλλα LE στο τσιπ, χρησιμοποιώντας ένα προγραμματιζόμενο δίκτυο διασυνδέσεων. Προφανώς, το προγραμματιζόμενο δίκτυο διασυνδέσεων εμπεριέχεται στο FPGA. Το προγραμματιζόμενο δίκτυο διασυνδέσεων που χρησιμοποιείται για τη σύνδεση των LE περιέχει διασυνδέσεις κατά μήκος όλου του τσιπ σε σειρές και σε στήλες. Επιπλέον, το προγραμματιζόμενο δίκτυο διασυνδέσεων συνήθως περιέχει μικρότερες και ταχύτερες διασυνδέσεις που περιορίζονται μόνο στα γειτονικά λογικά στοιχεία. Οι καθυστερήσεις των εσωτερικών διασυνδέσεων είναι ένας σημαντικός παράγοντα της συνολικής επίδοσης, δεδομένου ότι είναι της ίδιας τάξης μεγέθους με τους χρόνους καθυστέρησης των λογικών στοιχείων. Χρησιμοποιώντας μια συντομότερη διαδρομή διασύνδεσης σημαίνει λιγότερος χρόνος καθυστέρησης. Για την παραγωγή αθροιστών υψηλών επιδόσεων, υπάρχει συχνά μια ειδική διασύνδεση με γειτονικά λογικά στοιχεία για μεταφορά του κρατουμένου.

Μια δυσκολία που εμφανίζεται σε μεγάλα FPGA αφορά στο ρολόι και στις καθυστερήσεις που μπορεί να εμφανιστούν λόγω διασύνδεσης, οι οποίες ποικίλλουν σημαντικά στις διάφορες περιοχές του FPGA. Για το λόγο αυτό, υπάρχουν ειδικοί διάδρομοι για τη διανομή του σήματος του ρολογιού προς όλα τα flip-flops στη συσκευή την ίδια στιγμή. Η χρήση των διαδρόμων αυτή εξασφαλίζεται ορισμένες φορές από το λογισμικό τοποθέτησης ενώ άλλες πρέπει να εντοπιστούν τα σήματα από τον ίδιο το σχεδιαστή.

Ακίδες (pin) E/E γενικού σκοπού στο FPGA περιέχουν επαναπρογραμματιζόμενους αμφίδρομους οδηγούς τριών καταστάσεων (tristate drivers) και flip-flops. Οι ακίδες αυτές μπορούν να προγραμματιστούν για είσοδο, έξοδο ή και αμφίδρομη λειτουργία. Το σήμα E/E μπορεί να φορτωθεί σε ακίδες E/E του flip-flop ή να συνδεθεί άμεσα το δίκτυο διασύνδεσης και από εκεί να δρομολογηθεί σε εσωτερικά λογικά στοιχεία. Σε μεγάλα FPGA πολλές ακίδες τάσης και γείωσης είναι επίσης απαραίτητες. Σημειώνεται ότι ο εσωτερικός πυρήνας του FPGA λειτουργεί υπό τάση από 1,5 έως 5 Volt. Όταν ένα σχέδιο προσεγγίσει το μέγιστο μέγεθος της συσκευής, είναι δυνατόν να εξαντληθούν τα στοιχεία προγραμματιζόμενης λογικής, οι διασυνδέσεις ή οι διαθέσιμες ακίδες. Οι οικογένειες FPGA περιλαμβάνουν πολλές συσκευές με ευρύ φάσμα διαθέσιμων πυλών και με διαφορετικό πλήθος. Για την ελαχιστοποίηση του κόστους, μέρος του προβλήματος σχεδιασμού είναι η επιλογή συσκευής με αρκετά στοιχεία προγραμματιζόμενης λογικής, αρκετές διασυνδέσεις και ακίδες. Τέλος, ένας άλλος σημαντικός παράγοντας επιλογής συσκευής είναι η ταχύτητα ή συχνότητα ρολογιού που απαιτείται για ένα συγκεκριμένο σχεδιασμό.

Η δρομολόγηση του κυκλώματος (routing) ανάγεται στον κύριο περιοριστικό παράγοντα της απόδοσης των FPGA, λόγω της μη ντετερμινιστικής δρομολόγησης ενός δικτύου διασύνδεσης με πολλαπλά μονοπάτια. Τα μονοπάτια μεταξύ των λογικών κυττάρων μπορούν να λάβουν πολλαπλές διαδρομές, ορισμένες από τις οποίες μπορούν να παρέχουν την ίδια καθυστέρηση. Ωστόσο, οι πόροι διασύνδεσης είναι πεπερασμέ-

νοι, και γρήγορα προκύπτουν συνθήκες ανταγωνισμού ανάμεσα σε μονοπάτια για την ίδια διαδρομή. Οι κατασκευαστές των FPGA παρέχουν εργαλεία λογισμικού για να τη μετατροπή ενός πηγαίου κώδικα στην τελική μορφή, ένα δυαδικό αρχείο το οποίο χρησιμοποιείται για τον κατάλληλο προγραμματισμό του FPGA. Ανάλογα με την πολυπλοκότητα του σχεδίου (π.χ. την ταχύτητα και την πυκνότητα), η δρομολόγηση του σχεδίου στο FPGA μπορεί να χρειαστεί από μερικά λεπτά μέχρι πολλές ώρες. Ακριβώς λόγω των μη ντετερμινιστικών πόρων, ο χρόνος του FPGA μπορεί να ποικίλλει σημαντικά, ανάλογα με την ποιότητα της λογικής τοποθέτησης. Για μεγάλα και γρήγορα σχέδια απαιτούνται επαναληπτικοί αλγόριθμοι δρομολόγησης και τοποθέτησης. Η ανθρώπινη παρέμβαση μπορεί να είναι κρίσιμη για την επιτυχή δρομολόγηση και το χρονοδιάγραμμα ενός σύνθετου σχεδιασμού. Ορισμένες φορές χρειάζεται ένας μηχανικός να χωρίσει χειροκίνητα τη λογική σε ομάδες και μετά ρητά να τοποθετήσει τις ομάδες αυτές στο FPGA. Ωστόσο, όσο αυξάνει ο βαθμός της ανθρώπινης παρέμβασης, τόσο περιορίζεται η δρομολόγηση του λογισμικού και μειώνεται ο αριθμός των διαθέσιμων παραλλαγών που μπορούν να δοκιμαστούν για να επιτευχθεί ένα επιτυχημένο αποτέλεσμα.

Α.2 Αρχιτεκτονική FPGA Νέας Γενιάς

Οι νεότερες γενεές των FPGA συνεχίζουν να αυξάνονται σε μέγεθος και να αποκτούν επιπρόσθετα χαρακτηριστικά. Αρκετά FPGA περιλαμβάνουν πια ένα συνδυασμό πόρων που στηρίζονται σε LUT και πόρων που στηρίζονται σε όρους λογικών γινομένων. Οι τελευταίοι είναι πιο αποτελεσματικοί για πιο πολύπλοκες λογικές ελέγχου που εμφανίζονται σε μεγάλες μηχανές καταστάσεων (state machines) και σε αποκωδικοποιητές διεύθυνσης. PLL (Phase-locked Loops) είναι επίσης διαθέσιμα στα περισσότερα νεότερα FPGA επιτρέποντας τον πολλαπλασιασμό, τη διαίρεση και την αλλαγή φάσης των σημάτων του ρολογιού.

Τα πλέον πρόσφατα FPGA περιέχουν επιπλέον εσωτερικά ή ενσωματωμένα μπλοκ μνήμης RAM. Αν και η μνήμη RAM μπορεί να υλοποιηθεί με τη χρήση της λογικής του FPGA, είναι πιο αποδοτική η χρήση εξειδικευμένων μπλοκ μνήμης για μεγαλύτερα ποσά μνήμης RAM και ROM. Αυτά τα μπλοκ μνήμης είναι διαμοιρασμένα σε όλο το τσιπ και μπορούν να χρησιμοποιηθούν όταν το τσιπ προγραμματιστεί. Η χωρητικότητα αυτής της εσωτερικής μνήμης είναι περιορισμένη σε μερικές χιλιάδες bit ανά μπλοκ. Υλοποιήσεις με μεγάλες απαιτήσεις σε μνήμη θα χρειαστούν πάλι πρόσθετη εξωτερική συσκευή μνήμης.

Ένα επιπλέον χαρακτηριστικό νέων FPGA είναι η εσωτερική υλοποίηση, σε υλικό, πολλαπλασιαστών. Κάτι τέτοιο προσφέρει υψηλότερες επιδόσεις έναντι πολλαπλασιαστών κατασκευασθέντων με τη χρήση των στοιχείων της λογικής του FPGA και είναι χρήσιμο για εφαρμογές ψηφιακής επεξεργασίας σήματος (DSP) και γραφικών,

οι οποίες απαιτούν εντατικές πράξεις πολλαπλασιασμού. Αντίστοιχα, μεγάλα FPGA είναι διαθέσιμα με πυρήνες μικροεπεξεργαστών RISC.
Τα πλέον πρόσφατα FPGA υποστηρίζουν μια πλειάδα από πρότυπα E/E. Κάτι τέτοιο διευκολύνει την διασύνδεση με εξωτερικές συσκευές υψηλής ταχύτητας. Πολλά διαφορετικά πρότυπα E/E χρησιμοποιούνται για επεξεργαστές, μνήμες και περιφερειακές συσκευές - το επιθυμητό πρότυπο E/E επιλέγεται κατά τον προγραμματισμό της συσκευής. Γενικότερα η ενσωμάτωση λογικών πυρήνων αυξάνει το κόστος των FPGA λόγω των αδειών χρήσης που πρέπει αγοραστούν, αλλά η αυξημένη απόδοση που προσφέρεται από τις ενσωματωμένες αυτές υλοποιήσεις αποζημιώνει τον αγοραστή.
Τέλος, οι κατασκευαστές FPGA έχουν αρχίσει να χρησιμοποιούν περιττή λογική στα τσιπ για τη μείωση των ελαττωματικών στοιχείων. Όπως κάθε συσκευή VLSI, όσο αυξάνεται το μέγεθος του FPGA, αυξάνει και η πιθανότητα ενός κατασκευαστικού ελαττώματος. Ορισμένες συσκευές τώρα περιλαμβάνουν επιπλέον γραμμές ή στήλες λογικών στοιχείων. Αφού η συσκευή δοκιμαστεί, τυχόν ελαττωματικές σειρές λογικών στοιχείων διαγράφονται αυτόματα και αντικαθίστανται από μια πλεονάζουσα γραμμή. Αυτή η διαδικασία ακολουθείται όταν η συσκευή δοκιμάζεται αρχικά και ουσιαστικά η ύπαρξη της επιπλέον λογικής δεν επηρεάζει χρήστη.

A.3 Εφαρμογές των FPGA

Διάφοροι παράγοντες όπως υψηλότερη πυκνότητα, μεγαλύτερη ταχύτητα, αυξημένη πίεση για μείωση του χρόνου προς την αγορά οδήγησαν σε χρήση των προγραμματιζόμενων λογικών συσκευών σε μια ευρύτερη ποικιλία σχεδίων. Σχέδια με τη χρήση ενός FPGA μπορεί να χρειαστούν μόλις μερικές εργατοεβδομάδες μηχανικών αντί ολόκληρων μηνών που θα χρειάζονταν με χρήση ASIC. Τα τελευταία βέβαια, ακριβώς επειδή είναι πλήρως προσαρμοσμένα VLSI κυκλώματα χωρίς προγραμματιζόμενες διασυνδέσεις - και τις αντίστοιχες καθυστερήσεις, παρέχουν ταχύτερο ρολόι από τα FPGA. Τα ASIC ως πλήρως προσαρμοσμένα σχέδια VLSI δεν απαιτούν προγραμματιζόμενα κυκλώματα διασύνδεσης, επομένως χρησιμοποιούν λιγότερο χώρο στο τσιπ και έχουν χαμηλότερο κόστος παραγωγής ανά μονάδα σε μεγάλες ποσότητες. Το αρχικό κόστος για την ανάπτυξη ASIC από προσαρμοσμένα σχέδια VLSI είναι υψηλό, επομένως συχνά αναπτύσσονται με τη χρήση FPGA.
Τα FPGA έχουν γίνει πια ευρέως χρησιμοποιούμενα κατά την τελευταία δεκαετία. Οι υψηλότερες πυκνότητες, η βελτίωση της απόδοσης, καθώς και τα πλεονεκτήματα κόστους επέτρεψαν τη χρήση των FPGA σε μια ευρύτερη ποικιλία σχεδίων. Μια πρόσφατη έρευνα αγοράς [85] έδειξε ότι υπάρχουν πάνω από 10 φορές περισσότερες υλοποιήσεις βασισμένες σε FPGA από αντίστοιχες που βασίζονται σε ASIC. Οι νέες γενιές FPGA περιέχουν αρκετά εκατομμύρια πύλες και μπορεί να προσφέρουν συχνότητες ρολογιού που να πλησιάζει το 1 GHz. Πεδία εφαρμογής περιλαμβάνουν αντικα-

ταστάσεις παλαιών τσιπ, DSP, επεξεργασία εικόνας, εφαρμογές πολυμέσων, δικτύωση υψηλής ταχύτητας και εξοπλισμό επικοινωνιών, πρωτόκολλα όπως εκείνα των διαδρόμων δεδομένων π.χ. PCI, βοηθητικό ρόλο στη διασύνδεση του μικροεπεξεργαστή με άλλα λογικά κυκλώματα ή και συνεπεξεργαστές (coprocessors). RISC μικροεπεξεργαστές έχουν αρχίσει να κάνουν την εμφάνισή τους μέσα σε μεγάλα FPGA που προορίζονται για υλοποιήσεις System-on-a-Chip (SoC). Για εφαρμογές χωρίς κρίσιμο χρονικό σχεδιασμό έχουν μέγιστη δυνατή συχνότητα ρολογιού της τάξης των 50-400 MHz. Συχνότητες μέχρι το 1 GHz έχουν επιτευχθεί σε νέες γενιές FPGA, ενώ σε ορισμένα υπάρχει και η δυνατότητα για διασύνδεση εξωτερικού ρολογιού για υποστήριξη μεταφορών δεδομένων που αγγίζουν τα 10 GHz.

Αρκετά μεγάλα FPGA χρησιμοποιούνται για την κατασκευή υλικού εξομοίωσης (emulators), δηλαδή συστημάτων ειδικά σχεδιασμένων για τη δημιουργία πρωτοτύπων καθώς και τον έλεγχο περιπλόκων σχεδίων τα οποία αργότερα πιθανώς θα υλοποιηθούν σε ASIC ή VLSI. Αρκετοί από τους τελευταίους μικροεπεξεργαστές συμπεριλαμβανομένων και εκείνων της Intel και της AMD x86 που χρησιμοποιούνται σε προσωπικούς υπολογιστές υλοποιήθηκαν αρχικά σε υλικού εξομοίωσης Ένα νέο πεδίο εφαρμογής των FPGA είναι η συνεργασία τους με υπολογιστικά συστήματα. Τα FPGA επαναπρογραμματίζονται γρήγορα κατά τη διάρκεια των συνηθισμένων εργασιών, ώστε να μπορούν να εκτελούν διαφορετικές λειτουργίες σε διαφορετικές χρονικές στιγμές βοηθώντας συγκεκριμένες απαιτητικές εφαρμογές.

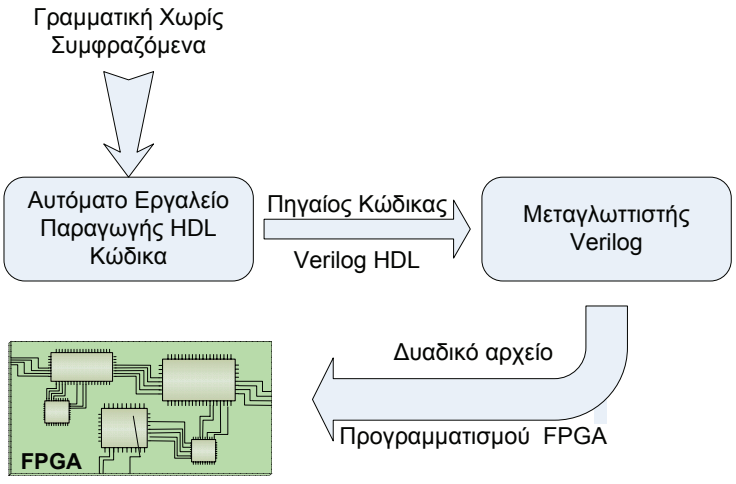
Παράρτημα Β

Εργαλείο Αυτόματης Παραγωγής Κώδικα HDL

Για τη διευκόλυνση του χρήστη κατά την παραγωγή του κώδικα HDL που περιγράφει το επιθυμητό σύστημα (βάσει της δοσμένης γραμματικής), κρίθηκε αναγκαία η δημιουργία ενός αυτόματου εργαλείου. Το αυτόματο εργαλείο δέχεται ως είσοδο την περιγραφή της γραμματικής και παράγει τον κώδικα HDL που υλοποιεί τα ζητούμενα. Έτσι, ο χρόνος που απαιτείται από το σχεδιαστή για την υλοποίηση ενός συστήματος, ελαττώνεται δραματικά.

Β.1 Κωδικοποίηση Γραμματικών Χωρίς Συμφραζόμενα

Η αρχιτεκτονική που παρουσιάστηκε στο κεφάλαιο 5 υλοποιεί ένα συντακτικό αναλυτή για γραμματικές χωρίς συμφραζόμενα. Όπως αναλύθηκε και στο σχετικό κεφάλαιο, η προτεινόμενη αρχιτεκτονική είναι σε παραμετροποιήσιμη μορφή και με τη βοήθεια



Σχήμα Β.1: Διαδικασία Παραγωγής με χρήση Αυτόματου Εργαλείου.

αυτόματου εργαλείου, παράγεται η περιγραφή του συντακτικού αναλυτή σε Verilog, για δεδομένη γραμματική. Συνεπώς, ο χρήστης πρέπει να δώσει με κάποια μορφή την γραμματική χωρίς συμφραζόμενα. Το αυτόματο εργαλείο, το οποίο είναι γραμμένο σε C, επεξεργάζεται την είσοδο και παράγει ως έξοδο τα αρχεία πηγαίου κώδικα HDL τα οποία ακολούθως με χρήση των εργαλείων του κατασκευαστή του FPGA, μετατρέπονται σε ένα δυαδικό αρχείο έτοιμο να προγραμματίσει κατάλληλα το FPGA. Η συγκεκριμένη διαδικασία αναπαρίσταται σχηματικά στο σχήμα Β.1.

Η μορφή στην οποία πρέπει να είναι το αρχείο περιγραφής της γραμματικής είναι πολύ απλή. Για το συντακτικό κανόνα A → BC, όπου τα A, B και C είναι μη τερματικά σύμβολα η αντίστοιχη μορφή είναι "$< A > -- >\ \ < B >< C >$", ενώ για την περίπτωση τερματικών συμβόλων, τα τελευταία πρέπει να είναι κλεισμένα μέσα σε εισαγωγικά ("). Σαν παράδειγμα παρουσιάζεται στον πίνακα Β.1 η κωδικοποίηση για τη γραμματική G_{op} του κεφαλαίου 6 στον πίνακα Β.1.

Συντακτικός Κανόνας	Συντακτικός Κανόνας στο Αρχείο Περιγραφής
$S \rightarrow E$	$< S > \ -- > \ < E >$
$E_1 \rightarrow T + E_2$	$< E > \ -- > \ < T >< "+" >< E >$
$E \rightarrow T$	$< E > \ -- > \ < T >$
$T \rightarrow F * T$	$< T > \ -- > \ < F >< "*" >< T >$
$T \rightarrow F$	$< T > \ -- > \ < F >$
$F \rightarrow (E)$	$< F > \ -- > \ < "(" >< E >< ")" >$
$F \rightarrow N$	$< F > \ -- > \ < N >$
$N \rightarrow DN$	$< N > \ -- > \ < D >< N >$
$N \rightarrow D$	$< N > \ -- > \ < D >$
$D \rightarrow 0$	$< D > \ -- > \ < "0" >$
...	...
$D \rightarrow 9$	$< D > \ -- > \ < "9" >$

Πίνακας Β.1: Αρχείο Περιγραφής για τους Συντακτικούς Κανόνες της Γραμματικής Αριθμητικών Πράξεων G_{op}.

Το αυτόματο εργαλείο, αφού διαβάσει την περιγραφή της γραμματικής βγάζει μια σύντομη περιγραφή της μορφής της γραμματικής, δηλαδή το πλήθος των κανόνων, το πλήθος των τερματικών και μη τερματικών κανόνων καθώς και το μέγιστο αριθμό συμβόλων στο αριστερό μέλος των κανόνων. Βάσει αυτών των στοιχείων, γίνεται η αναπαράσταση σε δυαδικά ψηφία των δεδομένων καθώς και η δημιουργία των δυαδικών εξισώσεων. Όλα τα παραπάνω ενσωματώνονται μέσα στα εξαγόμενα αρχεία πηγαίου κώδικα Verilog, τα οποία είναι έτοιμα προς μεταγλώττιση.

B.2 Κωδικοποίηση Κατηγορικών Γραμματικών

Για τη περίπτωση των κατηγορικών γραμματικών, εκτός από τους συντακτικούς κανόνες πρέπει να δοθούν και οι σημασιολογικοί. Καθώς η υλοποίηση του κεφαλαίου 7 αποτελεί επέκταση εκείνης του κεφαλαίου 5, η μορφή εισόδου για τους συντακτικούς κανόνες παραμένει η ίδια. Όσον αφορά στους σημασιολογικούς κανόνες, παίρνουν την ακόλουθη μορφή: <κωδικός ενέργειας>, <όνομα προσωρινής μεταβλητής$_1$>, <όνομα προσωρινής μεταβλητής$_2$>, ..., <όνομα προσωρινής μεταβλητής$_n$>, <πράξη μεταξύ των n προσωρινών μεταβλητών>. Οι κωδικοί ενέργειας καθορίζουν τη μορφή του κανόνα, π.χ. εάν πρόκειται για μια απλή ανάθεση - δηλαδή μια εισαγωγή σε στοίβα, εάν πρόκειται για πράξη μεταξύ δεδομένων που θα εξαχθούν από κάποια στοίβα ή τέλος εάν πρόκειται για κανόνες που δε χρειάζεται καμία αλληλεπίδραση με τις στοίβες. Στον πίνακα B.2 παρουσιάζεται η κωδικοποίηση για την γραμματική G_{op} του κεφαλαίου 6 και στον πίνακα B.3 οι χρησιμοποιούμενοι κωδικοί ενέργειας. Υπενθυμίζεται ότι για τη συγκεκριμένη γραμματική γίνεται χρήση μόνο μιας στοίβας, σε πιο σύνθετα παραδείγματα με περισσότερες στοίβες, καθορίζεται και η στοίβα προέλευσης για κάθε προσωρινή μεταβλητή καθώς και η στοίβα προορισμού.

Συντακτικός Κανόνας	Σημασιολογικός Κανόνας	Σημασιολογικός Κανόνας στο Αρχείο Περιγραφής
$S \rightarrow E$	$S_s = E_s$	1
$E_1 \rightarrow T + E_2$	$E_{1s} = T_s + E_{2s}$	2, x, y, x+y
$E \rightarrow T$	$E_s = T_s$	0
$T \rightarrow F * T$	$T_s = F_s * T_s$	2, x, y, x+y
$T \rightarrow F$	$T_s = F_s$	0
$F \rightarrow (E)$	$F_s = E_s$	0
$F \rightarrow N$	$F_s = N_s$	0
$N \rightarrow DN$	$N_s = 10 * D_s + N_s$	2, x, y, x+10*y
$N \rightarrow D$	$N_s = D_s$	0
$D \rightarrow 0$	$D_s = 0$	3, 0
...	...	...
$D \rightarrow 9$	$D_s = 9$	3, 9

Πίνακας B.2: Αρχείο Περιγραφής για τους Συντακτικούς Κανόνες της Γραμματικής Αριθμητικών Πράξεων G_{op}.

Και εδώ, το αυτόματο εργαλείο, αφού διαβάσει την περιγραφή της γραμματικής παράγει τα εξαγόμενα αρχεία πηγαίου κώδικα Verilog, τα οποία είναι έτοιμα προς μεταγλώττιση.

Κωδικός Ενέργειας	Σημασία
0	Καμία ενέργεια δε χρειάζεται να γίνει στη στοίβα.
1	Εξαγωγή του τελικού κανόνα από τη στοίβα.
2	Εξαγωγή n δεδομένων από τη στοίβα και χρησιμοποίησή τους για αποτίμηση της πράξης, το αποτέλεσμα της όποιας επανεισάγεται στην στοίβα.
3	Απλή εισαγωγή της τιμής στη στοίβα.

Πίνακας Β.3: Επεξήγηση Κωδικών Ενεργειών για τη Γραμματική Αριθμητικών Πράξεων G_{op}.

Πίνακας Ελληνο-Αγγλικών Όρων

Action Grammar	Γραμματική Ενεργειών.
AG	Attribute Grammar -
	Κατηγορική γραμματική.
Ambiguous Grammar	Διφορούμενη Γραμματική.
Attribute	Κατηγόρημα
Bit Vector	Διάνυσμα Δυαδικών Ψηφίων.
CFG	Context Free Grammar -
	Γραμματική Χωρίς Συμφραζόμενα.
CNF	Chomsky Normal Form.
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.
MMU	memory management unit -
	Μονάδας Ελέγχου της Μνήμης.
Module	Λειτουργικό Τμήμα.
N_{LIDB}	Natural Language Interface to a Database -
	Διεπαφή Φυσικής Γλώσσας σε Βάση Δεδομένων.
Parse Table	Συντακτικός Πίνακας.
Parse Tree	Συντακτικό Δέντρο.
Parser	Συντακτικός Αναλυτής.
Pattern Grammar	Γραμματική Προτύπων.
Recognizer	Αναγνωριστής.
rhss	Right Hand Side Symbol -
	Δεξιό Μέλος του Κανόνα Παραγωγής.

Synthesized Attribute	Συντιθέμενο Κατηγόρημα.
SoC	System-on-a-chip - Συστήματος στο ίδιο τσιπ.
Unification Process	Διαδικασία Ενοποίησης.

Κατάλογος Εικόνων

Κατάλογος Πινάκων