ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Σχεδίαση Επιλογής Κινήσεων με Αναγνώριση Προτύπων για Παίκτη Go Βασισμένο σε Αναδιατασσόμενη Λογική

Φίλος Βασίλειος

Επιβλέπων

Καθηγητής Απόστολος Δόλλας (ΗΜΜΥ)

Εξεταστική Επιτροπή

Καθηγητής Διονύσιος Πνευματικάτος (ΗΜΜΥ)

Αναπλ. Καθ. Ιωάννης Παπαευσταθίου (ΗΜΜΥ)

Χανιά, Αύγουστος 2014

ΠΕΡΙΛΗΨΗ

Μετά την διάδοση και εξάπλωση του παιχνιδιού Go, ήταν φυσικό επόμενο ο κλάδος της Τεχνητής Νοημοσύνης να μελετήσει και να εξερευνήσει το παιχνίδι προσπαθώντας να δημιουργήσει Computer Programs, τα οποία προσομοιώνουν ένα παίχτη Go. Ο κλάδος της Τεχνητής Νοημοσύνης έχοντας καταγράψει σημαντική επιτυχία στην μοντελοποίηση άλλων Mind Games, δημιουργώντας Computer Programs πολύ ισχυρότερα από ένα φυσικό παίχτη επιδίωξε το ίδιο και με το Go.

Παρά την απλότητα των κανόνων αλλά και την διαδικασία παιχνιδιού, το Go είναι ένα αρκετά συνδυαστικό παιχνίδι, το οποίο μετρά μεγάλο αριθμό παικτών που κερδίζουν τα καλύτερα και ισχυρότερα Computer Go Programs. Η προσπάθεια αντιμετώπισης του παιχνιδιού με δενδρική αναζήτηση, όπως και σε άλλα Mind Games(Chess, Othello,..) είναι αδύνατη. Εξαιτίας της φύσης του παιχνιδιού η αναπαράσταση ενός δένδρου αναζήτησης είναι τεράστια.

Για το λόγο αυτό οι ερευνητές - επιστήμονες στράφηκαν στον προσδιορισμό Ευριστικών Κανόνων(Heuristics) και τακτικής γνώσης, για να επιλέγεται η επόμενη κίνηση - φύλλο(leaf node) του δένδρου, ώστε να μειωθεί το μέγεθός του, για να μπορεί να μελετηθεί και αξιοποιηθεί. Επιπλέον χρησιμοποιήθηκαν και διάφορες άλλες τεχνικές για να ενισχυθούν τα Computer Go Programs, όπως τροποποιημένες τεχνικές Επεξεργασίας Εικόνας, τεχνικές Αναγνώρισης Προτύπων και τεχνικές Εκπαίδευσης Προγραμμάτων.

Η μέθοδος, όμως, με τα καλύτερα δυνατά αποτελέσματα ήταν εκείνη του Monte Carlo Tree Search(M.C.T.S.). Ο συνδυασμός της δενδρικής αναζήτησης και προσομοιώσεων Monte Carlo, οι οποίες χρησιμοποιούνται για την αξιολόγηση κάθε πιθανής επόμενης κίνησης. Όσο περισσότερες προσομοιώσεις Monte Carlo, τόσο καλύτερη αξιολόγηση. Με στόχο αυτό, μοντελοποιήθηκε και δημιουργήθηκε ένα σύστημα με την χρήση FPGA ώστε να επιτευχθούν όχι μόνο γρήγορες και αξιόπιστες προσομοιώσεις αλλά και "χρήσιμες". Για το λόγο αυτό, το σύστημα, τροποποιήθηκε για να προσομοιώνει ένα παίχτη Go και ενισχύθηκε με κάποιους τακτικούς κανόνες και μια μονάδα Αναγνώρισης Προτύπων Κινήσεων για καλύτερη επιλογή επόμενων κινήσεων κατά την διάρκεια των προσομοιώσεων Monte Carlo.

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω θερμά τον κύριο Απόστολο Δόλλα, καθηγητή στη σχολή Ηλεκτρονικών Μηχανικών και Μηχανικών Υπολογιστών και επιβλέπων της παρούσας διπλωματικής εργασίας, πρώτα για την άψογη συνεργασία, τις συμβουλές και την καθοδήγηση που μου παρείχε σε όλη τη διάρκεια εκπόνησής της, όπως επίσης για τη συμβολή και επιμονή του να ασχοληθώ με το εξαιρετικό θέμα του Go Game.

Παράλληλα, θα ήθελα να ευχαριστήσω εξίσου και τους άλλους δύο καθηγητές του τμήματος, μέλη της εξεταστικής, επιτροπής. Είχα την τύχη και την εμπειρία στην προπτυχιακή μου σταδιοδρομία να αποκτήσω γνώσεις - εφόδια και από τα τρία μέλη της επιτροπής, όλες εξίσου χρήσιμες για την εκπόνηση της εργασίας και όχι μόνο.

Επιπλέον, ευχαριστώ όλο το προσωπικό του εργαστηρίου για όλες τις συμβουλές και την επιπλέον βοήθεια, όποτε αυτή χρειάστηκε καθ' όλη την διάρκεια φοίτησης στο ίδρυμα αλλά και εκπόνησης της διπλωματικής εργασίας.

Επίσης θα ήθελα να ευχαριστήσω τους γονείς μου για την στήριξη που μου παρείχαν σε όλα τα μαθητικά και φοιτητικά μου χρόνια. Ευχαριστώ τους υπέροχους φίλους - συμφοιτητές - συναδέλφους - συμπαίκτες - γείτονες, που είχα την τιμή να γνωρίσω και να συνεργαστώ όλα αυτά τα χρόνια στα Χανιά. Χωρίς αυτούς δεν θα είχα φτάσει ως εδώ. Πάνω από όλα όμως θα ήθελα να ευχαριστήσω τη φίλη μου Αργυρώ Λάσκαρη για την υπομονή της, την συμπαράστασή της και τις συμβουλές της σε οποιαδήποτε σημαντική απόφαση της ζωής μου, αλλά και τον "κολλητό" μου Χρήστο Θεοδωράκη για όλες τις σημαντικές και καθοριστικές συζητήσεις - στιγμές.

ΠΕΡΙΕΓΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

Κεφάλαιο 1

Εισαγωγή

Το παιχνίδι Go από την εμφάνιση του, πριν 40 περίπου αιώνες, μέχρι και σήμερα δεν παύει να προσελκύει κόσμο να ασχοληθεί με αυτό. Είναι ένα παιχνίδι με εξαιρετικά απλούς κανόνες για ένα νέο παίχτη ώστε να το μάθει, αλλά με αρκετά πολύπλοκη και σύνθετη λογική ώστε να καταφέρει κάποιος να γίνει αρκετά ανταγωνιστικός σε αυτό. Για αυτό, αποτελεί πρόκληση για τον οποιοδήποτε.

Έτσι λοιπόν, το παιχνίδι δεν άργησε να μπει και στον χώρο της επιστήμης και μάλιστα αυτό της Τεχνίτης Νοημοσύνης αποτελώντας σημείο ενδιαφέροντος πολλών Μαθηματικών και Μηχανικών Ηλεκτρονικών Υπολογιστών. Η πλήρης κατανόησή του και η μοντελοποίησή του σε μαθηματικά μοντέλα αρχικά και η μεταφορά του σε υπολογιστικά συστήματα μετέπειτα αποτελούσε στόχο για τους επιστήμονες. Καθημερινά, όλο και περισσότερα άτομα αποφασίζουν να ασχοληθούν με το συγκεκριμένο παιχνίδι εξερευνώντας καινούριους τομείς που ίσως βοηθήσουν. "Ευτυχώς", δεν έχει βρεθεί ακόμη ο ιδανικός τρόπος - μέθοδος ώστε να κατασκευαστεί ένα πρόγραμμα ανταγωνιστικό στους πρώτης κατηγορίας παίχτες του Go.

Τα δυνατότερα προγράμματα έχουν καταφέρει να φτάσουν το επίπεδο ενός μέτριου - μεσαίου παίχτη. Τα προγράμματα όμως, όπως και οι ενεργοί παίχτες έχουν φτάσει σε τέτοιο αριθμό ώστε κάθε χρόνο να διοργανώνονται διεθνείς διαγωνισμοί και για τα μεν αλλά και για τους δε. Το μαθηματικό μοντέλο του παιχνιδιού έχει σχεδόν εξαντληθεί ερευνητικά παράγοντας αρκετούς ευριστικούς κανόνες και τακτικές που χρησιμοποιούνται σε συνδυασμό με τις δενδρικές αναζητήσεις και αποτελούν το μυαλό και την σκέψη ενός τεχνητού παίχτη. Δυστυχώς όμως, λόγω της φύσης του παιχνιδιού μια αποδοτική δενδρική αναζήτηση σε όλο το εύρος του

πλαισίου παιχνιδιού του παιχνιδιού απαιτεί πολύ μεγάλο χώρο - μνήμη αλλά και πάρα πολύ επεξεργασία.

Ως εκ τούτου, στραφήκαν όλοι οι προγραμματιστές στην τυχαία προσομοίωση των καλών κινήσεων που προκύπτουν από τις μεμονωμένες αναζητήσεις σε περιοχές - κομμάτια του πλαισίου παιχνιδιού με μεθόδους Monte Carlo ώστε να τις αξιολογήσουν και να διαλέξουν τη καλύτερη. Μάλιστα, ο συνδυασμός Δενδρικών Αναζητήσεων και τεχνικών Monte Carlo έχει δώσει αρκετά υποσχόμενα αποτελέσματα αλλά και πολλές νίκες σε παγκόσμια πρωταθλήματα Computer Go σε προγράμματα, τα οποία χρησιμοποιούν αυτή την τεχνική. Όσο περισσότερες φορές προσομοιωθεί τυχαία μια κίνηση τόσο καλύτερο και αξιόπιστο αποτέλεσμα προκύπτει για την αξιολόγησή της. Για αυτό, είναι αρκετά σημαντικό να επιτευχτεί η ταχύτερη δυνατή προσομοίωση ενός παιχνιδιού μέχρι το τέλος χρησιμοποιώντας όλα τα δυνατά μέσα. Τελευταία, πολλοί προγραμματιστές θέτουν τα προγράμματά τους αντιμέτωπα με επαγγελματίες παίκτες υψηλού επιπέδου, έχοντας υποσχόμενα αποτελέσματα. Αυτό συμβαίνει με την βοήθεια πολυ - πύρηνων και πολυ νηματικών μηχανιμάτων, τα οποιά φτάνουν στο επίπεδο προσομοίωσης αρκετών χιλιάδων παιχνιδιών το δευτερόλεπτο για κάθε κίνηση εκμεταλλευόμενα την τεράστια επεξεργαστική τους ισχύ.

Οι όροι τυχαία προσομοίωση και τέλος παιχνιδιού, όμως, δημιουργούν κάποια προβλήματα αρκετά δύσκολα και απαιτητικά. Μια γεννήτρια τυχαίων αριθμών - συντεταγμένων αν ακολουθεί κανονική κατανομή όπως πρέπει ενδέχεται πολλές φορές να γεννήσει κατειλημμένες ή λάθος συντεταγμένες. Επίσης, σύμφωνα με τους κανόνες του παιχνιδιού μια θέση μπορεί μια δεδομένη στιγμή να είναι κατειλημμένη, έπειτα να ελευθερωθεί και ύστερα να ξανά - καταληφθεί και ούτω καθ' εξής. Όποτε και το τέλος του παιχνιδιού είναι λιγάκι δύσκολο και πολύπλοκο να καθοριστεί. Έτσι, η συγκεκριμένη διπλωματική εργασία ασχολήθηκε με την αναζήτηση του κατάλληλου συνδυασμού τεχνικών που αποτρέπουν άσκοπες κινήσεις και γρήγορης προσομοίωσης τυχαίων παιχνιδιών για την αξιολόγηση κάθε κίνησης.

1.1 Συνεισφορά της Διπλωματικής

Η ανάγκη, λοιπόν, για όσο το δυνατόν ταχύτερη υλοποίηση ενός τυχαίου παιχνιδιού αλλά και η ικανότητα των FPGA να εκτελούν συγκεκριμένες σχεδιάσεις, ανάλογα με τον τρόπο υλοποίησης τους αλλά και την πολυπλοκότητα τους, πολύ ταχύτερα από έναν Η/Υ με έναν επεξεργαστή "Γενικής Χρήσης" έστρεψε το ενδιαφέρον μου προς εκεί. Έπειτα από μελέτη και ερεύνα για την έως τώρα αντιμετώπιση του παιχνιδιού από την κοινότητα Computer Go αλλά και την καθοδήγηση και βοήθεια από προηγούμενη διπλωματική εργασία συμφοιτητή υλοποίησα μια δυναμική σχεδίαση η οποία προσομοιώνει ένα τυχαίο παιχνίδι μέχρι το τέλος.

Η υλοποίηση βελτιώνει την προηγούμενη δουλειά υλοποιώντας και προσομοιώνοντας ένα παίχτη Go, ο οποίος επιλέγει τυχαία δυο κινήσεις και αποφασίζει εκείνη με το καλύτερο σκορ στο τέλος προσομοίωσης Monte Carlo. Η επιλογή των δύο κινήσεων που θα ελεγχτούν αλλά και των τυχαίων κινήσεων κατά την προσομοίωση του παιχνιδιού στηρίζεται στους κανόνες του παιχνιδιού αποφεύγοντας κινήσεις αυτοκτονίας και αποδυνάμωσης της κατάστασης του παίχτη. Επιπλέον, έχει ενισχυθεί από μια βάση δεδομένων από πρότυπες κινήσεις μεγέθους 3x3.

Ουσιαστικά, σκοπός της διπλωματικής εργασίας ήταν η χρησιμοποίηση μιας προηγούμενης αρχιτεκτονικής τοποθέτησης πετρών στο πλαίσιο παιχνιδιού του Go με στόχο την δημιουργία ενός συστήματος, το οποίο θα προτείνει μια επόμενη κίνηση σε αυτή ενός αντιπάλου. Η μέθοδος που αποφασίστηκε για την αξιολόγηση των πιθανών κινήσεων - απαντήσεις ήταν εκείνη των τυχαίων προσομοιώσεων. Επειδή, όμως, η τυχαιότητα έχει το αντίκτυπο των αρκετών εσφαλμένων κινήσεων, όπως κινήσεις σε ήδη κατειλημμένες θέσεις, ή εκτός τον ορίων του παιχνιδιού, ή κινήσεις με τακτικά αρνητικό αποτέλεσμα, έγινε προσπάθεια ελάττωσης αυτού του προβλήματος. Εφαρμόζοντας, λοιπόν κάποιους κανόνες και κάποιες τεχνικές, οδηγήθηκε το σύστημα στην επιλογή των λιγότερο κακών κινήσεων, αλλά και στην κατευθυνόμενη επιλογή κάποιων πρότυπων κινήσεων. Τέλος, εφαρμόστηκε μια τεχνική αντιστοίχησης κάθε πιθανής "κίνησης" σε άλλες τρεις εναλλακτικές ώστε, να αποφεύγονται πιασμένες θέσεις και σημεία εκτός ορίων του πλαισίου. Για αυτό το λόγο η εργασία θα μπορούσε να χαρακτηριστεί "Design Space Exploration" μεταξύ της βασικής σχεδίασης, που είναι το πλαίσιο παιχνιδιού και της εφαρμογής τεχνικών και κανόνων, που ναι μεν αυξάνουν την πολυπλοκότητα αισθητά και με αντίκτυπο, αλλά καθιστούν την επιλογή κίνησης λιγότερο τυχαία - κακή και περισσότερο στοχευμένη.

1.2 Δομή της Διπλωματικής

Σημαντικό ρολό στην υλοποίηση της εργασίας έπαιξε η πλήρης κατανόηση του παιχνιδιού αλλά και η κατανόηση της μέχρι τώρα σχετικής έρευνας και ανάπτυξης του αντικειμένου στον τομέα του Computer Go.

Το 2^{ο} κεφάλαιο περιλαμβάνει μια μικρή αναφορά στους κανόνες του παιχνιδιού και μια παρουσίαση των τακτικών και μεθόδων που έχουν χρησιμοποιηθεί μέχρι στιγμής για την υλοποίηση προγραμμάτων Go.

Το 3^{ο} κατά σειρά κεφάλαιο περιλαμβάνει μια μικρή αναφορά στην μοντελοποίηση που μελετήθηκε από την προηγούμενη διπλωματική εργασία και παρουσιάζει τις εξελίξεις της, τα προβλήματα που πρόεκυψαν, τα bottlenecks και πως αντιμετωπίστηκαν. Επίσης στο τέλος του 3^{ου} κεφαλαίου παρουσιάζεται και το ανώτατο επίπεδο της αρχιτεκτονικής που υλοποιήθηκε.

Στο επόμενο 4^{ο} κεφάλαιο αναλύεται η σχεδίαση και η υλοποίηση της αρχιτεκτονικής με όλα τα απαιτούμενα Σχεδιαστικά Διαγράμματα(Block Diagrams) για την καλύτερη δυνατή κατανόηση και ανάλυσή της.

Το 5^{ο} κεφάλαιο περιλαμβάνει την επιβεβαίωση λειτουργίας της σχεδίασης και ότι έλαβε μέρος, ώστε να φτάσουμε στην ορθή λειτουργία αλλά και σχόλια για μερικές δυσλειτουργίες - προβλήματα και τα αποτελέσματα της σχεδίασης.

Τέλος, στο 6^{ο} κεφάλαιο εμπεριέχεται η μελλοντική και πιθανή ανάπτυξη και βελτίωση της υπάρχουσας αρχιτεκτονικής μέσω κάποιων προτάσεων.

Κεφάλαιο 2

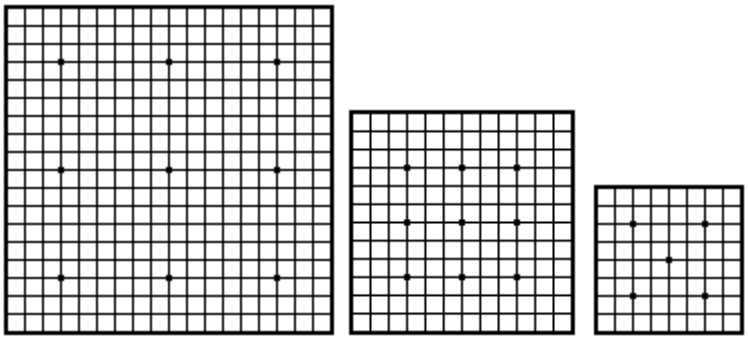
Σχετική Έρευνα

2.1 Το Παιχνίδι Go

2.1.1 Ιστορία και Εξέλιξη

Το παιχνίδι Go ξεκίνησε πριν από 4000 χρόνια κάπου μεταξύ Ιαπωνίας, Κίνας και Κορέας. Σήμερα στις χώρες αυτές είναι το πιο φημισμένο παιχνίδι και θεωρείται τιμή και κύρος κάποιος να είναι κάλος παίχτης Go. Κάθε χρόνο διοργανώνονται πρωταθλήματα και διαγωνισμοί ανάδειξης του καλύτερου παίχτη αλλά και του καλύτερου προγράμματος Η/Υ, το οποίο προσομοιώνει τον πιο δυνατό παίχτη.

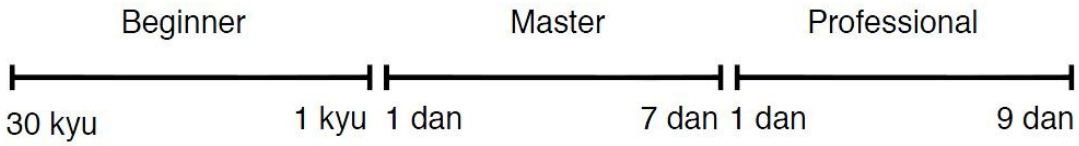
Η κλασσική έκδοση του παιχνιδιού αποτελείται από ένα πλαίσιο παιχνιδιού 19 καθέτων και 19 οριζοντίων γραμμών, οι οποίες δημιουργούν 361 σημεία στα οποία μπορεί να τοποθετηθεί πέτρα χρώματος μαύρου ή άσπρου. Εναλλακτικά, για παιχνίδια μικρότερης χρονικής διάρκειας και για πιο αρχάριους παίχτες χρησιμοποιούνται πλαίσια παιχνιδιού διαστάσεων 13x13 ή 9x9 αντίστοιχα.



εικόνα 1 : Πλαίσια παιχνιδιών 19x19 13x13 9x9, πηγή: British GO Association

2.1.2 Βαθμονόμηση Ικανότητας Παιχτών Go

Η βαθμονόμηση δύναμης και ικανότητας των παιχτών είναι η εξής:



εικόνα 2 : Βαθμονόμηση του επιπέδου των παικτών

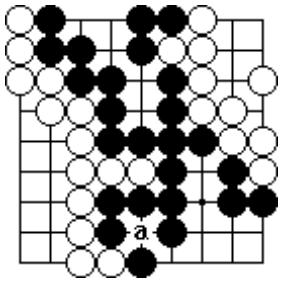
Όσο πιο δεξιά βρίσκεται κάποιος στην γραμμή τόσο πιο καλός παίχτης Go θεωρείται. Η βαθμολογία προκύπτει από τον αριθμό παιχνιδιών που έχει κερδίσει αλλά και το επίπεδο των αντιπάλων.

2.1.3 Κανόνες Παιχνιδιού Go

2.1.3.1 Γενικά

Το παιχνίδι ξεκινάει με το πλαίσιο παιχνιδιού να είναι άδειο. Ας θεωρήσουμε ότι υπάρχουν άπειρες πέτρες άσπρου και μαύρου χρώματος. Ο ένας παίχτης παίρνει τις άσπρες και ο άλλος τις μαύρες. Οι παίχτες παίζουν εναλλάξ ξεκινώντας πάντα αυτός που έχει το μαύρο χρώμα. Οι πέτρες τοποθετούνται στα σημεία τομής των γραμμών. Ο κύριος στόχος του παιχνιδιού είναι χρησιμοποιώντας τις πέτρες σου να καταλάβεις περιοχές στο πλαίσιο παιχνιδιού, επίσης υπάρχει η δυνατότητα να αιχμαλωτίσεις τις πέτρες του αντιπάλου αν τις περικυκλώσεις πλήρως. Οι πέτρες αυτές αφαιρούνται από το πλαίσιο παιχνιδιού και κρατούνται από τον παίχτη που τις αιχμαλώτισε για την μέτρηση του τελικού σκορ. Το παιχνίδι τελειώνει, όταν και οι δύο παίχτες συνεχόμενα πάνε "πάσο".

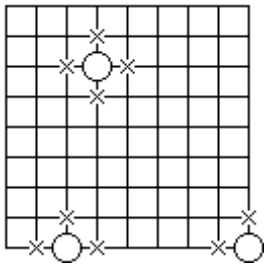
Στο τέλος του παιχνιδιού κάθε παίχτης μετρά τις πέτρες του χρώματός του που υπάρχουν στο πλαίσιο παιχνιδιού αλλά και τις πέτρες του αντιπάλου που έχει αιχμαλωτίσει. Κάθε μια πέτρα αντιστοιχεί σε ένα πόντο. Ο παίχτης με τους περισσότερους πόντους στο τέλος του παιχνιδιού είναι νικητής. Όπως φαίνεται και στην εικόνα **3** ο παίχτης με τις μαύρες πέτρες θα πρέπει να μετρήσει και έναν επιπλέον πόντο για την πέτρα του αντιπάλου που αιχμαλώτισε στο σημείο a.



εικόνα 3 : Πιθανή κατάσταση πλαισίου παιχνιδιού στο τέλος ενός παιχνιδιού, πηγή: British GO Association

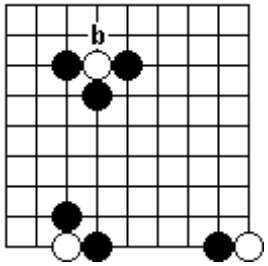
2.1.3.2 Αιχμαλώτιση(Capture) Πετρών και Αρίθμηση Ελευθεριών(Liberties)

Οι κενές διαθέσιμες θέσεις δυτικά, βόρεια, ανατολικά και νότια, που εφάπτονται με μια πέτρα τοποθετημένη στο πλαίσιο παιχνιδιού αποτελούν των αριθμό ελευθεριών της πέτρας. Όταν μια πέτρα αντίπαλου χρώματος τοποθετηθεί στο τελευταίο σημείο ελευθερίας μιας πέτρας, τότε εκείνη αιχμαλωτίζεται και αφαιρείται από το πλαίσιο παιχνιδιού. Στην εικόνα **4** φαίνονται τρεις άσπρες πέτρες με τέσσερις(4), τρεις(3) και δύο(2) ελευθερίες.



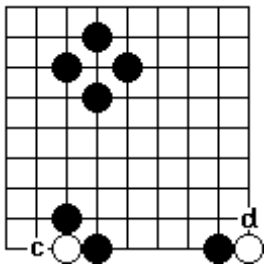
εικόνα 4 : Πέτρες με τέσσερις(4), τρεις(3) και δύο(2) ελευθερίες, πηγή: British GO Association

Στην επόμενη εικόνα **5** και οι τρεις άσπρες πέτρες έχουν από μια ελευθερία, εφόσον οι υπόλοιπες ελευθερίες έχουν καλυφτεί από πέτρες αντίπαλου χρώματος. Μάλιστα το σημείο b φανερώνει που πρέπει να τοποθετηθεί η επόμενη μαύρη πέτρα, ώστε να αιχμαλωτιστεί η άσπρη.



εικόνα 5 : Άσπρες πέτρες με μια μόνο ελευθερία, πηγή: British GO Association

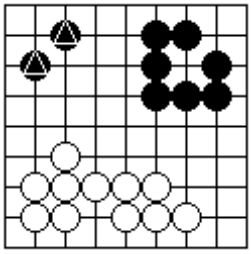
Η επόμενη εικόνα **6** απεικονίζει την κατάσταση του πλαισίου παιχνιδιού αν τοποθετηθεί μαύρη πέτρα στη θέση b και υποδεικνύει και τα άλλα δύο σημεία c, d, όπου αν τοποθετηθούν αντίστοιχα μαύρες πέτρες θα αιχμαλωτίσουν και τις άλλες δύο άσπρες.



εικόνα 6 : Αιχμαλώτιση μιας άσπρης πέτρας, πηγή: British GO Association

2.1.3.3 Ομάδα(Group) Πετρών

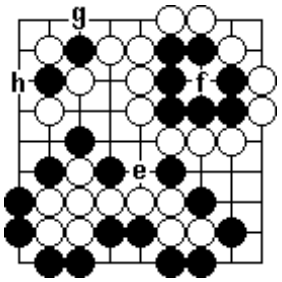
Ομάδα από πέτρες δημιουργούνται αν και μόνο αν πέτρες ίδιου χρώματος εφάπτονται στον οριζόντιο ή κατακόρυφο άξονα. Πέτρες ίδιου χρώματος, οι οποίες εφάπτονται διαγώνια δεν ανήκουν στην ίδια ομάδα. Όταν δημιουργείται μια ομάδα από πέτρες, εκείνη συμπεριφέρεται σαν μια μεγάλη πέτρα, η οποία μοιράζεται τις ελευθερίες των μικρότερων που την αποτελούν. Ένα σημείο στο πλαίσιο παιχνιδιού, που αποτελεί κοινό σημείο ελευθερίας δύο πετρών που ανήκουν στην ίδια ομάδα, προφανώς μετριέται σαν μια ελευθερία. Οι μαύρες πέτρες πάνω αριστερά της εικόνα **7** δεν αποτελούν ομάδα ενώ οι άσπρες κάτω και οι μαύρες δεξιά αποτελούν ομάδα.



εικόνα 7 : Ομάδες πετρών, πηγή: British GO Association

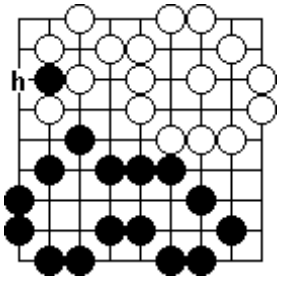
2.1.3.4 Αιχμαλώτιση(Capture) Ομάδας(Group) Πετρών

Εφόσον μια ομάδα πετρών συμπεριφέρεται σαν μια μεγάλη πέτρα, η διαδικασία που απαιτείται για την αιχμαλώτισή της είναι να καλυφτούν όλες οι ελευθερίες της. Η εικόνα **8** μας βοηθά να το καταλάβουμε υποδεικνύοντας τα σημεία e, f όπου αν τοποθετηθεί μαύρη και άσπρη πέτρα αντίστοιχα αιχμαλωτίζονται οι ομάδες που εμφανίζονται και στην εικόνα **7**. Επίσης τα σημεία g, h είναι εκείνα που αν τοποθετήσουμε άσπρη πέτρα, τότε αιχμαλωτίζονται και οι δύο μεμονωμένες πέτρες.



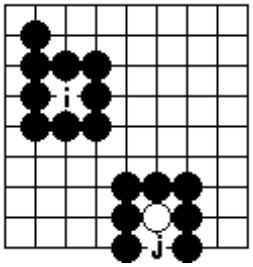
εικόνα 8 : Ένα βήμα πριν την αιχμαλώτιση ομάδας πετρών, πηγή: British GO Association

Τοποθετώντας, λοιπόν αντίπαλες πέτρες στα σημεία e, f, g το πλαίσιο παιχνιδιού θα μοιάζει κάπως έτσι:



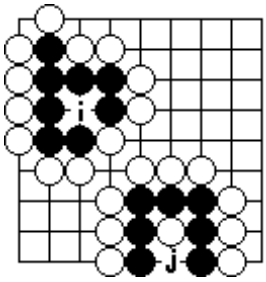
εικόνα 9 : Αιχμαλώτιση ομάδας πετρών, πηγή: British GO Association

Βέβαια, κατά την διάρκεια του παιχνιδιού μπορεί να συμβούν περίεργες καταστάσεις. Μια από αυτές είναι η δημιουργία ομάδας που μοιάζει με "δαχτυλίδι", η οποία απαγορεύει αυτόματα στον αντίπαλο να τοποθετήσει πέτρα στην μέση γιατί ουσιαστικά αυτό - αιχμαλωτίζει την πέτρα του, ή καλύπτει και το τελευταίο σημείο ελευθερίας της δικής του ομάδας. Αυτό φαίνεται παρακάτω στην εικόνα **10** όπου ο παίχτης με τα άσπρα δεν μπορεί να τοποθετήσει πέτρα στα σημεία i, j διότι είναι σαν να αυτοκτονεί δίνοντας πόντους στον αντίπαλο.



εικόνα 10 : Σημεία αυτοκτονίας, πηγή: British GO Association

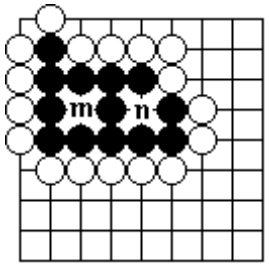
Όμως, οι δύο μαύρες ομάδες της παραπάνω εικόνας έχουν αυτή την δυνατότητα, διότι η τοποθέτηση της πέτρας στο ενδιάμεσο μέρος τους δεν καλύπτει και το τελευταίο σημείο ελευθερίας τους και ταυτόχρονα αποτελεί τοποθέτηση σε σημείο με καμία δυνατή ελευθερία(i), ή σημείο που μειώνει τις ελευθερίες της ομάδας που θα δημιουργηθεί στο 0. Αν, βέβαια, οι παραπάνω δύο μαύρες ομάδες ήταν περικυκλωμένες από άσπρες πέτρες έχοντας μοναδικό σημείο ελευθερίας το i και j αντίστοιχα, τότε ήταν καταδικασμένα. Τοποθετώντας ο αντίπαλος μια άσπρη πέτρα στα σημεία θα αιχμαλώτιζε τις δύο ομάδες.



εικόνα 11: Μετατροπή των σημείων αυτοκτονίας σε σημεία ισχύος, πηγή: British GO Association

2.1.3.5 Life and Death, Eyes

Με τον όρο Eye εννοούμε το σημείο εκείνο το οποίο δεν περιέχει πέτρα, αλλά περιμετρικά του (δυτικά, βόρεια, ανατολικά, νότια) βρίσκονται πέτρες που ανήκουν στο ίδιο χρώμα. Ένας στόχος των έμπειρων και ικανών παιχτών του Go είναι να δημιουργήσουν ομάδες με 2 Eyes, γιατί τέτοιες ομάδες παραμένουν μονίμως ζωντανές ότι και να γίνει. Όπως, είναι φανερό και στην εικόνα **12**

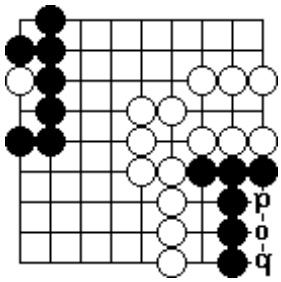


εικόνα 12 : Ομάδα με δύο Eyes, πηγή: British GO Association

η ομάδα των μαύρων πετρών δε αιχμαλωτίζεται ποτέ.

Ο αντίπαλος το έχει περικυκλώσει πλήρως όμως αδυνατεί να τοποθετήσει πέτρα στα m, n. Αφενός, δεν αποτελεί κίνηση που εκμηδενίζει τις ελευθερίες του αντιπάλου, αφετέρου δεν μπορεί να μείνει στο πλαίσιο παιχνιδιού (και να παιχτεί στον επόμενο γύρο η άλλη πέτρα), διότι άμεσα αιχμαλωτίζεται αφού δεν έχει καμία ελευθερία. Αν κάποια ομάδα έχει την δυνατότητα να δημιουργήσει δύο Eyes με την τοποθέτηση μιας και μόνο πέτρας τότε η "βιωσιμότητα" της εξαρτάται από το ποιος θα παίξει πρώτος.

Η κατάσταση αυτή ονομάζεται Life and Death. Στην παρακάτω εικόνα **13** περιγράφεται μια τέτοια κατάσταση.

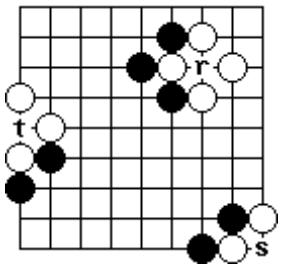


εικόνα 13 : Life and Death κατάσταση, πηγή: British GO Association

Ο πρώτος παίχτης που θα τοποθετήσει πέτρα στο σημείο ο είναι εκείνος που θα καθορίσει αν θα δημιουργηθεί ομάδα με δύο Eyes ή όχι. Αν ο παίχτης με τα μαύρα παίζει πρώτος στο σημείο ο τότε η ομάδα δεν υπάρχει περίπτωση να αιχμαλωτιστεί. Σε αντίθετη περίπτωση αν τοποθετηθεί άσπρη πέτρα σε εκείνο το σημείο τότε η ομάδα θα αιχμαλωτιστεί σίγουρα. Στην εικόνα **13** απεικονίζεται και μια ακόμη ομάδα μαύρων πετρών πάνω αριστερά, η οποία έχει εξασφαλίσει την ύπαρξη της στην συνέχεια του παιχνιδιού.

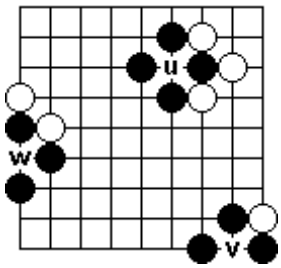
2.1.3.6 Ο Κανόνας ko

Ο κανόνας αυτός εξασφαλίζει την μη ύπαρξη κινήσεων, οι οποίες μπορεί να αποδώσουν μια ανούσια επαναληψιμότητα στο παιχνίδι, η οποία χωρίς τον κανόνα μπορεί να καταλήξει ατέρμονη. Η εικόνα **14** παρουσιάζει ένα ενδεχόμενο πλαίσιο παιχνιδιού στο οποίο η τοποθέτηση μαύρης πέτρας στα σημεία r, s, t θα επιφέρει την αιχμαλώτιση άσπρων πετρών.



εικόνα 14 : Κανόνας ko 1, πηγή: British GO Association

Έστω, λοιπόν ότι τοποθετείται μαύρη πέτρα σε κάποιο από τα σημεία και το πλαίσιο παιχνιδιού μετά έχει αυτή την μορφή(απεικονίζεται η αλλαγή που θα είχε το πλαίσιο παιχνιδιού σε κάθε σημείο r, s, t τοποθέτησης μαύρης πέτρας):

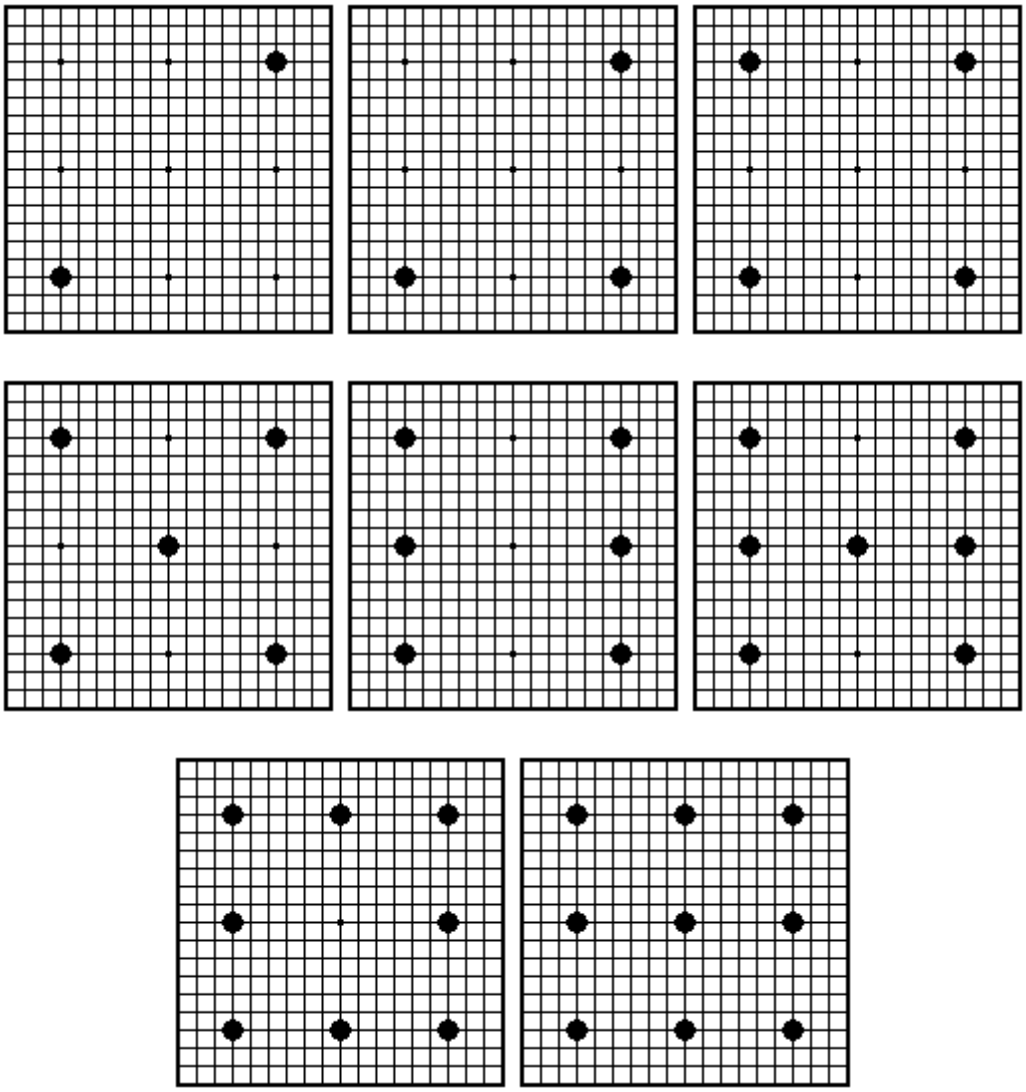


εικόνα 15 : Κανόνας ko 2, πηγή: British GO Association

Είναι εμφανές ότι μπορεί τώρα ο άσπρος παίχτης με την σειρά του να παίξει σε κάποιο από τα σημεία u, v, w και να επαναφέρει το πλαίσιο παιχνιδιού σε μια πρωτύτερη κατάσταση. Αυτό απαγορεύεται και είναι ο πιο "πολύπλοκος" κανόνας του Go.

2.1.3.7 Handicap

Ο κανόνας αυτός είναι εναλλακτικός του επόμενου και τελευταίου κανόνα και δείχνει τον τρόπο(φαίνεται στην εικόνα **16**) με τον οποίο μπορούν να τοποθετηθούν από δύο μέχρι και εννέα πέτρες πριν ξεκινήσει το παιχνίδι για τον παίχτη που είναι ο πιο αδύναμος. Όσο πιο αδύναμος είναι ο παίχτης τόσες περισσότερες πέτρες έχει δικαίωμα να τοποθετήσει στο πλαίσιο παιχνιδιού πριν ξεκινήσει το παιχνίδι.



εικόνα 16 : Handicap, πηγή: British GO Association

2.1.3.8 Komi

Ο τελευταίος κανόνας καταδεικνύει την μεγαλειότητα αλλά και την πολυπλοκότητα του παιχνιδιού παρά τους εύκολους κανόνες. Komi ονομάζονται οι αρχικοί πόντοι τους οποίους παίρνει ο παίκτης με τις άσπρες πέτρες γιατί απλά δεν ξεκάνει πρώτος. Συνήθως, όταν παίζουν δύο παίχτες ίδιου επιπέδου εκείνος με τις άσπρες πέτρες έχει ένα προβάδισμα 7 πόντων λόγω της κρισιμότητας του πλεονεκτήματος να παίζει πρώτος ο αντίπαλος.

2.2 Computer Go

2.2.1 Εισαγωγή

Με την εμφάνιση της Τεχνητής Νοημοσύνης στον τομέα της επιστήμης υπολογιστών, ένα από τα πρώτα πεδία, το οποίο τέθηκε υπό έρευνα και εξερεύνηση ήταν αυτό των Mind Games. Σήμερα, πολλά προγράμματα, τα οποία προσομοιώνουν παίχτες των παραπάνω παιχνιδιών εμφανίζουν αποτελέσματα άκρως ελκυστικά όταν αντιμετωπίζουν έναν κοινό άνθρωπο - παίχτη επιβεβαιώνοντας την αποδοτικότητα των μεθόδων της Τεχνητής Νοημοσύνης. Οι αλγόριθμοι MinMax και Alpha - Beta είναι κύριο χαρακτηριστικό της πλειοψηφίας των προγραμμάτων. Είναι το σημείο από το οποίο οι περισσότεροι προγραμματιστές ξεκινούν τις προσπάθειες τους.

Το Go -όμως- δεν συγκαταλέγεται στη πλειοψηφία των παιχνιδιών, μη έχοντας καταφέρει κάποιος να εντοπίσει το σαφώς πιο πολύπλοκο και δύσκολο cornerstone των προγραμμάτων, που υλοποιούν τον Go Player.

2.2.2 Χαρακτηριστικά του Παιχνιδιού Go

Το παιχνίδι Go είναι ένα συνδυαστικό παιχνίδι(combinational game) με τα εξής χαρακτηριστικά:

- **two - person:** Παιχνίδι το οποίο παίζεται από δύο παίχτες.
- **zero - sum:** Οι πόντοι που κερδίζει ή χάνει ένας παίχτης είναι ακριβώς εκείνοι που χάνει ή κερδίζει αντίστοιχα ο αντίπαλος.
- **finite:** Το παιχνίδι τελειώνει έπειτα από συγκεκριμένο αριθμό κινήσεων ασχέτως το πως παίζεται.
- **perfect information:** Οι παίχτες είναι γνώστες της πλήρους κατάστασης του πλαισίου παιχνιδιού και των εναπομείναντων διαθέσιμων κινήσεων.

2.2.3 Πολυπλοκότητα του Go

Μελετώντας την πολυπλοκότητα των παιχνιδιών της κατηγορίας, στην οποία ανήκει και το Go όσο αφορά τον αριθμό των δυνατών θέσεων στη αρχή του παιχνιδιού (E) και τον ελάχιστο αριθμό κόμβων που μπορεί να έχει ένα δένδρο για την επίλυση του παιχνιδιού (A) προκύπτουν 2 εξαιρετικά σημαντικοί πίνακες.

Game	$\log_{10}(E)$	$\log_{10}(A)$	Computer - Human results
Checkers	17	32	Chinook > H
Othello	30	58	Logistello > H
Chess	50	123	Deep Blue >= H
Go	160	400	Handtalk << H

εικόνα 17 : Πίνακας 1 πληροφοριών πολυπλοκότητας Mind Games

Game	$\log_{10}(E)$	$\log_{10}(A)$	Computer - Human results
Checkers	17	32	Chinook > H
Othello	30	58	Logistello > H
9x9 Go	40	85	Strongest Go Program << H
Chess	50	123	Deep Blue >= H
19x19 Go	160	400	Strongest Go Program << H

εικόνα 18 : Πίνακας 2 πληροφοριών πολυπλοκότητας Mind Games

Αξιοσημείωτο είναι ότι παρόλο ότι το 9x9 Go είναι λιγότερο πολύπλοκο από το σκάκι, το ικανότερο Go program είναι πολύ κατώτερο από έναν άνθρωπο σε αντίθεση με το Deep Blue.

2.2.4 Διαφορετικότητα και Απαιτήσεις του Go

Όλα τα παραπάνω παιχνίδια -όπως και το σκάκι- έχουν ένα κοινό χαρακτηριστικό, όσο αφορά την προσομοίωση παίχτη τους με την χρήση Η/Υ και υλοποίηση προγραμμάτων - συγκεκριμένων δομών δεδομένων. Κυρίαρχο χαρακτηριστικό είναι η εξάρτηση αυτών των προγραμμάτων από Μεγάλου Εύρους(Large Scale) Δενδρικές Αναζητήσεις και όχι από την γνώση ή τον τρόπο ανάπτυξης του παιχνιδιού. Έτσι το χαρακτηριστικό αυτό καθίσταται κοινό, εφόσον η διαφορετικότητα του κάθε παιχνιδιού επηρεάζει στο ελάχιστο το κάθε πρόγραμμα, εφόσον εμφανίζεται μόνο στην τήρηση των κανόνων του κάθε παιχνιδιού.

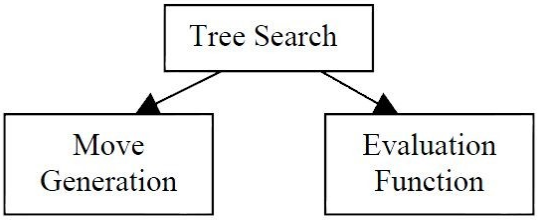
Εν αντιθέσει, στο παιχνίδι Go το μείζον και επιτακτικό ζήτημα είναι η υλοποίηση ενός προγράμματος, το οποίο να μπορεί να συμπεριφερθεί όσο το δυνατόν περισσότερο σαν άνθρωπος, να έχει γνώση του παιχνιδιού και του σωστού τρόπου παιξίματος, όσο αυτό είναι δυνατόν. Αυτό συμβαίνει αφενός λόγω της δομής του παιχνιδιού, η οποία απαιτεί ένα γιγαντιώδες δένδρο αναζήτησης και αφετέρου διότι η παικτική λογική ακόμη και ενός αρχάριου αντιπάλου είναι αρκετά πιο σύνθετη ακόμη και από Brute Force λογικές, για τις οποίες οι Η/Υ είναι ιδανικοί.

Έτσι, το παιχνίδι Go υπήρξε ένα πολύ ενδιαφέρον έναυσμα αλλά και ένας σημαντικός αρωγός για την οποιαδήποτε εξέλιξη του κλάδου της Τεχνητής Νοημοσύνης στον τομέα των παιχνιδιών την τελευταία τριακονταετία. Ο συνδυασμός της Τεχνητής Νοημοσύνης με την ανάγκη των προγραμμάτων για απόκτηση γνώσης σχετικά με το παιχνίδι οδήγησε την πλειοψηφία προγραμμάτων που προσομοιώνουν ένα παίχτη Go να χρησιμοποιούν τεχνικές όπως Αναγνώρισης Προτύπων(Pattern Recognition) και Εκμάθησης Μηχανών(Machine Learning) και άλλες.

2.2.5 Βασικά Χαρακτηριστικά Προγραμμάτων Go

Κατά γενική ομολογία αλλά και παρατήρηση των προγραμμάτων, που έχουν δημιουργηθεί μέχρι σήμερα παρατηρείται πως αποτελούνται από τις δομές:

- **Δενδρική Αναζήτηση**(Tree Search)
- **Συνάρτηση Αξιολόγηση**(Evaluation Function)
- **Γεννήτρια Κίνησης**(Move Generation)



εικόνα 19 : Tree Search, Move Generation, Evaluation Function, πηγή: [9]

τις οποίες προσπαθούν συνεχώς να εξελίξουν και θα αναλυθούν παρακάτω.

Πολλοί ερευνητές στηρίζουν τις προσπάθειες τους στον ιδανικό συνδυασμό Δενδρικής Αναζήτησης με Συνάρτηση Αξιολόγησης και Γεννήτρια Κίνησης αντίστοιχα. Η δενδρική αναζήτηση προσφέρει σημαντική βοήθεια στην αξιολόγηση κάποιας κίνησης αλλά και στους τακτικούς υπολογισμούς, που απαιτούνται για την δημιουργία κάθε νέας κίνησης. Ως εκ τούτου, μοντέλα με περισσότερη γνωστική ικανότητα(Knowledge) Go, δηλαδή τακτική και αντίληψη στηριζόμενη όχι μόνο στους κανόνες αλλά και στον τρόπο παιχνιδιού του αντιπάλου εμφανίζουν τα καλύτερα δυνατά αποτελέσματα. Συνεπώς, τεχνικές και μέθοδοι όπως Εκμάθηση Μηχανών(Machine Learning) και Αυτοματοποιημένη Ανάπτυξη Γνωστικής Ικανότητας(Automatic Generation of Knowledge) καθίστανται άκρως ελκυστικές[7], [16].

Μέχρι και σήμερα προγράμματα, που χρησιμοποιούν τον παραπάνω αυτοματοποιημένο τρόπο για την εξέλιξη τους στηρίζονται κατά κόρον σε τρεις μεθόδους:

- Temporal Difference Method
- Retrograde Analysis
- Logic Metaprogramming

2.2.5.1 Συνάρτηση Αξιολόγησης(EF)

Η αξιολόγηση μιας θέσης είναι αναγκαία σε κάθε πρόγραμμα, το οποίο θέλει να συσχετίσει το σκορ με το αντίστοιχο παιχνίδι. Η εύρεση μιας ιδανικής ή έστω ικανοποιητικής Evaluation Function είναι πολύ μεγάλο, αν όχι το μεγαλύτερο,

εμπόδιο του Computer Go[7]. Πλέον η αξιολόγηση κάθε θέσης συνίσταται από δύο μέρη - ειδή:

- Concrete EF
- Conceptual EF

Η πρώτη είναι συγκεκριμένη και κοινή σε όλα σχεδόν τα προγράμματα. Αφορά την μετάφραση σε αριθμό κάθε θέσης ανάλογα με τι βρίσκεται σε αυτή, 1 για άσπρο, -1 για μαύρο και 0 για κενό παραδείγματος χάρη. Η χρήση της Conceptual EF, η οποία είναι διαφορετική για κάθε πρόγραμμα, παρέχει πληροφορίες σχετικά με την επιρροή κάθε ομάδας πετρών, την Life or Death κατάστασή της, την δυνατότητα να ενωθεί με γειτονικές ομάδες και άλλες. Ουσιαστικά η Conceptual EF κυρίως με χρήση Δενδρικής Αναζήτησης εφαρμόζει την γνώση σχετικά με το Go που έχει εντάξει ο κάθε προγραμματιστής στο πρόγραμμα του.

Από την άλλη, σε προγράμματα, τα οποία δεν χρησιμοποιούν σχεδόν καθόλου γνωστικό υπόβαθρο(Knowledge) Go το παιχνίδι μετατρέπεται σε Incomplete Information Game και δεσπόζει η τεχνική Monte Carlo με παραπάνω από θετικά αποτελέσματα συγκρινόμενη με γνωστικά προγράμματα Go(Go Knowledge Programmes).

2.2.5.2 Γεννήτρια Κίνησης(MG)

Η ύπαρξη της μονάδας γέννησης κίνησης είναι άρρηκτα συνδεδεμένη με αυτή των EF και TS[7]. Η Συνάρτηση Αξιολόγησης χρειάζεται για τον έγκαιρο τερματισμό του παιχνιδιού όταν αυτό φτάσει στο τέλος και η μονάδα Δενδρικής Αναζήτησης για παροχή τακτικών πληροφοριών σχετικά με κάποιο συνολικό(global) ή τοπικό(local) στόχο(goal) ανάλογα.

Ακλουθώντας την εξέλιξη των άλλων δύο μονάδων η Γεννήτρια Κίνησης συμπεριέλαβε και μια ακόμη μονάδα, την Γεννήτρια Στόχου(Goal Generation). Χωρίς να τροποποιήσει την ανατροφοδότηση πληροφοριών(feedback), που λάμβανε από τις άλλες 2 μονάδες, ανάλογα με την ροή του παιχνιδιού γεννούσε κινήσεις κατάλληλες είτε για την επέκταση μιας ομάδας πετρών, ή για την προστασία κάποιας άλλης, είτε για την πραγματοποίηση επίθεσης σε μια συγκεκριμένη ομάδα του αντιπάλου, ή μια συγκεκριμένη περιοχή του πλαισίου παιχνιδιού. Προφανώς, η μονάδα δημιουργίας στόχου μπορεί να αποτελείται από υπό - μονάδες όπως Γεννήτρια Κίνησης Συνολικού Στόχου(Global Goal MG), Γεννήτρια Κίνησης Επιμέρους - Τοπικού Στόχου (Local Sub Games MG) και ακόμη πολλά, αναλόγως τις ανάγκες κάθε υλοποιημένου προγράμματος.

2.2.5.3 Δενδρική Αναζήτηση(TS)

Ο στόχος της δενδρικής αναζήτησης σε παιχνίδια όπως το Go είναι η εύρεση μιας κίνησης μεταξύ πολλών χρησιμοποιώντας την EF σαν ένα "μαύρο κουτί"[7]. Αλγόριθμοι όπως ο MinMax και Alpha - Beta κατέχουν εξέχουσα θέση στην κατηγορία αυτή. Βέβαια όσο αφορά το ασιατικό παιχνίδι τα πράγματα είναι λιγάκι διαφορετικά λόγω του Locality αλλά και του Selectivity, των οποίων ο σωστός ορισμός και καθορισμός είναι κομβικής σημασίας για το παιχνίδι.

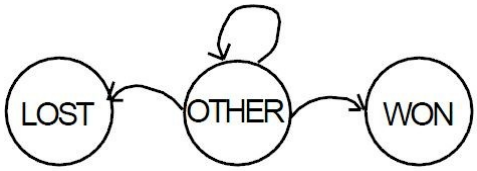
Βέβαια, αυτό που παρατηρείται είναι διαφορετική χρήση αλλά και απόδοση βαρύτητας σε μεθόδους Δενδρικής Αναζήτησης από πρόγραμμα σε πρόγραμμα. Το HandTalk δημιουργεί μόνο 4 πιθανές κινήσεις στο ανώτερο - συνολικό(global) επίπεδο έτσι ώστε να επιτύχει μικρές Δενδρικές Αναζητήσεις, υποστηρίζοντας πως υπάρχουν άλλα πράγματα μεγαλύτερης βαρύτητας που επηρεάζουν το πρόγραμμα. Αντιθέτως, το Many Faces of Go διαθέτει διαφορετικές TS για δεδομένους υποστόχους όπως αιχμαλώτισης μια ομάδας πετρών, σύνδεσης δύο ομάδων, σχηματισμού ομάδας κατάστασης Dead or Alive και άλλα. Την πολυπλοκότητα των TS του Many Faces of Go δεν ενστερνίζεται το Go4++ το οποίο δημιουργεί απλά 40 κινήσεις και καταλήγει να παίζει αυτή με τη μεγαλύτερη τιμή, που προκύπτει από την Συνάρτηση Αξιολόγησης. Τέλος το Go Intellect φαίνεται να είναι το πιο ισορροπημένο από όλα, διότι χρησιμοποιεί επιλεκτικές(selective) ολικές(global) Δενδρικές Αναζητήσεις. Δηλαδή στηρίζει τις αναζητήσεις του σε κάποιους ευριστικούς κανόνες, οι οποίοι είναι και υπεύθυνοι να τις σταματήσουν όταν κάνουν κάποια ταύτιση(match), ή όταν κάποια επιθυμητή τιμή επιτευχτεί.

Δεδομένου του γεγονότος ότι σε μια θέση του πλαισίου παιχνιδιού Go μπορεί να επανατοποθετηθεί πέτρα παραπάνω από μια φορές αυτό καθιστά τις Global Searches τύπου Brute - Force άχρηστες ασχέτως υπολογιστικής ισχύος. Για αυτό αλλά και για την πιο ρεαλιστική προσομοίωση ενός φυσικού παίχτη οι περισσότεροι προγραμματιστές επιλέγουν τοπικές Δενδρικές Αναζητήσεις ή γενικές Δενδρικές Αναζητήσεις βάσει κάποιου στόχου.

Το μεγάλο πρόβλημα, που προκύπτει σχετικά με τις τοπικές αναζητήσεις σε υποπαιχνίδια είναι ο τεκμηριωμένα ιδανικός χώρος αναζήτησης. Τι καθιστά μια αναζήτηση τοπική; Πόσο πρέπει να είναι το εύρος της αναζήτησης; Πότε πρέπει να σταματά; Είναι όλα ερωτήματα που δύσκολα απαντούνται αλλά και δύσκολα θα απαντηθούν. Οπότε πολλά προγράμματα έχουν καταλήξει απλά στο να διασπούν την γενική Δενδρική Αναζήτηση βάσει κάποιου στόχου σε "βέλτιστες" τοπικές Δενδρικές Αναζητήσεις βάσει κάποιου στόχου.

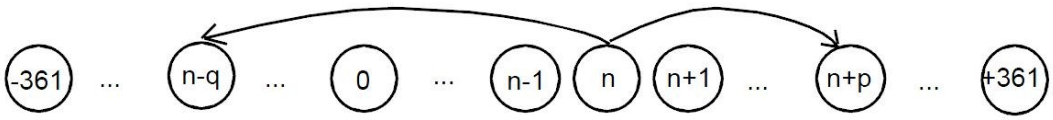
2.2.5.4 Μέθοδος Εξέλιξης των Τοπικών Δενδρικών Αναζητήσεων

Η εφαρμογή και χρησιμοποίηση των τοπικών Δενδρικών Αναζητήσεων για των προσδιορισμό κινήσεων, που αιχμαλωτίζουν μια ομάδα πετρών, ή συνδέουν δύο ομάδες είχαν αρχικά ένα 3 - κατάστατο αποτέλεσμα[4].



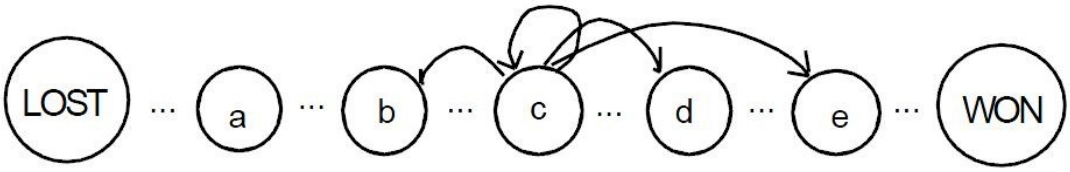
εικόνα 20 : Δυνατές εκβάσεις τακτικών Local Tree Search, πηγή: [4]

Τα αποτελέσματα από τις εκάστοτε αναζητήσεις καθόριζαν και τις κινήσεις που θα επιλέγονταν αργότερα ώστε να ελεγχθεί η επιρροή τους στο συνολικό - γενικό στόχο. Με αυτό τον τρόπο λοιπόν θα απορρίπτονταν μια κίνηση, που έχει αρνητικό αντίκτυπο στο συνολικό - γενικό στόχο ενώ για το υπο-παιχνίδι είναι ιδανική, ή το ανάποδο μια κίνηση με εξαιρετικό αποτέλεσμα στο συνολικό - γενικό στόχο δεν θα ελέγχονταν ποτέ. Έτσι, ο Bouzy πρότεινε μια τεχνική λιγάκι διαφορετική προσπαθώντας να δώσει λύση. Πρότεινε κάθε κίνηση που θα ελέγχονταν από τις τοπικές Δενδρικές Αναζητήσεις να έχει σαν αποτέλεσμα και την επιρροή που έχει απέναντι στο συνολικό σκορ.



εικόνα 21 : Δυνατές εκβάσεις τακτικών Local τοπικών Δενδρικών Αναζητήσεων σε σχέση με το σκορ σύμφωνα με τον Bouzy, πηγή: [4]

Η τεχνική αυτή έδινε λύση στο αρχικό πρόβλημα, δημιουργούσε όμως κάποια άλλα. Η αντιμετώπιση του τελευταίου προβλήματος περιλαμβάνει την ύπαρξη μεταβλητού αριθμού ενδιάμεσων καταστάσεων μεταξύ Win και Loss για το συνολικό - γενικό στόχο, ώστε να γίνονται αποδεκτές κινήσεις που οδηγούν στην κατάσταση Win ή κυμαίνονται γύρω από την ενδιάμεση κατάσταση Other(η οποία συνήθως είναι η μεσαία) και να απορρίπτονται μόνο ακραίες "ιδανικές" κινήσεις.



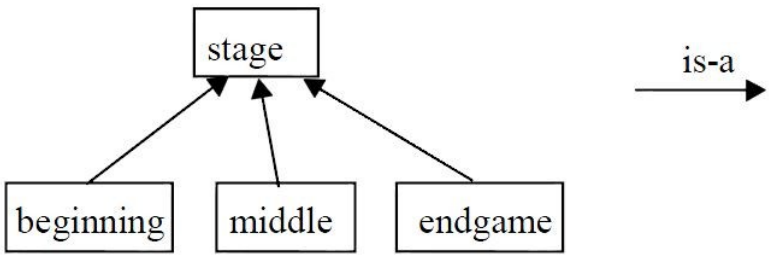
εικόνα 22 : Δυνατές καταστάσεις μετάβασης του συνολικού - γενικού στόχου σύμφωνα με τον Bouzy, πηγή: [4]

Η μεταβλητότητα στον αριθμό των ενδιάμεσων θέσεων σχετίζεται με το αντίκτυπο στο σκορ που μπορεί να έχει μια θέση αλλά και το σημείο του παιχνιδιού στο οποίο βρισκόμαστε. Ο αριθμός είναι αντιστρόφως ανάλογος στον αριθμό κινήσεων που έχουν πραγματοποιηθεί αλλά δεν μπορεί να επιλεγεί ούτε πειραματικά διότι εξαρτάται και από τους κανόνες και την τακτική κάθε προγράμματος.

2.2.6 Αντικειμενοστραφής Ανάλυση του Go

Η συντριπτική πλειοψηφία των προγραμμάτων Go χρησιμοποιεί κάποια κοινά και συγκεκριμένα πλαίσια [5] τακτικών για την απλοποίηση αλλά και την βελτίωση της Συνάρτησης Αξιολόγησης. Όσο αφορά το χρονικό πλαίσιο χωρίζουν το παιχνίδι σε 3 χρονικές περιόδους:

- Την **αρχική,** που διαρκεί το πολύ μέχρι την $50^{η}$ πρώτη κίνηση. Σε αυτή την περίοδο το πρόγραμμα παίζει τυχαία γεμίζοντας τις άδειες θέσεις του πλαισίου παιχνιδιού.
- Την **ενδιάμεση** περίοδο, όπου το πρόγραμμα φροντίζει να επιτεθεί σε αντίπαλες ομάδες αρκετών πετρών ή αμύνεται των δικών του "μεγάλων" ομάδων.
- Την **τελευταία** περίοδο, όπου απλά φροντίζει να καλύψει τα τελευταία κενά.

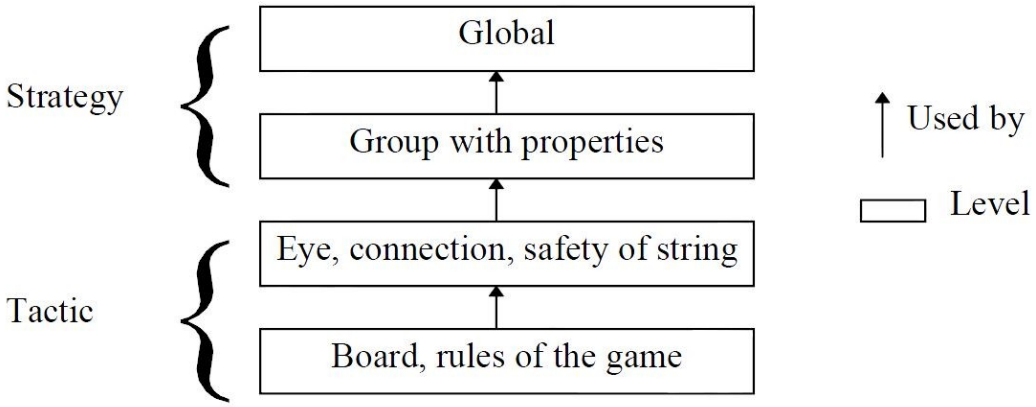


εικόνα 23 : Χρονικό πλαίσιο διαχωρισμού των παιχνιδιών, πηγή: [5]

Ένα ακόμη χρήσιμο πλαίσιο είναι αυτό του στόχου, το οποίο είναι υπεύθυνο για την απόρριψη κινήσεων που έχουν θετικό αντίκτυπο σε υπο-παιχνίδια και όχι συνολικά. Επιπροσθέτως, το χωρικό περιθώριο επιστρέφει στο συνολικό σύστημα πληροφορίες σχετικά με τον βαθμό επιρροής κάθε ομάδας πετρών ή πέτρας.

Τέλος το ολικό πλαίσιο συνενώνει όλα τα παραπάνω εκμεταλλευόμενο και την πληροφορία του σκορ. Για παράδειγμα τροφοδοτεί το σύστημα με συνδυαστικές πληροφορίες όπως : ο αντίπαλος προηγείται αλλά ελάχιστα, ή ο αντίπαλος ελέγχει περισσότερη περιοχή, ή ο αντίπαλος έχει περισσότερες ομάδες ή η επιρροή των αντιπάλων ομάδων είναι πιο ισχυρή.

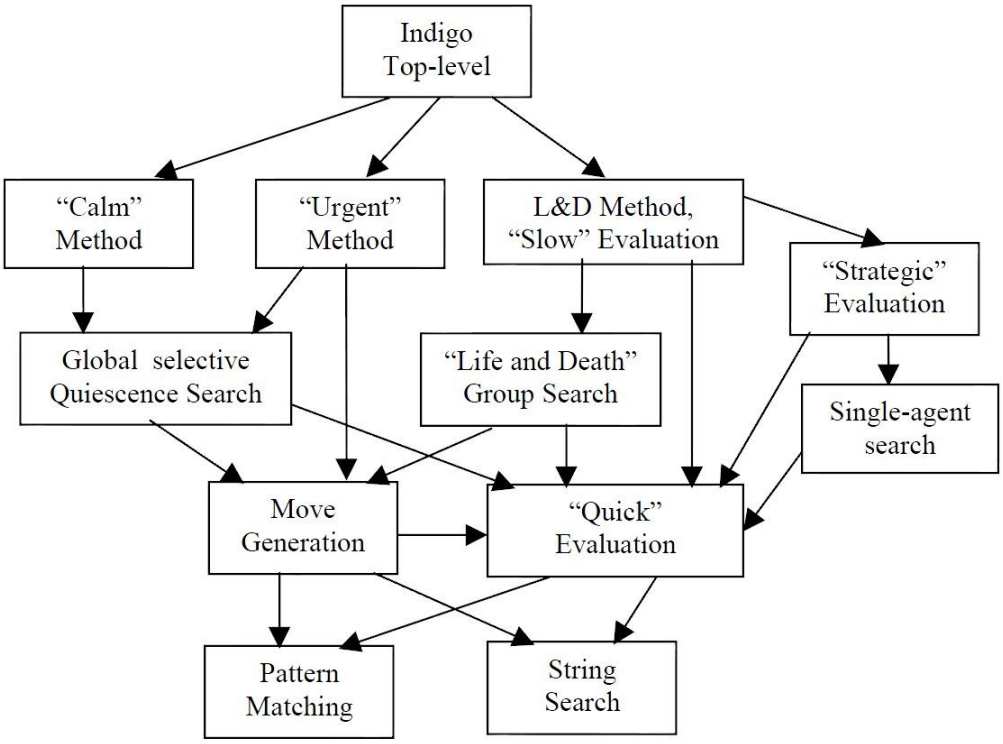
Συνεπώς προκύπτει μια υποτυπώδης ιεραρχική, πολυεπίπεδη κοινή αρχιτεκτονική των παραπάνω προγραμμάτων που θα μπορούσε να μοιάζει κάπως έτσι:



εικόνα 24 : Ιεραρχική σχεδίαση, με διαφορετικό επίπεδο για κάθε κανόνα, τακτική, στόχο, πηγή: [5]

2.2.7 Υποκειμενικότητα κάθε Προγράμματος Go

Ένα πολύ καλό παράδειγμα της χρήσης, εφαρμογής και υποκειμενικής τροποποίησης των τριών παραπάνω δομικών μονάδων είναι ο τρόπος με τον οποίο το INDIGO [9] αποφασίζει για την επόμενη κίνηση όπως φαίνεται στην εικόνα.



εικόνα 25 : Στρατηγική Απόφασης Επόμενης Κίνησης του INDIGO, πηγή: [9]

Την θέση της EF έχει πάρει η μονάδα "Quick" Evaluation(η οποία χρησιμοποιώντας Mathematical Morphology χαρακτηρίζει κάθε ομάδα ως "Alive", "Dead" ή "Undetermined"), η μονάδα Move Generator παραμένει ίδια(και σε αντίθεση με άλλων παιχνιδιών, που στηρίζεται μόνο στους κανόνες στο Go τρέχει - λειτουργεί παράλληλα με το Quick Evaluation δίνοντας θέσεις που δεν υπακούν απλά στους κανόνες παιχνιδιού αλλά και σε αναγκαιότητα και τακτική), ενώ η Tree Search μονάδα έχει διαχωριστεί σε μικρότερες μονάδες και αποτελείται από τα:

- Global selective Quiescence Search
- "Life and Death" Group Search
- Pattern Matching
- String Search

Παρατηρούμε, λοιπόν, πως το INDIGO πραγματοποιεί τοπική Δενδρική Αναζήτηση ώστε να διαχωρίσει ποιες ομάδες μπορεί να σώσει και κάτω από ποιές προϋποθέσεις, αλλά και γενικές - συνολικές Δενδρικές Αναζητήσεις βάσει κάποιου στόχου ψάχνοντας πρότυπες κινήσεις(Patterns) για την επόμενη κίνηση, αντίπαλες ομάδες με αριθμό ελευθεριών ≤ 2, ώστε να επιτεθεί αλλά και μελλοντικές κινήσεις, που έχουν αρνητικό αντίκτυπο στην ροή του παιχνιδιού.

2.2.8 Συγκεντρωτική Ανάλυση Πρώτης Έκδοσης ενός Προγράμματος Go(INDIGO)

Το Πρόγραμμα INDIGO[2],[3],[8]

Το πρόγραμμα αυτό είναι από τα πρωτοπόρα στην προσπάθεια δημιουργίας προγραμμάτων, τα οποία όχι μόνο προσομοιώνουν ένα παίχτη Go αλλά προσπαθούν να του δώσουν και κάποιου είδους ανθρωπινή παικτική λογική για τη λήψη αποφάσεων σχετικά με την επόμενη κίνηση στο παιχνίδι. Έτσι, στα μέσα της δεκαετίας του 1990 ο Γάλλος Bruno Bouzy ξεκίνησε το INDIGO, το πρόγραμμα που ήταν εφαλτήριο για την σημαντική ερευνητική του συνεισφορά αλλά και την προσωπική καταξίωση στο χώρο του Computer Go.

Το INDIGO βασίζεται στην λογική ότι ένας άνθρωπος όταν παίζει Go εξαρτά τις αποφάσεις του από τις υπάρχοντες ομάδες, τις ελευθερίες αυτών, τις πιθανές νέες ομάδες που θα προκύψουν ή θα αποφευχθούν από την κίνηση του. Επίσης τα Eyes κάθε ομάδας παίζουν σημαντικό ρολό όπως και η κατάσταση Life or Death που την χαρακτηρίζει. Για το λόγο αυτό λοιπόν το INDIGO φροντίζει να εισάγει την χρήση μαθηματικών(Mathematical Morphology) στην ανάπτυξη του Computer Go. Από την στιγμή που το μέγιστο μέγεθος ενός πλαισίου παιχνιδιού του παιχνιδιού φτάνει το 19x19 ένα εμβαδό πολύ μικρότερο από εικόνες οι οποίες επεξεργάζονται από υπολογιστικά συστήματα πολλές από τις μεθόδους κατάλληλα τροποποιημένες για το Go φάνηκαν αρκετά χρήσιμες.

Με την χρήση τεχνικών και μεθόδων όπως Spatial Reasoning[2] και έπειτα από συγκεκριμένη αρχικοποίηση τιμών ακολουθουμένη από συγκεκριμένο αριθμό dilation και erosion

In this initial situation (let us suppose the line above is the edge of the board) :

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	128	0	0	128	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

After three dilations, the algorithm gives :

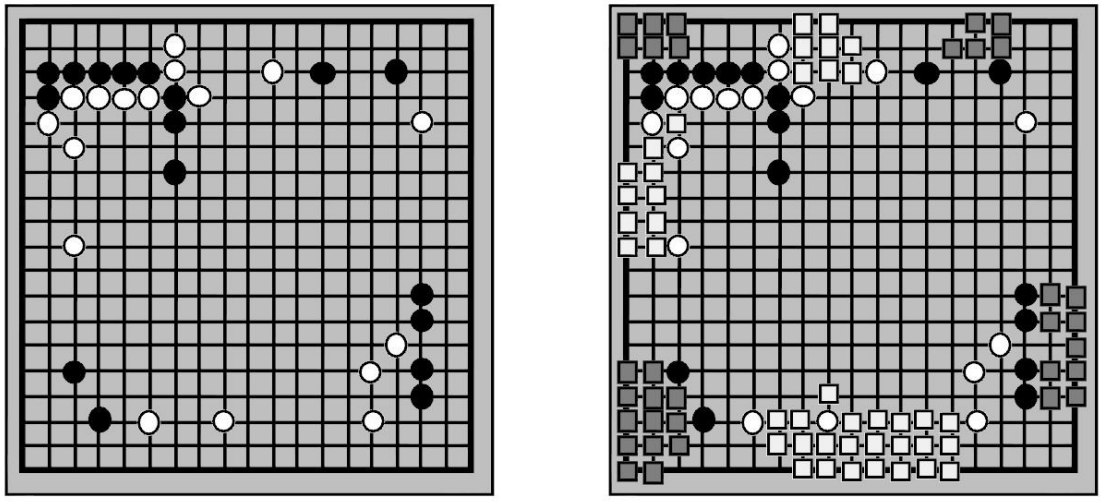
0	0	0	1	0	0	1	0	0	0
0	0	2	2	2	2	2	2	0	0
0	2	4	6	5	5	6	4	2	0
1	2	6	136	7	7	136	6	2	1
0	2	4	6	5	5	6	4	2	0
0	0	2	2	2	2	2	2	0	0

Then after seven contractions, it gives a small "territory" :

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	128	6	6	128	0	0	0
0	0	0	0	4	4	0	0	0	0
0	0	0	0	0	0	0	0	0	0

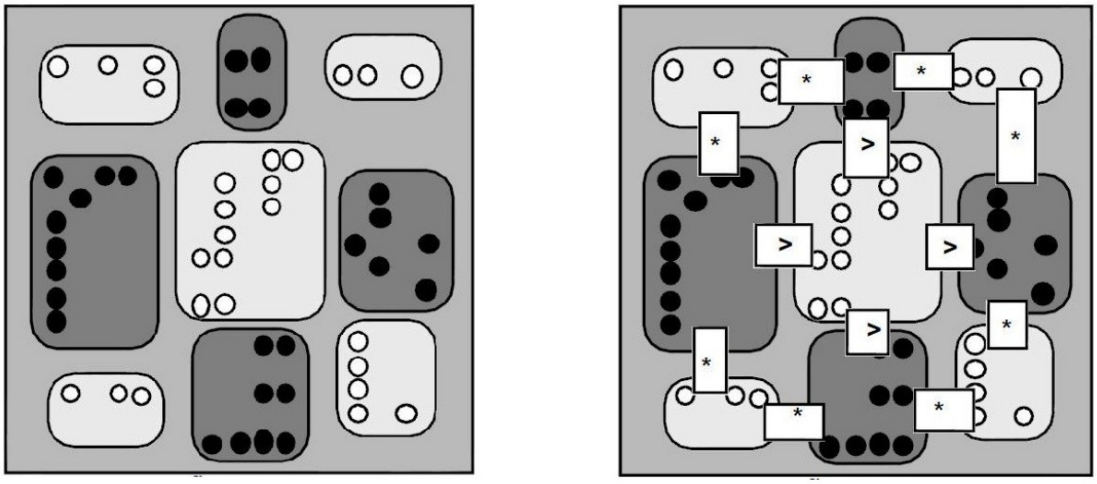
εικόνα 26 : Απλό παράδειγμα dilation & erosion, πηγή: [2]

να καθορίζει την επιρροή κάθε πέτρας ή ομάδας πετρών στο χώρο



εικόνα 27 : Περιοχές επιρροής πετρών στο πλαίσιο παιχνιδιού, πηγή: [2]

αλλά και την σχέση που έχουν οι ομάδες μεταξύ τους, αν είναι δηλαδή κάποιο λιγότερο ή περισσότερο συμπαγές από κάποιο άλλο ή αν είναι σχεδόν ίδιας "πυκνότητας".



εικόνα 28 : Σχέση περιοχών επιρροής πετρών στο πλαίσιο παιχνιδιού, πηγή: [3]

Σύμφωνα, λοιπόν, με την βαθμονόμηση κάθε διατομής του πλαισίου παιχνιδιού, η οποία αποτελεί και το επίπεδο επιρροής κάθε ομάδας - πέτρας στο χώρο το INDIGO καταλαβαίνει ποια ομάδα βρίσκεται κοντά σε αντίπαλου ή ιδίου χρώματος ομάδα και έτσι περιορίζει τους ελέγχους σε συγκεκριμένα σημεία του πλαισίου παιχνιδιού για την εύρεση της νέας κίνησης.

Η τεχνική αυτή είχε κάποια θετικά και κάποια αρνητικά στοιχειά. Λόγω της συνάρτησης επιρροής από την μια ο έλεγχος σπάει σε μικρότερα υπο-παιχνίδια και έτσι είναι πιο εύκολος και πιο γρήγορος, από την άλλη όμως το παιχνίδι χάνει το γενικό - συνολικό χαρακτήρα του και έτσι είναι αποδοτικό μόνο τοπικά στο πλαίσιο παιχνιδιού.

Γνωρίζοντας το αυτό ο Bouzy ισχυρίστηκε ότι το INDIGO δεν μπορεί να τεθεί εναντίον ανθρώπων - παιχτών και να αξιολογηθεί, διότι κάθε άνθρωπος έχει την δυνατότητα να ελέγχει και να αξιολογεί ταυτόχρονα την επιρροή κάθε στοιχείου στο πλαίσιο παιχνιδιού τοπικά και συνολικά. Όντως. όταν το INDIGO αντιμετώπισε άλλα προγράμματα Go είχε αξιόλογα αποτελέσματα.

2.2.9 碁(Go)

Όσο αφορά τους προγραμματιστές από τις χώρες καταγωγής του Go προσπάθησαν και ενδυνάμωσαν τα προγράμματα τους και με έναν επιπλέον τρόπο. Στηριζόμενοι στους κανόνες του παιχνιδιού αλλά και τον τρόπο ανάδειξης του νικητή σας τον παίχτη με τις περισσότερες πέτρες στο πλαίσιο παιχνιδιού, προσπάθησαν απλά να εξασφαλίσουν την δημιουργία όχι απλών ομάδων με δυνατή επιρροή στο χώρο αλλά "αμετακίνητων".

Έτσι προεβήκαν σε Life and Death ανάλυση των ομάδων. Δηλαδή, προσπάθησαν να προβλέψουν ποιά θα είναι η κατάληξη κάθε ομάδας πετρών αναλόγως με ποιος

παίζει που. Για να πετύχουν κάτι τέτοιο χρησιμοποίησαν αρχικά Δενδρικές Αναζητήσεις στην περιοχή κάθε ομάδας.

Σπάνια -ως συνήθως- κάτι λειτουργεί εξ αρχής ιδανικά με θετικά και μόνο αποτελέσματα. Έτσι λοιπόν τα προβλήματα που αντιμετώπισαν ήταν το αναγκαστικό μέγεθος των δένδρων αναζήτησης, η αναγκαστική επεξεργαστική ισχύ, που απαιτούσαν οι αναζητήσεις αυτές και συνεπώς ο συνολικός απαιτούμενος χρόνος που χρειαζόταν. Στραφήκαν, λοιπόν, σε διαφορετική λύση με την ιδία βάση και προφανώς το ίδιο αντίκτυπο. Πλέον χρησιμοποίησαν Static Analysis of Life and Death. Δανειζόμενοι μόνο το όνομα από την προγενέστερη τακτική οι K. Chen και Z. Chen προγραμματιστές των επιτυχημένων και βραβευμένων Go Intellect και HandTalk δημιούργησαν κάποιους γενικούς ευριστικούς κανόνες με σκοπό τον υπολογισμό της δυνατότητας κάθε ομάδας πετρών να δημιουργήσει Eyes ελέγχοντας τις άμεσα γειτονικές περιοχές. Συνεπώς, κατέληξαν με την δυνατότητα χαρακτηρισμού της "ζωτικής ικανότητας" κάθε ομάδας.

Προφανώς, όπως παραδεχτήκαν και οι ίδιοι οι προγραμματιστές, οι ευριστικοί κανόνες που δημιουργήθηκαν κάλυπταν ένα αρκετά μεγάλο ποσοστό περιπτώσεων και όχι όλες. Παρ' όλα αυτά ήταν ικανοί και αρκετοί ώστε ενσωματωμένοι στο υπόλοιπο πρόγραμμα να κάνουν τα Go Intellect και HandTalk αρκετές φορές πρωταθλητές σε πολλές Computer Go διοργανώσεις.

Όλα τα state of the art προγράμματα σήμερα χρησιμοποιούν όλες τις παραπάνω τεχνικές, με δικό του τρόπο και μέθοδο το καθένα, στηριζόμενα σε γρήγορες τακτικές αναζητήσεις(rapid tactical searches) και πιο χρονοβόρες γενικές αναζητήσεις για τις ομάδες πετρών(slower global group searches).

2.2.10 Μέθοδοι Monte Carlo

2.2.10.1 Monte Carlo(MC)

Με τον όρο Monte Carlo αναφερόμαστε σε μια ευρύτερη περιοχή αλγορίθμων, οι οποίοι επαναλαμβανόμενα εκτελούν πειράματα στατιστικής δειγματοληψίας με σκοπό να παρέχουν μια προσεγγιστική λύση σε ένα μεγάλο εύρος προβλημάτων. Αρχικά, χρησιμοποιήθηκαν για επίλυση προβλημάτων στην επιστήμη της φυσικής, όμως μετέπειτα εισήρθαν στις οικονομικές επιστήμες αλλά και στον τομέα της Τεχνητής Νοημοσύνης με σημαντική συνεισφορά.

Χρησιμοποιώντας μαθηματικά μπορούμε να μοντελοποιήσουμε την βασική ιδέα ως εξής: Θεωρούμε μια τυχαία μεταβλητή X με συνάρτηση πυκνότητας πιθανότητας $f_X(x)$, η οποία είναι θετική για ένα σετ τιμών της X και μια συνάρτηση g της τυχαίας

μεταβλητής. Θεωρώντας ότι η τ.μ. X είναι συνεχής λαμβάνουμε n τυχαία δείγματα της X, $(x^{(1)},\ldots,x^{(n)})$ και υπολογιζουμε την μεση τιμη της $g(x)$.

$$g_n(x) = \frac{1}{n}\sum_{i=1}^{n} g(x^{(i)})$$

Σε συνδυαστικά παιχνίδια όπως το Go κατά την διάρκεια των οποίων ο κάθε παίχτης ακολουθεί την δική του στρατηγική, το τελικό αποτέλεσμα ποικίλει και εμπεριέχει κατά κάποιο βαθμό αβεβαιότητα. Το κύριο χαρακτηριστικό και πλεονέκτημα των προσομοιώσεων Monte Carlo είναι ότι μοντελοποιούν μεγάλο αριθμό τυχαίων παιχνιδιών μέχρι το τέλος, δίνοντας την δυνατότητα υπολογισμού ποσό "πιθανό" είναι να έρθει κάποιο αποτέλεσμα.

Παραδείγματος χάρη εάν ένας παίχτης θέλει να παίξει μια κίνηση τότε μπορεί να πραγματοποιήσει αρκετά παιχνίδια μέχρι τέλους με αρχική κίνηση την συγκεκριμένη και τυχαία συνέχεια ώστε να υπολογίσει πόσο πιθανό είναι η κίνηση του να τον οδηγήσει σε νίκη, ήττα η ισοπαλία. Έτσι λοιπόν η παραπάνω εξίσωση $g_n(x)$ τροποποιημένη ανάλογα για ένα συνδυαστικό παιχνίδι σύμφωνα με τους Gelly και Silver[28] θα είναι:

$$Q(s,a) = \frac{1}{N(s,a)}\sum_{j=1}^{N(s)} II_j(s,a) z_j$$

όπου, $N(s,a)$ είναι ο αριθμός των φορών όπου η κίνηση a πραγματοποιήθηκε από την κατάσταση s, ο συντελεστής $N(s)$ δείχνει τον συνολικό αριθμό των τυχαίων παιχνιδιών που πραγματοποιήθηκαν, $II_i(s,a)$παίρνει τιμή 1 ή 0 εάν η κίνηση a πραγματοποιήθηκε από την κατάσταση s στην j - ιωστή επανάληψη ή όχι αντίστοιχα και τέλος z_j είναι το αποτέλεσμα της j - ιωστής επανάληψης που ξεκίνησε από την κατάσταση s.

2.2.10.2 Simulated Annealing

Το πρώτο πρόγραμμα, που χρησιμοποίησε αυτή την τεχνική είναι το Gobble[1] του Bernd Brügmann την δεκαετία του 1980. Ο αλγόριθμος που χρησιμοποιούσε το παιχνίδι ήταν ο εξής: Αρχικά, έπαιζε έναν εύλογο αριθμό παιχνιδιών μέχρι το τέλος εντελώς τυχαία. Ανάλογα με το αποτέλεσμα του κάθε παιχνιδιού ανανέωνε την μέση τιμή κάθε θέσης δικού του χρώματος. Έπειτα διαμόρφωνε μια λίστα με την οποία θα έπαιζε τις κινήσεις του με πρώτη αυτή με την καλύτερη βαθμολογία και ούτω καθ εξής. Σε περίπτωση που δεν ήταν δυνατό να παίξει την κίνηση i μετέβαινε στην επόμενη διαθέσιμη. Μετά από κάποιες αρχικές κινήσεις και των δύο παικτών αυτό που εφάρμοζε ήταν ότι παίζοντας ο αντίπαλος την κίνηση του επαναξιολογούσε τις κινήσεις του με έναν αριθμό τυχαίων παιχνιδιών -προφανώς μικρότερο από τον

αρχικό- προσδίδοντας ένα επιπλέον χαρακτηριστικό το οποίο είναι η πιθανότητα να μετατοπιστούν σε γειτονικές θέσεις ανάλογα με την διαφορά της ανανεωμένης με την προηγούμενη μέση τιμή διαιρεμένη με μια παράμετρο που έχει τον ρόλο της θερμοκρασίας στον κλασσικό αλγόριθμο της μεθόδου simulated annealing. Η αρχική εξίσωση της πιθανότητας μιας θέσης από την λίστα να μετατοπιστεί κατά n θεσης είναι η παρακάτω:

$$p(n) = (p_{swap})^n = \exp(-n/T)$$

Στόχος είναι να μειωθεί γραμμικά με το πέρας του παιχνιδιού η τιμή p_{swap} ώστε να γίνει 0 στους τελευταίους γύρους υποδεικνύοντας το κοντινότερο τοπικό μέγιστο. Το Gobble αντιμετώπισε το Many Faces of Go πρωταθλητή του North American Computer Go Champion και τα αποτελέσματα ήταν θετικά αναλογικά με την αξία και ικανότητα του αντίπαλου προγράμματος του David Fotland.

2.2.10.3 Μέθοδος Bandit - Based

Συνήθως, στις περισσότερες προσομοιώσεις η επιλογή τυχαίων κινήσεων ακολουθεί κανονική κατανομή. Όμως ο Coulom[17] απέδειξε ότι με την χρήση κάποιων ευριστικών κανόνων και μη ακλουθώντας κανονική κατανομή για την επιλογή των τυχαίων θέσεων αλλά αυτή που καθορίζεται από τους κανόνες προκύπτουν καλύτερα αποτελέσματα με την εφαρμογή τεχνικών Monte Carlo σε Δενδρικές Αναζητήσεις. Δεν υπάρχει όμως αλγόριθμος ο οποίος να μην υστερεί ή να έχει μειονεκτήματα.

Το θέμα που προκύπτει λοιπόν είναι το γνωστό Exploration vs. Exploitation, μιας και η κατανομή του αποτελέσματος κάθε κίνησης είναι άγνωστη σε κάθε παιχνίδι. Δηλαδή το Trade - Off αν θα προσομοιώσουμε περισσότερες κινήσεις σαν αρχικές ή σε μεγαλύτερο βάθος κάποιες υποσχόμενες με θετικά αποτελέσματα. Για να κατανοήσουμε τις προτάσεις κάποιον προγραμματιστών στο συγκεκριμένο θέμα καλό είναι να αναφερθούμε στις Bandit - Based Methods. Το καλύτερο πρόβλημα - παράδειγμα που επιλύουν μέθοδοι Bandit - Based είναι αυτό της επιλογής του επόμενου βραχίονα, από τους απείρους διαθέσιμους, ενός κουλοχέρη ώστε να μεγιστοποιήσουμε το κέρδος μας.

Μοντελοποιώντας μαθηματικά το πρόβλημα, έστω ότι ο παίχτης επιλέγει ένα βραχίονα, του οποίου το αποτέλεσμα είναι ένα δείγμα από την συγκεκριμένη κατανομή πιθανότητας που ανήκει σε αυτό το βραχίονα. Συνεπώς το συνολικό κέρδος που επιθυμεί να μεγιστοποιήσει ο παίχτης μπορεί να περιγράφει έτσι:

$$R_t = r_{t+1} + r_{t+2} + r_{t+3} + \cdots = \sum_{k=1}^{\infty} r_{t+k}$$

Συγκεκριμενοποιώντας το πρόβλημα, έστω ότι έχουμε ένα καθορισμένο αριθμό βραχιόνων N το αποτέλεσμα μ_i που θα λάβουμε αν τραβήξουμε τον βραχίονα i είναι ένα δείγμα μιας κατανομής P_i. Τα αποτελέσματα, που προκύπτουν από το τράβηγμα του i βραχίονα, τα οποία βελτιώνουν το συνολικό κέρδος είναι ανεξάρτητα, ομοίως κατανεμημένα. Ο παράγοντας R_N που φανερώνει την μέτρηση της επιτυχίας των Bandit - Based μεθόδων είναι ο Regret, ο οποίος είναι αντιστρόφως ανάλογος του κέρδους στο παράδειγμα με τον κουλοχέρη, διότι όσο μικρότερος είναι τόσο καλύτερα. Ο παράγοντας Regret ορίζεται σαν την αναμενόμενη διάφορα μεταξύ του καλύτερου δυνατού αποτελέσματος και του αθροίσματος των αποτελεσμάτων έπειτα από n τραβήγματα.

$$R_N = \sum_{i=1}^{M} (\mu^* - \mu_i) E[T_i(n)] = \mu * n - \sum_{i=1}^{M} \mu_i E[T_i(n)]$$

όπου, $T_i(n)$ είναι οι φορές όπου επιλέχθηκε ο βραχίονας i στα n τραβήγματα που προσομοιώθηκαν, $E[T_i(n)]$ οι φορές να επιλέγει ο βραχίονας i, $\mu^* = \mu * n = \max_{1 \le i \le M} \mu_i$ το βέλτιστο δυνατό αποτέλεσμα, μ_i το αναμενόμενο αποτέλεσμα του βραχίονα i.Σκοπός, λοιπόν είναι να ελαχιστοποιήσουμε την τιμή Regret, ή ακόμη πιο ρεαλιστικά να την σταθεροποιήσουμε σε χαμηλές τιμές. Αυτό προσφέρει ο αλγόριθμος UCB[18].

2.2.10.4 Αλγόριθμος UCB

Έχει αποδεδειχθεί ότι κανένας αλγόριθμος δεν μπορεί να εγγυηθεί ότι η τιμή Regret θα αυξάνεται με ρυθμό μικρότερο του $O(\ln n)$, συνεπώς θα είναι ευτύχημα αν καταφέρουμε να κρατήσουμε τον ρυθμό αύξησης σταθερό κοντά στον $O(\ln n)$. Για αυτό σύμφωνα με τον αλγόριθμο μειώνεται ο αριθμός των τυχαίων παιχνιδιών για υπο-βέλτιστες(Sub - Optimal) κινήσεις και ώστε να ελεγχτούν σε μεγαλύτερο βάθος κινήσεις με καλύτερο αποτέλεσμα. Το μαθηματικό μοντέλο του αλγορίθμου είναι το παρακάτω:

$$UCB(j) = \bar{X}_j + \sqrt{\frac{2 * \ln n}{n_j}}$$

όπου, $\bar{X}_j$ το μέσο σκορ της κίνησης j, n είναι ο αριθμός των παιχνιδιών που έχουν προσομοιωθεί και n_j τον αριθμό των φορών που παίχτηκε η κίνηση j.

Ο ψεδοκώδικας για του αλγορίθμου είναι ο εξής:

Policy: UCB1
Initialization: Play each machine
Loop:

- Play machine j that maximises: $\bar{X}_j + C * \sqrt{\frac{2*\ln n}{n_j}}$
- Get reward r_j
- $t = t + 1 \ldots$

, where $j \in 1..,N$ number of machines

Factor $\bar{X}_j$ urges the exploitation of actions that yield higher rewards, while factor $\sqrt{\frac{2*\ln n}{n_j}}$ ensures that less visited game states will not be completely neglected.

εικόνα 29 : Αλγόριθμος UCB

Η τροποποίηση του αλγορίθμου UCB ώστε να χρησιμοποιηθεί σε Δενδρικές Αναζητήσεις ονομάζεται UCT και έγινε το 2006 από τους Kocsis & Szepesvári. Πριν φτάσουμε όμως εκεί ας αναφερθούμε στην πολύ σημαντική μέθοδο Monte Carlo Tree Search(M.S.T.C.), την οποία επινόησε και πρωτο - εφάρμοσε ο Coulom [18]την ίδια χρονιά (2006).

2.2.10.5 Η Άποψη του Coulom

Η τεχνική Progressive Pruning αλλά και τα θετικά αποτελέσματα που είχε οδήγησαν στη περαιτέρω ανάπτυξη της [15], [18]. Ο Coulom συνδύασε την τεχνική MC και TS με ένα μοναδικό τρόπο. Η ρίζα κάθε δένδρου είναι η αρχική κατάσταση πριν ξεκινήσει η προσομοίωση του τυχαίου παιχνιδιού. Το δένδρο χτίζεται όσο εξελίσσεται η προσομοίωση με κάθε επόμενη κίνηση να αποτελεί κόμβο παιδί της προηγούμενης. Προφανώς, δεν αποθηκεύονται όλες οι κινήσεις μέχρι το τέλος του παιχνιδιού, διότι το δέντρο θα έφτανε σε πολύ μεγάλο βάθος συγκεντρώνοντας και άχρηστη πληροφορία, αλλά διότι όσο μεγαλώνει το βάθος τόσο λιγότερο επιρροή ασκούν οι βαθύτεροι κόμβοι. Έτσι, λοιπόν, κάθε φορά που ένας κόμβος του δένδρου προσπελαύνεται για δεύτερη φορά, τότε ένας νέος κόμβος - παιδί δημιουργείται με ρίζα το κόμβο αυτό. Σε κάθε κόμβο του δένδρου αποθηκεύονται οι εξής πληροφορίες: ο **αριθμός τον παιχνιδιών**, τα οποία διέσχισαν αυτό τον κόμβο αλλά και η **μέση τιμή του σκορ** των παιχνιδιών που διέσχισαν τον κόμβο αυτό όπως και το **τετράγωνο της μέσης τιμής**. Αυτό, γιατί η αξιολόγηση και το επικείμενο κλάδεμα κάποιων θέσεων στηρίζεται στο Κεντρικό Οριακό Θεώρημα. Σύμφωνα με το οποίο, εφόσον (θεωρώντας το σκορ μια τυχαία μεταβλητή μέσης τιμής μ και διακύμανση σ^2) η μέση τιμή του σκορ N ανεξάρτητων τυχαίων παιχνιδιών τείνει να ακολουθεί κανονική κατανομή με μέση τιμή μ και διακύμανση σ^2/N μας δίνεται η δυνατότητα

να ψαλιδίσουμε κινήσεις των οποίων η πιθανότητα του μέσου σκορ ξεπερνά αυτή που ορίζεται από την κατανομή του Κεντρικού Οριακού Θεωρήματος. Το πρόγραμμα του Coulom Crazy Stone που υιοθέτησε αυτή την τακτική κατάφερε να κερδίσει έναν διεθνή διαγωνισμό πριν από αυτό όμως είχε κάποια αποτελέσματα άξια σχολιασμού απέναντι στο GNUGo και το INDIGO όπως φαίνεται στον πίνακα.

Player	Opponent	Winning Rate	Komi
CrazyStone (5 min / game)	Indigo 2005 (8 min / game)	61% (±4.9)	6.5
Indigo 2005 (8 min / game)	GNUGo 3.6 (level 10)	28% (±4.4)	6.5
CrazyStone (4 min / game)	GNUGo 3.6 (level 10)	25% (±4.3)	7.5
CrazyStone (8 min / game)	GNUGo 3.6 (level 10)	32% (±4.7)	7.5
CrazyStone (16 min / game)	GNUGo 3.6 (level 10)	36% (±4.8)	7.5

εικόνα 30 : Πίνακας αποτελεσμάτων CrazyStone vs. INDIGO & CrazyStone vs. GNUGo

Φαίνεται πως υπερτερεί του INDIGO και από τα μεταξύ τους αποτελέσματα αλλά και από την καλύτερη αποδοτικότητα του Crazy Stone απέναντι στο πανίσχυρο GNUGo, το οποίο υπερισχύει όσο και αν αυξηθούν οι προσομοιώσεις MC(και συνεπώς ο απαιτούμενος χρόνος ολοκλήρωσης κάθε παιχνιδιού) από το Crazy Stone.

2.2.10.6 Monte Carlo Tree Search (M.S.T.C.)

Προφανώς και αυτή η μέθοδος όπως όλες έχει πλεονεκτήματα αλλά και μειονεκτήματα.

Πλεονεκτήματα:

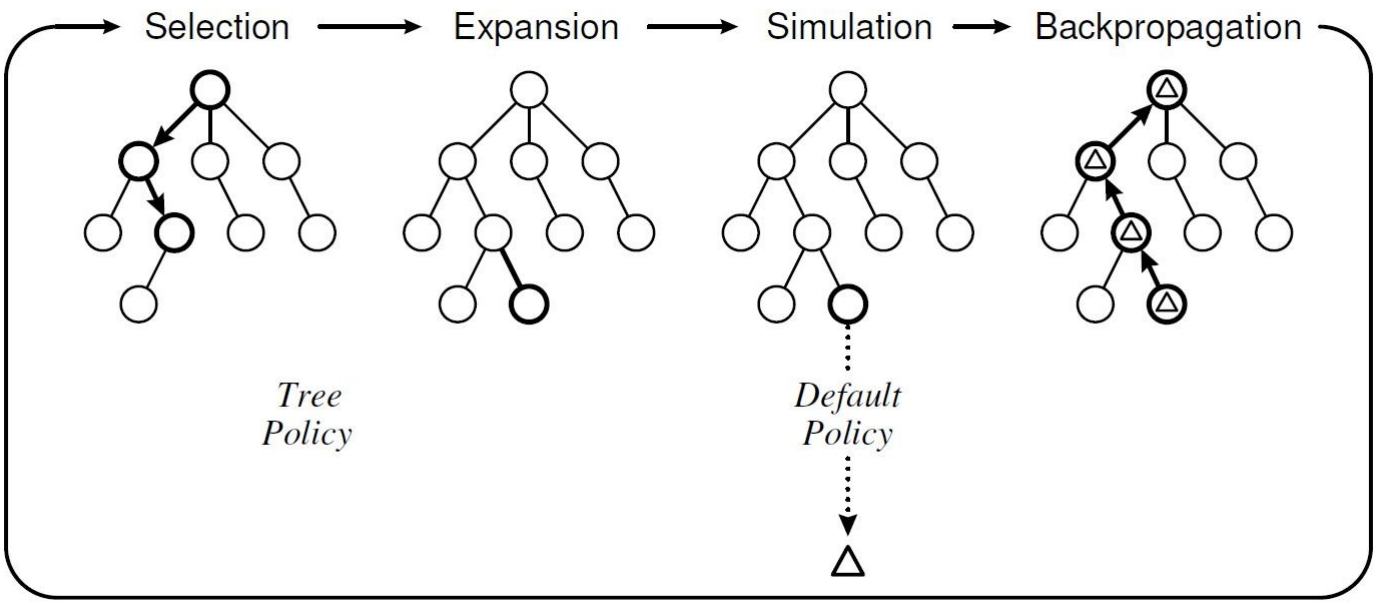
- Πολύ σημαντικό το γεγονός ότι λειτουργεί με θετικά αποτελέσματα ακόμη και χωρίς την ύπαρξη ευριστικών κανόνων ή τακτικής σχετικά με το παιχνίδι.
- Η ασυμμετρία που εμφανίζει το δένδρο που δημιουργείται κλίνοντας προς τις κινήσεις με μεγαλύτερη πιθανότητα θετικού αντίκτυπου, διευκολύνει στην καλύτερη κατανόηση του παιχνιδιού.
- Το γεγονός ότι ανά πάσα στιγμή σταματήσει η προσομοίωση λόγω περιορισμού χρόνου ή αριθμού επαναλήψεων θα μας επιστραφεί η καλύτερη τιμή μέχρι εκείνη την στιγμή, αφού σε κάθε προσομοίωση όλοι οι κόμβοι που συμμετέχουν ανανεώνονται άμεσα.

Μειονεκτήματα:

- Η απλούστερη έκφραση του αλγορίθμου οδηγεί κάποιες φορές σε λανθασμένη εύρεση κίνησης, διότι σταματά την αναζήτηση αρκετά νωρίς περιορίζοντας το exploration αγνοώντας κινήσεις που θα έδιναν σίγουρα θετικό αποτέλεσμα.
- Η απαίτηση προσομοίωσης αρκετά μεγάλου αριθμού παιχνιδιών, ώστε να καταλήξει σε μια καλή κίνηση αλλά και η δενδρική αναζήτηση που ακολουθεί καταναλώνουν αρκετό χρόνο με αποτέλεσμα να μην θεωρείται και από τις ταχύτερες προσεγγίσεις.

Ο αλγόριθμος M.S.T.C. αποτελείται από 4 κυρία βήματα:

- **Selection:** Επιλογή κόμβου, ο οποίος δεν έχει προσομοιωθεί - εξαπλωθεί. Μια αρκετά σημαντική διαδικασία, διότι η τακτική που θα χρησιμοποιηθεί για την επιλογή του επόμενου κόμβου είναι εκείνη που καθορίζει την συμπεριφορά του αλγορίθμου στο trade - off exploration vs. exploitation. Άπληστες(greedy) τακτικές επιλογής του κόμβου με το καλύτερο σκορ δεν έχουν τα καλύτερα αποτελέσματα, επειδή το υψηλό σκορ ενός κόμβου δεν συνεπάγεται και αρκετή εξερεύνηση του ώστε να θεωρείται και αρκετά αξιόπιστη επιλογή. Το πρόβλημα αυτό μοιάζει αρκετά με τα multi - armed based προβλήματα, που αναφερθήκαμε παραπάνω. Δυστυχώς, όμως αρκετές τέτοιες επιλογές πρέπει να γίνουν και μάλιστα στην αρχή του δένδρου μέχρι και βάθος ίσο με 3 περίπου, για κινήσεις που σε παιχνίδια σαν το Go παρόλο που είναι αρχικές μπορεί να αποδειχθούν αρκετά κρίσιμες. Υπάρχουν βέβαια και κάποιες τακτικές που απαιτούν αρκετό χρόνο εκτέλεσης σύμφωνα με τις οποίες στην αρχή του παιχνιδιού κάθε κόμβος πρέπει να έχει προσπελαστεί τουλάχιστον μια φορά πριν αρχίσει η επέκταση του δένδρου.
- **Expansion:** Επέκταση του δένδρου με τοποθέτηση παιδιού στον παραπάνω κόμβο. Και σε αυτό το κομμάτι, υπάρχουν τεχνικές που διαφέρουν από την απλή τοποθέτηση ενός κόμβου παιδιού μόνο στο γεγονός ότι ανάλογα το σημείο του παιχνιδιού μπορεί να τοποθετηθούν περισσότεροι κόμβοι επεκτείνοντας πιο γρήγορα αλλά όχι πιο σωστά απαραίτητα το δένδρο.
- **Simulation:** Προσομοίωση τυχαίου παιχνιδιού. Κλασσική διαδικασία κατά την οποία το παιχνίδι εικονικά φτάνει μέχρι το τέλος, ώστε να αξιολογήσουμε την κίνηση με τη οποία ξεκινά η προσομοίωση.
- **Backpropagation:** Ανανέωση των τιμών των κόμβων ξεκινώντας από τον νεότερο προς τον αρχαιότερο. Διαδικασία, στη οποία ανανεώνονται οι οποιεσδήποτε τιμές αποθηκεύει κάθε κόμβος σύμφωνα με τον αλγόριθμο που ακολουθείται.



εικόνα 31 : Τα τέσσερα κύρια βήματα του M.S.T.C., πηγή: [21]

Ο ψεδοκώδικας του αλγορίθμου είναι ο εξής:

Algorithm 2 Monte Carlo Tree Search pseudo-code

Input: *root_node*

Output: *optimal_move*

while (*WithinComputationalBudget*) **do**
  current_node ← *root_node*
  % Selection Phase
  while (*current_node* ∈ *SearchTree*) **do**
    last_node ← *current_node*
    current_node ← *Select*(*current_node*)
  % Expand Phase
  last_node ← *Expand*(*last_node*)
  % Simulation Phase
  result ← *Simulation*(*last_node*)
  % Backpropagation Phase
  while (*current_node* ∈ *SearchTree*) **do**
    current_node.Backpropagate(*reward*)
    current_node.visit_number ← *current_node.visit_number* + 1
    current_node ← *current_node.parent*
optimal_move = $argmax_{N \in N_{c(root\_node}}(N.best\_move)$
return *optimal_move*

εικόνα 32 : Αλγόριθμος M.S.T.C.

Τέλος, σύμφωνα με τον Chaslot [27] μπορεί να υπάρξει και κάποιου είδους τακτικής στην επιλογή της τελικής κίνησης.

- **Max Child:** Επιλογή του παιδιού με την υψηλότερη τιμή.
- **Robust Child:** Επιλογή του παιδιού που είχε την μεγαλύτερη επισκεψιμότητα.
- **Robust - max Child:** Επιλογή του παιδιού που έχει και τις περισσότερες επισκέψεις αλλά και το μεγαλύτερο σκορ. Εάν δεν υπάρχει αυτή η κίνηση, η τακτική επιβάλλει την συνέχιση των simulations ώστε να προκύψει.
- **Secure Child:** Επιλογή της κίνησης η οποία μεγιστοποιεί το lower confidence bound.

Αξίζει να σημειωθεί, ότι έπειτα από πειραματικές διαδικασίες η τακτική **Max Child** τείνει να είναι η πιο αδύναμη.

2.2.10.7 Αξιολόγηση και Εξέλιξη Τεχνικών Monte Carlo

Όσο προχωρούσαν οι έρευνες με σκοπό την εξέλιξη και καθιέρωση τεχνικών Monte Carlo στο Go έχοντας εξαιρετικά αποτελέσματα τόσο περισσότεροι προγραμματιστές αναδιαμόρφωναν τους κώδικες τους δίνοντας την δυνατότητα στα προγράμματα τους να εκτελούν ένα σεβαστό αριθμό τυχαίων παιχνιδιών ώστε να αποφασίζουν καλύτερα. Η βασική ιδέα ήταν κοινή[14], όμως όπως και η EF κάθε προγράμματος έτσι και οι εξελίξεις έφεραν την προσωπική αντίληψη του δημιουργού.

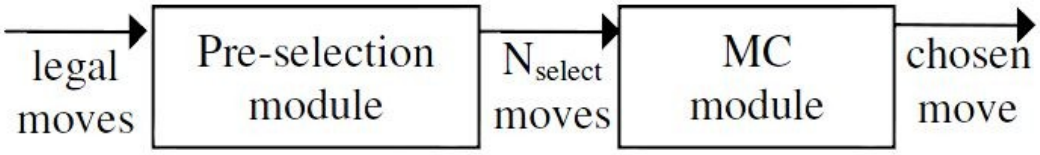
Περίπου από το 2004 και μετά όλα τα γνωστικά(knowledge - based) προγράμματα χρησιμοποιούσαν και τεχνικές MC. Έτσι λοιπόν στράφηκαν όλοι να βελτιστοποιήσουν και να εκμεταλλευτούν όσο το δυνατόν περισσότερο αυτές τις τεχνικές, οι οποίες σε πολύ σύντομο χρόνο εκτέλεσης και με εξίσου μικρή πολυπλοκότητα εμφάνιζαν χρησιμότατα αποτελέσματα. Ο κύριος άξονας αναζήτησης όλων ήταν πως θα τροφοδοτούν τις μονάδες MC με τις λιγότερες αρκετές κινήσεις ώστε να εκμεταλλευτούν το γεγονός πως όσο περισσότερα τυχαία παιχνίδια ξεκινούν με μια συγκεκριμένη θέση τόσο περισσότερη και πιο αξιόπιστη πληροφορία συγκεντρώνεται για την θέση αυτή.

Για το λόγο αυτό ο Bouzy αφού εισήγαγε στο INDIGO τεχνικές MC και τις ενίσχυσε με χωρικούς(territory) και ιστορικούς(history) ευριστικούς κανόνες. Όσο αφορά τους χωρικούς ευριστικούς κανόνες ισχυρίστηκε ότι για κάθε μια από τις διαθέσιμες θέσεις που καταλαμβάνονται από πέτρες κατά την εκτέλεση τυχαίων παιχνιδιών μπορούμε να κρατάμε την μέση τιμή του χρώματος με το οποίο κατέληξαν στο τέλος της προσομοίωσης $+1$ για μαύρο, -1 για άσπρο και 0 για κενό παραδείγματος χάρη. Μεταφέροντας το αποτέλεσμα x από το εύρος τιμών $[-1,+1]$ στο $[0,1]$ και χρησιμοποιώντας την συνάρτηση $F(x) = exp(-kx^2)$, όπου k μια θετική σταθερά προκύπτει μια τιμή αναγκαιότητας την οποία αν μετατρέψουμε σε πιθανότητα μας δείχνει ποιες θέσεις έχουν μικρή πιθανότητα να καλυφτούν από

χρώμα μας. Αυτόν τον κανόνα μπορούμε να τον χρησιμοποιήσουμε και πριν αλλά και κατά την διάρκεια της προσομοίωσης παιχνιδιών.

Χρησιμοποιώντας τον πριν έχουμε την δυνατότητα να τροφοδοτήσουμε την μονάδα MC με θέσεις μικρής πιθανότητας να ελέγξουμε, οπότε να στραφούμε σε μια προσπάθεια να το πετύχουμε. Ενώ χρησιμοποιώντας τον κατά την διάρκεια μετατρέπουμε το παιχνίδι μας σε πιο ορθολογικό και λιγότερο τυχαίο γιατί θα αποφεύγουμε να τοποθετήσουμε πέτρες του χρώματος μας σε θέσεις που έχουμε μικρή δυνατότητα να καλύψουμε. Για κάθε μια από τις παραπάνω περιπτώσεις χρησιμοποιείται προφανώς διαφορετικός συντελεστής k. Όπως και οι πρώτοι ευριστικοί κανόνες έτσι και οι δεύτεροι έχουν πάρει το όνομα τους από τον τρόπο λειτουργίας τους. Σύμφωνα με τους ιστορικούς ευριστικούς κανόνες η κίνηση ή οι κινήσεις που προσομοιώθηκαν σαν αρχικές με τεχνικές MC και δεν επιλέχτηκαν αλλά είχαν το χαμηλότερο ή αρνητικό μέσο όρο κόβονται από την επομένη προεπιλογή όπου είναι μεγάλη η πιθανότητα να ξαναβρεθούν.

Ο τρόπος με τον οποίο γίνεται αυτό είναι με το να μειώνεται η πιθανότητα επιλογής της συγκεκριμένης κίνησης διαιρούμενη από έναν αριθμό, ο οποίος όσο προχωρά το παιχνίδι μικραίνει ώστε να μην επηρεάζει τόσο πολύ προς το τέλος όπου μειώνεται οι κινήσεις και είναι εύκολο να ελεγχτούν όλες ή σύμφωνα με τον κανόνα να μην ελεγχτεί καμία. Οπότε η νέα αρχιτεκτονική είχε σαν σκοπό να αντλεί πληροφορίες από τις προσομοιώσεις MC και να τις επιστρέφει σε μια μονάδα, την οποία θα τις χρησιμοποιούσε ώστε να αξιολογήσει τις εναπομείναντες διαθέσιμες κινήσεις και να τροφοδοτήσει την μονάδα τυχαίων παιχνιδιών με τις νέες αρχικές θέσεις προς προσομοίωση.



εικόνα 33 : Σχεδίαση μονάδας MC με μονάδα προ - επιλογής κινήσεων, πηγή: [16]

Πραγματοποιώντας πειράματα - αναμετρήσεις αρχικά μεταξύ συστημάτων με τους κανόνες και χωρίς διαπίστωσε ότι οι κανόνες που αφορούσαν το ιστορικό παρουσίασαν βελτίωση +10 πόντων μόνο, ενώ οι χωρικοί και μάλιστα το πρόγραμμα που αναλάμβανε να εκτελέσει έναν μικρό αριθμό τυχαίων παιχνιδιών έξω από την μονάδα MC ώστε να τροφοδοτήσει εκείνο με θέσεις που δύσκολα θα καλύπτονταν από δικό μας χρώμα πέτρες παρουσίασε +40 πόντους βελτίωση. Έτσι όταν το πρόγραμμα με τους "External Territory Heuristics"(ETH) τέθηκε ενάντια στον πρόγονό του που δεν είχε αυτούς χωρίς τους κανόνες αυτούς αλλά και το GNUGo παρατηρηθήκαν θετικά αξιοσημείωτα αποτελέσματα για όλα τα μεγέθη των πλαισίων παιχνιδιών.

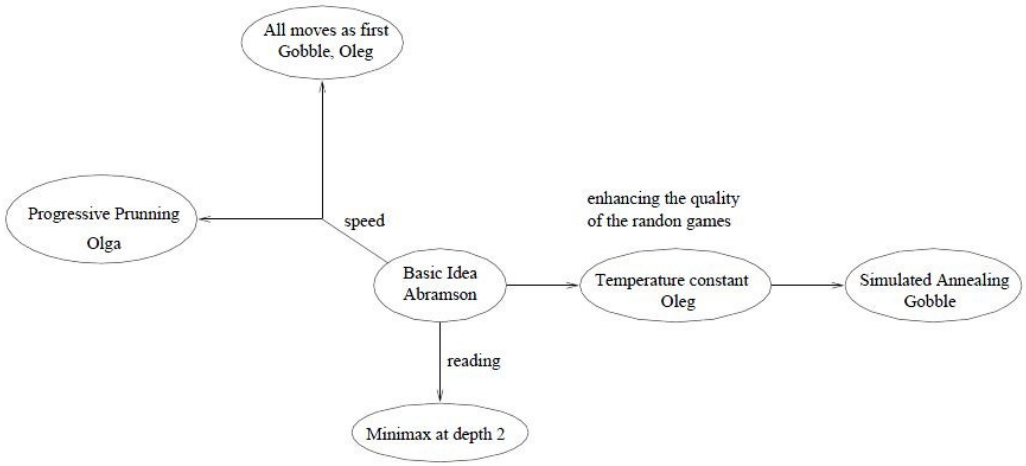
Board Size	Self - play	GNUGo
9x9	+1	-1
13x13	+15	+3
19x19	+43	+10

εικόνα 34 : Πίνακας αποτελεσμάτων INDIGO with ETH vs. INDIGO & INDIGO with ETH vs. GNUGo

Βλέπουμε την μεγάλη διαφορά δυναμικότητας απέναντι στο πρόγραμμα πρόγονο αλλά και την ελάχιστα μικρή επικράτηση του δυνατού GNUGo στο κλασσικό πλαίσιο παιχνιδιού, ένα πολλά υποσχόμενο αποτέλεσμα.

2.2.10.8 Η Άποψη των Bouzy & Helmstetter

Οι Bouzy και Helmstetter[10] αφιέρωσαν αρκετό χρόνο για να δημιουργήσουν δύο προγράμματα, ώστε να συγκρίνουν, να αξιολογήσουν και να βελτιώσουν τις υπάρχουσες τεχνικές Monte Carlo.



εικόνα 35 : Βελτιώσεις των Bouzy & Helmstetter στον άξονα της ταχύτητας, πηγή: [10]

Ο πρώτος υλοποίησε το OLGA που χρησιμοποιεί τις δομές δεδομένων του INDIGO και ο δεύτερος το OLEG έχοντας δανειστεί τις δομές του GNUGo. Η υλοποίηση αφορά 9x9 πλαίσιο παιχνιδιού με το OLGA να παίζει 7000 παιχνίδια / sec και το OLEG 10000 παιχνίδια / sec σε έναν H/Y 2 GHz. Έχοντας κύριο εμπνευστή τον Bernd Brügmann και το Gobble ξεκίνησαν τις αναβαθμίσεις και τα πειράματα.

Αρχικά, χρησιμοποιήθηκε η τεχνική Progressive Pruning σύμφωνα με την οποία κάθε κίνηση έχει μια μέση τιμή m και μια και μια τυπική απόκλιση σ ένα κατώτατο αναμενόμενο σκορ - βαθμολογία m_l και ένα ανώτατο m_r. Για κάθε κίνηση θεωρούμε : $m_l = m - \sigma r_d$ και $m_r = m + \sigma r_d$, όπου r_d είναι ένας σταθερός παράγοντας,

που προέκυψε πειραματικά. Μια κίνηση M_1είναι στατιστικά κατώτερη από μια κίνηση M_2 εάν $M_1 * mr_r < M_2 * m_l$. Εάν και για τις δύο κινήσεις ισχύει: $M_1 * \sigma < \sigma_e$ και $M_2 * \sigma < \sigma_e$, όπου σ_e είναι η πειραματική τιμή της τυπικής απόκλισης της ισότητας, τότε χαρακτηρίζονται στατιστικά ίσες.

Στην τεχνική αυτή κάθε αρχική κίνηση προσομοιώνεται $N_m = 100$ φορές και αν είναι στατιστικά κατώτερη από κάποια άλλη τότε δεν λαμβάνεται υπ' όψιν σαν μελλοντική κίνηση. Αυτό μπορεί να συμβαίνει μέχρι να μείνουν οι καλύτερες κινήσεις ή καλύτερη κίνηση ή όταν φτάσουμε ένα όριο(threshold) επαναλήψεων που έχουμε ορίσει. Το μέγιστο όριο ορίζεται σαν τον αριθμό των έγκυρων διαθέσιμων κινήσεων στο πλαίσιο παιχνιδιού πολλαπλασιασμένος με το 10.000.

Ο παράγοντας r_d είναι ανάλογος με τον αριθμό τον αποδεκτών κινήσεων, όσο μεγαλύτερος είναι τόσο λιγότερες κινήσεις θα απορρίπτονται(pruned) έχοντας καλύτερα αποτελέσματα ο αλγόριθμος αλλά απαιτώντας περισσότερο χρόνο.

r_d	1	2	4	8
mean	0	+5.6	+7.3	+9.0
time	10'	35'	90'	150'

εικόνα 36 : Πίνακας επιρροής διαφόρων τιμών του r_d

Όσο αφορά την τιμή σ_e είναι εμφανές ότι όσο πιο μικρή είναι τόσο λιγότερες είναι οι στατιστικές ισότητες άρα πιο αποδοτικός ο αλγόριθμος πληρώνοντας όμως σε ταχύτητα.

σ_e	0.2	0.5	1
mean	0	-0.7	-6.7
time	10'	9'	7'

εικόνα 37 : Πίνακας επιρροής διαφόρων τιμών του σ_e

Το επόμενο πείραμα που έγινε προς εξέλιξη των μεθόδων αξιολόγησε τον ευριστικό κανόνα All-Move-As-First Heuristic(AMAF). Όπως φαίνεται από το όνομα όλες οι διαθέσιμες κινήσεις θεωρούνται αρχικές, οπότε από την εκάστοτε υπάρχουσα κατάσταση του πλαισίου παιχνιδιού σε όσα τυχαία παιχνίδια ξεκινούν κάθε κίνηση που παίζεται θεωρείται σαν αρχική και αξιολογείται ανάλογα. Όπως αναμενόταν το μοναδικό και ουσιαστικό αποτέλεσμα και πλεονέκτημα του ευριστικού αυτού κανόνα ήταν η ταχύτητα η οποία μείωσε τον χρόνο που απαιτούσε η αρχική υλοποίηση για να βρει την βέλτιστη κίνηση κατά 95%.

Όσο αφορά όμως την αποδοτικότητα παρατηρούμε ότι δεν κέρδισε ούτε την μέθοδο Progressive Pruning ούτε την αρχική υλοποίηση, διότι είχε αρκετά καλό αποτέλεσμα μόνο για τις πραγματικά αρχικές κινήσεις κάθε προσομοίωσης.

Basic idea	PP
+13.7	+4.0

εικόνα 38 : Πίνακας αποτελεσμάτων τακτικής All-Move-As-First εναντίον Basic idea & PP

Βέβαια, η ταχύτητα του αλγορίθμου εκμεταλλευόμενη την απλότητα μας επιτρέπει να επαναλάβουμε αρκετά τυχαία παιχνίδια και έτσι να έχουμε ένα καλύτερο αλγόριθμο με αυξημένο χρόνο απόκρισης φυσικά. Παρ' όλα αυτά αν συγκρίνουμε τον All-Move-As-First Heuristic των συνολικά 10.000 κινήσεων με αυτούς τον 1.000 και 100.000 αντίστοιχα παρατηρούμε βελτίωση ανάλογη με των επιτρεπτό αριθμό επαναλήψεων όμως σε πολύ μικρό επίπεδο.

1000	**100,000**
-12.7	+3.2

εικόνα 39 : Πίνακας αποτελεσμάτων τακτικής All-Move-As-First 10000 κινήσεων εναντίον All-Move-As-First 1000 & All-Move-As-First 100000 κινήσεων

Επιπροσθέτως, στράφηκαν στον πειραματικό προσδιορισμό της κατάλληλης σταθερής θερμοκρασίας τροποποιώντας την μέθοδο Simulated Annealing. Αντικατέστησαν την λογική εκκίνησης με υψηλή θερμοκρασία, η οποία θα μειωνόταν όσο περισσότερο προχωρούσε το παιχνίδι με σταθερή τιμή. Οπότε, πλέον η πιθανότητα να επιλεχθεί κάποια κίνηση ήταν ανάλογη με $exp(Ku)$, όπου u είναι το υπάρχον βάρος - αξιολόγηση - τιμή της κίνησης και K μια σταθερά αντιστρόφως ανάλογη της θερμοκρασίας ($K = 0 \rightarrow T = \infty$).

Έτσι λοιπόν συγκρίνοντας τις διαφορές τιμές K που πρόεκυψαν πειραματικά έφτασαν στο συμπέρασμα ότι ελαφρώς θετικά αποτελέσματα παρουσιάζονται για K = 2 και K = 5 και αυτό γιατί η τιμή της θερμοκρασίας παραμένει σταθερή σε υψηλά επίπεδα οπότε όλες οι καλές κινήσεις διαλέγονται στην αρχή. Συγκρίνοντάς τα μεταξύ τους ξεκινώντας όπως ήταν λογικό από το μικρότερο καλό K διαπιστωθηκε οτι η τιμη 5 ειναι η αποδοτικότερη.

K	**0**	**5**	**10**	**20**
mean	-8.1	+2.6	-4.9	-11.3

εικόνα 40 : Πίνακας αποτελεσμάτων για K = 2, εναντίον σχεδιάσεων με διαφορετικά K

Το αρνητικό όμως είναι η λογική καθυστέρηση του αλγορίθμου, διότι απαιτούνται αρκετοί έλεγχοι και αρκετές μεταθέσεις θέσεων από την αρχική λίστα καθ' όλη την διάρκεια της προσομοίωσης του τυχαίου παιχνιδιού, διότι εφόσον η θερμοκρασία παραμένει σταθερά υψηλή αυξάνονται και οι πιθανότητες μετάθεσης ανεξαρτήτως το σημείο του παιχνιδιού.

Με σκοπό λοιπόν να ελέγξουν και να αξιολογήσουν τις προσπάθειες τους ο Bouzy έγραψε ένα πρόγραμμα, που χρησιμοποίησε την τεχνική Progressive Pruning ($r_d = 1, \sigma_e = 0.2$) και ο Helmstetter ένα που χρησιμοποιούσε τον ευριστικό κανόνα All-Move-As-First αλλά και την μέθοδο Simulated Annealing με $K = 2$ και τα έθεσαν αντιμέτωπα σε ένα πλαίσιο παιχνιδιού 9x9 μεταξύ τους αλλά και με τα γνωστικά προγράμματα, INDIGO και GNUGo. Τα ενδιαφέροντα αποτελέσματα φαίνονται στον πίνακα.

	Olga	Indigo	GNUGo
Oleg	+10.4	-4.9	+31.5
Olga		+1.8	+33.7
Indigo			+8.7

εικόνα 41 : Πίνακας αποτελεσμάτων αναμετρήσεων διαφόρων προγραμμάτων Go

Παρατηρούμε ότι απέναντι στο INDIGO και οι δύο αρχιτεκτονικές Monte Carlo συμπεριφέρονται αρκετά καλά αναλογικά με την πολυπλοκότητα και την τακτική που χρησιμοποιούν αυτά τα προγράμματα. Τέλος να σημειώσουμε ότι πραγματοποιήθηκε και μικρός αριθμός πειραμάτων και σε μεγαλύτερα πλαίσια παιχνιδιών χωρίς όμως ουσιαστικά αποτελέσματα, διότι ο χρόνος που απαιτείται ώστε να ολοκληρωθεί ένας δεδομένος αριθμός τυχαίων παιχνιδιών συμφώνα με τις αρχιτεκτονικές Monte Carlo είναι πολύ μεγαλύτερος από τον χρόνο απόκρισης και των δύο γνωστικών προγραμμάτων.

2.2.10.9 Η Άποψη του Bouzy

Ο Bouzy[11], όμως, προσπάθησε και κατάφερε να εξελίξει και άλλο το πρόγραμμα Olga, έχοντας τις γνώσεις την εμπειρία αλλά και την βοήθεια από το INDIGO φυσικά.

Οι τροποποιήσεις που έκανε ο Γάλλος ήταν ουσιαστικά δύο. Οι πιθανότητα εμφάνισης κάθε κίνησης κατά την διάρκεια των τυχαίων παιχνιδιών πλέον δεν ακολουθεί κανονική κατανομή αλλά εξαρτάται από τον υπολογισμό της αναγκαιότητάς(urgency) της και από την ταύτιση με μια 3x3 βάση δεδομένων πρότυπων κινήσεων. Η αναγκαιότητα προκύπτει από μια Συνάρτηση Αναγκαιότητας(Static Urgent Function) που αυξάνει την πιθανότητα σε κινήσεις που ενώνουν ομάδες και μειώνουν τις ελευθερίες στις ομάδες του αντιπάλου. Επιπλέον, προσέθεσε τακτική εισάγοντας την Γεννήτρια Κίνησης του INDIGO, ο οποίος τροφοδοτούσε την μονάδα Monte Carlo με συγκεκριμένο αριθμό θέσεων N_s για να καταλήξει με την καλύτερη. Όπως και στην πρώτη υλοποίηση του προγράμματος OLGA έτσι και τώρα το έθεσε αντιμέτωπο και με το INDIGO αλλά και με το GNUGo. Αυτή τη φορά όμως όχι μόνο για πλαίσιο παιχνιδιού διαστάσεων 9x9 αλλά και 13x13 και 19x19.

Board Size	9x9	13x13	19x19
mean	+18	+35	+44
% wins	76%	88%	75%
time	1'30"	10'	1h30'

εικόνα 42 : Πίνακας αποτελεσμάτων OLGA($N_s = 10, N_m = 50, r_d = 1.0, \sigma_e = 0.4$) vs. INDIGO

Παρατηρούμε ότι ανεξαρτήτως μεγέθους του πλαισίου παιχνιδιού το πρόγραμμα OLGA υπερτερεί αισθητά έναντι του INDIGO για αυτό και αρκετός χρόνος ολοκλήρωσης ενός κλασσικού παιχνιδιού δεν είναι άξιος σχολιασμού. Ο παρακάτω

πίνακας εμφανίζει πολύ σημαντικές πληροφορίες για των αριθμό αρχικών θέσεων N_s που παράγει η Γεννήτρια Κίνησης σε ένα 19x19 παιχνίδι.

N_s	2	4	7	10	15	20
mean	-7	+18	+25	+44	+56	+68
% wins	41%	59%	66%	75%	90%	91%
time	15'	40'	1h10'	1h30'	2h	2h30'

εικόνα 43 : Πίνακας επιρροής διαφόρων τιμών του N_s

Είναι εμφανές ότι ο αριθμός είναι ανάλογος της επιτυχίας και του μέσου σκορ απέναντι στο INDIGO αλλά και του χρόνου διάρκειας του παιχνιδιού, ο οποίος αυξάνεται σημαντικά για να επιτευχθούν τα παραπάνω αποτελέσματα.

	Indigo 2002			Olga		
Board size	9x9	13x13	19x19	9x9	13x13	19x19
% wins	35%	13%	6%	37%	33%	19%
Mean	-9	-34	-83	-5	-15	-40
Time	20"	1'	2'	12'	1h	5h

εικόνα 44 : Πίνακας αποτελεσμάτων GNUGo vs. INDIGO & GNUGo vs. OLGA(N_s = 10,N_m = 100,r_d = 2.0,σ_e = 0.4)

Όσο αφορά την σύγκριση του με το GNUGo, ενός από τα πιο ισχυρά και γρήγορα γνωστικά(domain - depended knowledge) προγράμματα, το οποίο αντιμετωπίστηκε και από το INDIGO παρατηρούμε ότι δυστυχώς όσο αυξάνεται το εμβαδό του πλαισίου παιχνιδιού τόσο πιο αδύναμο και αργό είναι το OLGA κερδίζοντας σχεδόν 1/3 παιχνίδια. Παρόλα αυτά να σημειωθεί ότι κερδίζει σχεδόν τα διπλάσια παιχνίδια από το INDIGO και χάνει κατά μέσο όρο με τους μισούς πόντους από ότι εκείνο. Δεν παύει όμως να είναι πολλές φορές πιο αργό στην ολοκλήρωση του παιχνιδιού.

2.2.10.10 Τακτική Αναζήτηση(Tactical Search) & Monte Carlo

Ακόμη μια ενδιαφέρουσα πρόταση για τον συνδυασμό Τακτικής Αναζήτησης και Monte Carlo σύμφωνα με τους Cazenave και Helmstetter[12] ήταν η ενσωμάτωση απλής στατιστικής σε ένα πρόγραμμα Monte Carlo Go. Με τον όρο τακτική αναζήτηση οι Γάλλοι αναφέρονταν σε αλγορίθμους αναζήτησης αιχμαλωσίας / σωτηρίας μιας ομάδας πετρών ή σύνδεσης / αποσύνδεσης δύο ομάδων. Όσο αφορά την διαδικασία προσομοίωσης Monte Carlo των κινήσεων που προέκυπταν από τις αναζητήσεις παραμένει ιδία.

Η καινοτομία που πρότειναν, υλοποίησε και είχε καλύτερα αποτελέσματα από ένα απλό πρόγραμμα Monte Carlo Go ήταν η επιλογή κίνησης με χρήση στατιστικής. Δηλαδή, για κάθε κίνηση που εξυπηρετούσε ένα στόχο ανάλογα από ποια αναζήτηση πρόεκυπτε κρατούσαν τον μέσο όρο της διαφοράς πόντων στα παιχνίδια που κέρδισε αλλά και σε αυτά που έχασε. Η κίνηση η οποία επιλέγεται εν τέλει είναι εκείνη με την μεγαλύτερη διαφορά μέσου όρου κερδισμένων μείων μέσου όρου χαμένων παιχνιδιών. Η υλοποίηση αυτή κερδίζει εκείνη που στηρίζεται σε απλή λογική Monte

Carlo κατά μέσο όρο με 52,1 πόντους ενώ η τυπική απόκλιση του σκορ είναι 34,2 πόντοι σε ένα πλαίσιο παιχνιδιού 9x9. Κατά την πραγματοποίηση των πειραμάτων και τα δύο προγράμματα έπαιζαν 10.000 παιχνίδια πριν αποφασίσουν για την καλύτερη κίνηση. Ενώ το συνδυαστικό πρόγραμμα υλοποιημένο σε ένα Celeron 1.8GHz παίζει μια κίνηση σε 10" χρόνο διπλάσιο από τον αντίπαλο του, όπως αναμενόταν, εφόσον απαιτείται και χρόνος για τις αναζητήσεις τις οποίες πραγματοποιεί.

Για να αντισταθμιστεί λοιπόν η διαφορά χρόνου επαναπραγματοποιήθηκαν οι δοκιμές με το συνδυαστικό πρόγραμμα να εκτελεί μόνο 1.000 κινήσεις πριν αποφασίσει, κάτι που το έκανε δύο φορές πιο γρήγορο τώρα. Παρόλο τον μικρότερο αριθμό παιχνιδιών κέρδιζε πάλι κατά μέσο όρο με 24,6 πόντους και τυπική απόκλιση 40 πόντους σε πλαίσιο παιχνιδιού ίσου μεγέθους με πριν. Παρατηρείται λοιπόν η ουσιαστική συνεισφορά των Τακτικών Αναζητήσεων και της στατιστικής. Για να ενδυναμώσουν αυτή την παρατήρηση οι δύο Γάλλοι καταμέτρησαν τις ικανότητες της σχεδίασης τους με τα 10.000 παιχνίδια προσομοίωσης απόφασης με το πρόγραμμα Golois σε ένα 9x9 πλαίσιο παιχνιδιού. Το Golois είναι το γνωστικό πρόγραμμα που χρησιμοποιεί ακριβώς τους ίδιους τακτικούς αλγορίθμους με το συνδυαστικό. Τα δύο προγράμματα αναμετρηθήκαν για 40 παιχνίδια όπου η τεχνική Monte Carlo ενισχυμένη με τακτική υπερίσχυσε κατά μέσο όρο με 26 πόντους.

2.2.10.11 Αλγόριθμος UCT

Ο στόχος της μεθόδου M.S.T.C. είναι να αξιολογηθούν προσεγγιστικά οι ενέργειες που γίνονται από μια συγκεκριμένη φάση (κόμβο) του παιχνιδιού (δένδρου) και έπειτα. Αυτό επιτυγχάνεται από το ασύμμετρο δένδρο που δημιουργείται μετά τις τυχαίες προσομοιώσεις που εκτελούνται. Η δημιουργία(επέκταση) του δένδρου εξαρτάται από την επιλογή των κόμβων που θα θεωρηθούν ως αρχικές καταστάσεις στις προσομοιώσεις Monte Carlo. Για αυτό η τακτική που ακολουθείται για την επιλογή των κόμβων είναι το σημαντικό και καθοριστικό στοιχείο επιτυχίας, ιδιαίτερα στο παιχνίδι Go.

Η τροποποιημένη για Δενδρικές Αναζητήσεις μορφή του αλγορίθμου UCB αφορά την επιλογή παιδιού - κόμβου j ώστε να μεγιστοποιηθεί η τιμή:

$$\mathrm{UCT} = \overline{\mathrm{X_j}} + 2\mathrm{C_p}\sqrt{\frac{2\ln \mathrm{n}}{\mathrm{n_j}}}$$

όπου n είναι ο αριθμός φορών που έχει επισκεφτεί ο κόμβος πατέρας, $\mathrm{n_j}$ είναι οι φορές που έχει επισκεφτεί το παιδί, ενώ η σταθερά $\mathrm{C_p}$(ιδανικά: $\mathrm{C_p} = 1/\sqrt{2}$), είναι εκείνη που καθορίζει το trade - off exploration vs. exploitation αφού πειραματικά αποδεδείχθηκε ότι όταν $\mathrm{C_p} > 1/2$ ευνοείται το exploration ενώ όταν $\mathrm{C_p} < 1/2$ ευνοείται το exploitation.

2.2.10.12 Βελτιώσεις M.S.T.C. Μέθοδοι

Μέχρι σήμερα, πολλά προγράμματα έχουν καταφέρει να εξελίξουν και να βελτιώσουν τις M.S.T.C. μεθόδους σε αρκετά ικανοποιητικό επίπεδο. Μεγαλύτερη επιτυχία είχε το κομμάτι των Δενδρικών Αναζητήσεων, στο οποίο υπάρχει και η μεγαλύτερη μέχρι στιγμής γνωστική βάση αλλά και χρήση σε πολλούς - ποικίλους επιστημονικούς τομείς. Για αυτό και οι αποδοτικότερες βελτιώσεις ήταν στο κομμάτι Selection.

Ένα από τα πρώτα προγράμματα το οποίο χρησιμοποίησε βελτιωμένη έκδοση των M.S.T.C. μεθόδων(συγκεκριμένα του αλγορίθμου UCT) ήταν το Mango[21]. Οι χαρακτηριστικές ιδιαιτερότητες του Mango βρίσκονται -όπως αναμένονταν- κυρίως στο στάδιο Selection του M.S.T.C.. Αυτές είναι οι τεχνικές

- **Progressive bias**
- **Progressive unpruning**

Progressive bias(PB): Είναι μια τεχνική η οποία σκοπεύει να κατευθύνει καλύτερα την δενδρική αναζήτηση σε σχέση με ευριστικούς κανόνες, οι οποίοι καταναλώνουν αρκετό χρόνο. Το αποτέλεσμα και η επιρροή της παραπάνω τακτικής εμφανίζονται, όταν έχουν παιχτεί σχετικά λίγα παιχνίδια και η εφαρμογή της κλασσικής τακτικής Selection δεν έχει και τα πιο αξιόπιστα αποτελέσματα. Συνεπώς, συμφώνα με την Progressive bias τακτική επιλέγεται ο κόμβος k, που ικανοποιεί την συνθήκη

$$k = \underset{i \in I}{\operatorname{argmax}}\left(v_i + C\sqrt{\frac{\ln n_p}{n_i}} + f(n_i)\right)$$

όπου $f(n_i) = \frac{H_i}{n_i+1}$ και H_i είναι ο συντελεστής ενός ευριστικού κανόνα του Mango που εξαρτάται μόνο από την κατάσταση του πλαισίου παιχνιδιού μετά την εφαρμογή της κίνησης του κόμβου i.

Όταν ο αριθμός των παιχνιδιών n_p ενός κόμβου είναι ίσος με 30 τότε εφαρμόζεται η κλασσική έκδοση της Selection τακτικής του M.S.T.C.. Μέχρι τότε για κάθε παιδί του κόμβου όπου $n_i = 0$, η τιμη $\sqrt{\frac{\ln n_p}{n_i}}$ αντικαθιστάται από έναν συγκεκριμένο αριθμό M, $M \gg v_i$. Επιπλέον η τιμή v_i αντικαθιστάται με το 0 όταν $n_i = 0$. Έτσι κάθε παιδί που δεν έχει επισκεφτεί επιλέγεται τουλάχιστον μια φορά και η σειρά επίσκεψης εξαρτάται μόνο από τον ευριστικό κανόνα.

Αν έχουν εκτελεστεί μόνο λίγες προσομοιώσεις (από 30 μέχρι 100 για το Mango) και η τιμή του H_i είναι υψηλή τότε ο συντελεστής $\frac{H_i}{n_i+1}$ είναι κυρίαρχος για την επιλογή επόμενης κίνησης. Ως εκ τούτου οι προσομοιώσεις που εκτελούνται μέχρι τότε βασίζονται κυρίως σε "παικτική γνώση"(Domain Knowledge), κάτι σημαντικό

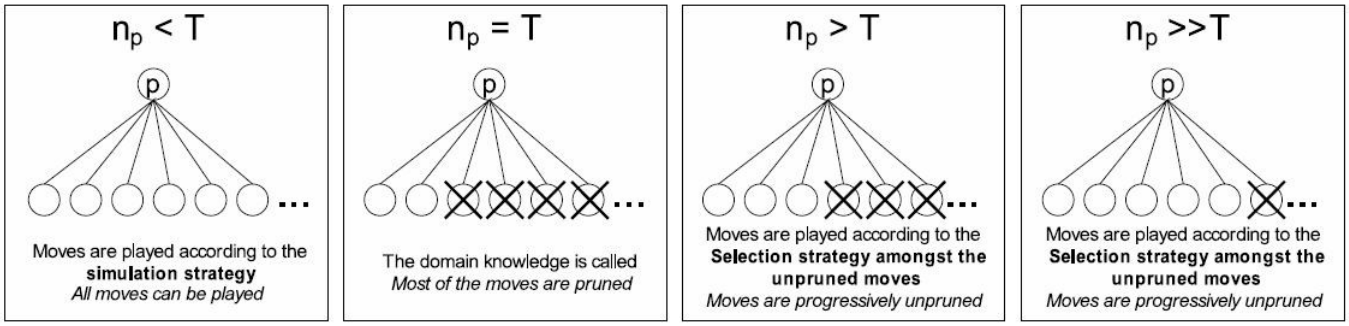
και χρήσιμο για τα προκαταρτικά στάδια, διότι οι προσομοιώσεις σε αυτό το σημείο εμπεριέχουν σε μεγάλο βαθμό αβεβαιότητα.

Όταν έχει συμπληρωθεί ένας ικανοποιητικός αριθμός επαναλήψεων(από την $100^{η}$ μέχρι την $500^{η}$ προσομοίωση για το Mango) τότε το αντίκτυπο του κανόνα και αυτό των προσομοιώσεων εξισορροπούνται ανάλογα.

Τέλος, όταν ξεπεραστεί το όριο των 500 προσομοιώσεων τότε κυριαρχεί το αποτέλεσμα των προσομοιώσεων και μειώνεται η επιρροή από την "παικτική γνώση"(Domain Knowledge). Τότε ο αλγόριθμος διαφέρει από την κλασσική έκδοση του UCT μόνο στην περίπτωση όπου δύο θέσεις στο πλαίσιο παιχνιδιού έχουν την ίδια τιμή $v(v_i = v_j)$ όπου εκεί επιλέγεται εκείνη με την καλύτερη τιμή από τον ευριστικό κανόνα.

Progressive unpruning(PU): Η τεχνική αυτή όπως φανερώνει και η ονομασία της μοιάζει να είναι αντίστροφη της Progressive Pruning. Στο πρόγραμμα Mango όπως και σε όλα σχεδόν τα Computer Go προγράμματα, όταν τελειώνει το διαθέσιμο χρονικό όριο(Timeout) για την εύρεση κίνησης και ταυτόχρονα ο συντελεστής - αριθμός διακλαδώσεων(Branching Factor) μεγαλώνει τότε η τεχνική M.S.T.C. υστερεί εμφανώς σε αποδοτικότητα.

Η εφαρμογή της δεδομένης τεχνικής έχει ως αποτέλεσμα την τεχνητή μείωση του Branching Factor όταν εφαρμόζεται η Συνάρτηση Επιλογής και την βαθμιαία αύξηση του όταν περισσότερος χρόνος γίνεται διαθέσιμος. Για αυτό λοιπόν θέτουμε ένα χρονικό όριο T. Όταν ο αριθμός προσομοιώσεων n_p για ένα κόμβο p ισούται με το T, τότε η τεχνική Progressive Pruning "κόβει" τα περισσότερα από τα παιδιά κρατώντας μόνο εκείνα $k_{init}(k_{init} = 5$ για το Mango) με τις υψηλότερες τιμές από τον ευριστικό κανόνα. k κόμβοι - παιδιά επανεπιλέγονται(unpruned) όταν ο αριθμός των προσομοιώσεων στον πατέρα - κόμβο ξεπεράσει την τιμή $A \times B^{k-k_{init}}$(πειραματικά προσδιοριστήκαν οι τιμές $A = 40$, $B = 1.3$).



εικόνα 45 : Progressive Pruning στο Mango, πηγή: [21]

Τα αποτελέσματα των δύο αυτών τεχνικών απέναντι στο GNUGo σε πειραματικό επίπεδο αλλά και απέναντι σε άλλα προγράμματα σε διαγωνισμούς είχαν εξαιρετικά αποτελέσματα όπως φαίνεται παρακάτω στους 2 χαρακτηριστικούς πίνακες.

Board Size	Simulations / Move	GNU Go's Level	PB	PU	Winning Rate	Games
19	20,000	0			0%	200
19	20,000	0		×	3.1%	200
19	20,000	0	×		4.8%	200
19	20,000	0	×	×	48.2%	500
13	20,000	10			8.5%	500
13	20,000	10		×	15.6%	500
13	20,000	10	×		30.0%	500
13	20,000	10	×	×	35.1%	500
9	20,000	10			33.2%	1000
9	20,000	10		×	37.2%	1000
9	20,000	10	×		58.3%	1000
9	20,000	10	×	×	61.7%	2000

εικόνα 46 : Πίνακας αποτελεσμάτων GNUGo vs. Mango

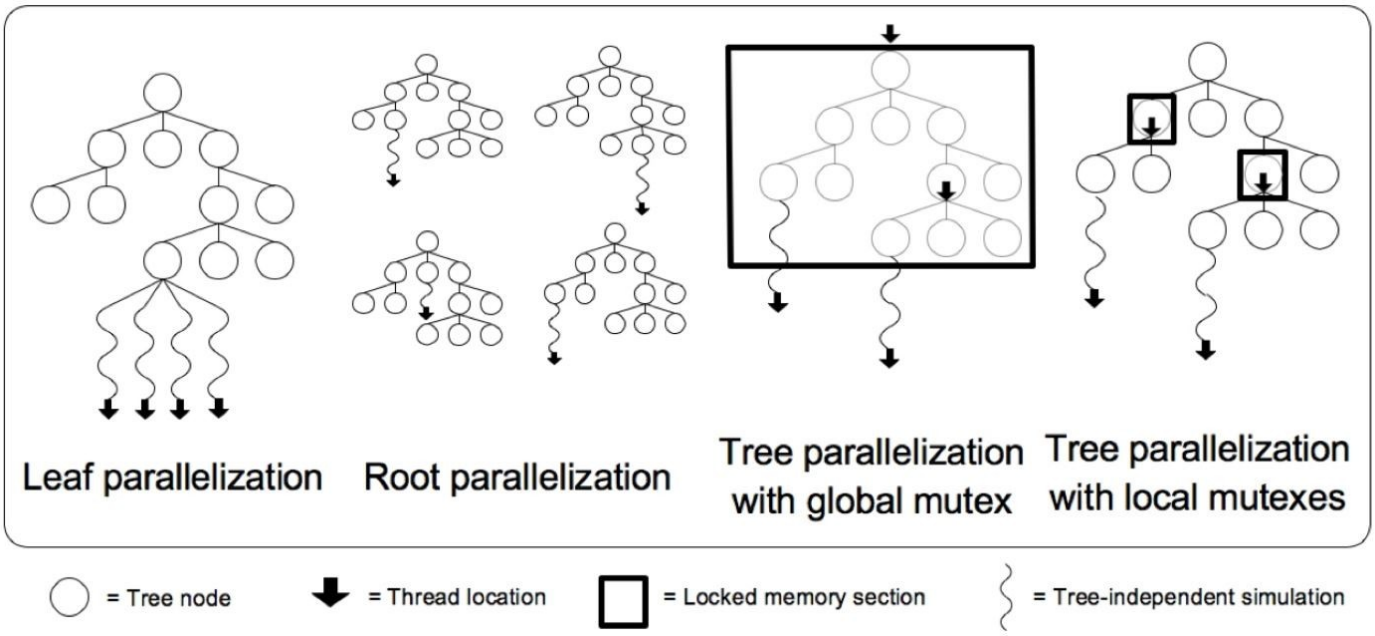
Tournament	Board Size	Participants	Mango's Rank
KGS January 2007	13x13	10	2nd
KGS March 2007	19x19	12	4th
KGS April 2007	13x13	10	3rd
KGS May 2007	13x13	7	2nd
12th Computer Olympiad	9x9	10	5th
12th Computer Olympiad	19x19	8	4th
KGS July 2007	13x13	10	4th

εικόνα 47 : Πίνακας αποτελεσμάτων του Mango σε διαγωνισμούς το 2007

Εκτός από το Mango πολλά προγράμματα [19], [23], [24] εξέλιξαν τις M.S.T.C. μεθόδους με ποικίλους τρόπους άλλοτε με μέτρια και άλλοτε με θετικά αποτελέσματα. Παρ' όλα αυτά κανένα δεν έχει καταφέρει να εξελιχτεί στο άξονα του παικτικού επιπέδου παραμένοντας στάσιμα στο επίπεδο ενός μετρίου Go Player.

2.2.10.13 Παραλληλοποίηση M.S.T.C.

Η αυτόνομη φύση κάθε προσομοίωσης στον αλγόριθμο M.S.T.C. δίνει το δικαίωμα ισχυρισμού πως είναι ένας αλγόριθμος που χρήζει παραλληλοποίησης. Η παραλληλοποίηση του αλγορίθμου έχει τα πλεονεκτήματα ότι πολλές υλοποιήσεις μπορούν να πραγματοποιηθούν ταυτοχρόνως σε δεδομένο χρόνο και ότι μπορεί να εκμεταλλευτεί πλήρως πολυ - πύρηνα και πολυ - νηματικά μηχανήματα. Βέβαια δεν λείπουν τα πρόβλημα συγχρονισμού και κατάλληλου συνδυασμού των αποτελεσμάτων που θα προκύψουν από κάθε μια παράλληλη προσομοίωση.



εικόνα 48 : MCST Parallelisation

Τα είδη παραλληλοποίησης μπορεί να θεωρηθεί πως είναι γενικά τρία:

- **Leaf Parallelisation**
- **Root Parallelisation**
- **Tree Parallelisation**

Leaf Parallelisation: Αυτού του είδους η παραλληλοποίηση αφορά την ταυτόχρονη υλοποίηση πολλών προσομοιώσεων σε ένα κόμβο, ο όποιος εξερευνείται εκείνη την στιγμή. Η βασική ιδέα είναι να συγκεντρωθούν περισσότερες πληροφορίες σε μικρότερο χρόνο για ποιοτικότερη αξιολόγηση μιας θέσης. Το εμφανές πρόβλημά της είναι ότι εξαιτίας του διαφορετικού μήκους και διαφορετικής χρονικής ολοκλήρωσης κάθε προσομοίωσης απαιτείται η αναμονή μέχρι το πέρας της πιο αργής.

Root Parallelisation: Η δημιουργία περισσότερων, αντί για ένα, δένδρων είναι μια μορφή παραλληλοποίησης του M.S.T.C.. Το δέντρο που δημιουργείται αλλά και η πληροφορίες που αποθηκεύει σαν δομή δεδομένων δεν είναι αρκετά απαιτητική σε χωρικούς περιορισμούς δίνοντας την δυνατότητα ακόμη και σε ένα απλό μέσο "υπολογιστικό σύστημα" να υλοποιήσει περισσότερα από ένα. Το κύριο πλεονέκτημα της μεθόδου αυτής είναι ότι υπάρχει η δυνατότητα να διακοπεί ανά πάσα ώρα, ασχέτως του βάθους κάθε δένδρου, αφού κάθε ένα από αυτά παρέχει πληροφορία ανά πάσα χρονική στιγμή. Το θέμα της συγκεκριμένης υλοποίησης είναι ότι δεν υπάρχει ιδανικός τρόπος αξιολόγησης της ανατροφοδότησης πληροφοριών(feedback) κάθε δένδρου.

Tree Parallelisation: Τέτοια παραλληλοποίηση αφορά την ταυτόχρονη υλοποίηση διαφορετικών βημάτων της ιδίας προσομοίωσης στο ίδιο δένδρο από διαφορετικά νήματα. Αυτός ο τρόπος απαιτεί αρκετή προσοχή στην υλοποίησή του έτσι ώστε να μην υπάρξει το φαινόμενο κάποιο νήματος(thread) να προσπελάσει κα να τροποποιήσει κάποιο ήδη δημιουργημένο και προσπελασμένο υπο-δένδρο. Μια γενική αντιμετώπιση του προβλήματος αυτού είναι με ένα ολικού(Global) Mutex σε περίπτωση που οι προσομοιώσεις διαρκούν αρκετό χρόνο και μια φτάνει στην κατάσταση ενημέρωσης των κόμβων, που έχει προσπελάσει ενώ οι άλλες διασχίζουν ακόμη το δένδρο. Επίσης μπορούν να τοποθετηθούν και τοπικά(Local) Mutexes σε κάθε κόμβο τα οποία να παρέχουν πρόσβαση σε κάθε νήμα, που ζητά να τα διασχίσει. Το πρόβλημα που προκύπτει με αυτό τον τρόπο παραλληλοποίησης είναι ότι υπάρχει μεγάλη πιθανότητα διαφορετικά νήματα να διασχίσουν το δένδρο με παρόμοιο τρόπο δίνοντας ίδια, άρα άχρηστα, αποτελέσματα. Λύση στο πρόβλημα δίνει η εικονική "κακή" αξιολόγηση του κόμβου από το πρώτο νήμα, που θα τον επισκεφτεί, ώστε τα επόμενα να επιλέξουν απροσπέλαστους κόμβους να εξερευνήσουν.

2.2.11 Πρότυπα(Patterns)

Πρόσφατα, εντάθηκε η τεχνική Pattern Recognition στην ανάπτυξη των προγραμμάτων Go. Ερευνητές παρατήρησαν κορυφαίους παίχτες μελετώντας τον τρόπο παιχνιδιού τους και καταγράφοντας κλασσικές κινήσεις τους. Επίσης πολλά προγράμματα, τα οποία χρησιμοποιούσαν τεχνικές Monte Carlo δημιούργησαν και τις δικές τους πρότυπες κινήσεις[13] κατά την διάρκεια των τυχαίων παιχνιδιών. Κινήσεις με υψηλό αντίκτυπο στην αύξηση του σκορ, ή στην μείωσή του αποθηκευτήκαν ως πρότυπα προς επιλογή ή αποφυγή αντίστοιχα. Πρωτοπόρο στην εφαρμογή της τεχνικής αυτής ήταν το πρόγραμμα Goliath, δύο φορές συνεχόμενος παγκόσμιος πρωταθλητής του Computer Go Championship μετά την εφαρμογή των μονάδων δημιουργίας και ανίχνευσης πρότυπων κινήσεων.

Κεφάλαιο 3

Μοντελοποίηση

3.1 Εισαγωγή

Όπως έγινε εμφανές και στο προηγούμενο κεφάλαιο, κύριο και κοινό χαρακτηριστικό όλων των προγραμμάτων και υλοποιήσεων μετά την εμφάνιση της μεθόδου M.S.T.C., είναι η τυχαία προσομοίωση ολοκλήρωσης του παιχνιδιού ξεκινώντας από μια δεδομένη κατάσταση. Όσο περισσότερες φορές και όσο πιο γρήγορα μπορεί να επιτευχτεί το παραπάνω με επιτυχία τόσο λιγότερη αβεβαιότητα θα εμπεριέχει η αξιολόγηση κάθε κίνησης και τόσο πιο ορθολογικά θα αναπτύσσεται το δένδρο.

Για αυτό λοιπόν, βασικός στόχος της διπλωματικής αυτής ήταν η υλοποίηση ενός πλαισίου παιχνιδιού Go σε μια FPGA, το οποίο θα προσομοίωνε ένα ολόκληρο παιχνίδι αποδοτικά, αποτελεσματικά άλλα και όσο το δυνατόν πιο γρήγορα. Η σχεδίαση, που υλοποιήθηκε είχε σαν στόχο την επέκταση και βελτίωση της επιτυχημένης προσπάθειας του Σ. Κόκκαλη[22] -απόφοιτο του Τμήματος Ηλεκτρονικών Μηχανικών και Μηχανικών Υπολογιστών του Πολυτεχνείου Κρήτης- το 2006.

Η προσπάθεια, η οποία έγινε, επιβεβαίωσε πλήρως την παρακάτω δήλωση του γνωστού Γάλλου, B. Bouzy με εξαιρετική ερευνητική και όχι μόνο συνεισφορά στο πεδίο του παιχνιδιού Go.

"*The difficulty lies in performing introspection. New knowledge interacts with existing knowledge in unpredictable ways. Whenever a programmer tries to improve his program by adding knowledge relating to a particular sub - problem, this new knowledge often interacts with other knowledge in another part of the program, and*

finally produces bad results. Furthermore, even if the programmer is a very good Go player, he has difficulties in finding rules without exceptions. He often inserts new rules, forgetting the exceptions, and produces bad results again."

Αρχικά η επίλυση προβλημάτων που προέκυπταν για την βελτίωση και κατά κάποιο τρόπο ενίσχυση της σχεδίασης είχε θετικό πρόσημο. Όμως τελικά η ελπίδα που δημιουργούσε ήταν αρκετή για την εφαρμογή της τελευταίας ενίσχυσης του προγράμματος, η οποία μείωσε την συχνότητα λειτουργιάς του ρολογιού κοντά στα 10 MHz.

Αρχικά, όμως, πρέπει να αναφέρουμε τα βασικά στοιχεία της σχεδίασης του Σ. Κόκκαλη, που αποτελούν εφαλτήριο αυτής της διπλωματικής εργασίας.

3.2 Προηγούμενη Σχεδίαση Υλοποίησης ενός Πλαισίου Παιχνιδιού Go

3.2.1 Βασική Ιδέα

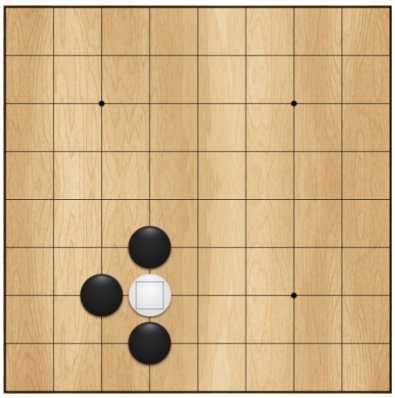
Κύριος στόχος της σχεδίασης ήταν η υλοποίηση ενός πλαισίου παιχνιδιού Go, το οποίο θα δέχονταν κινήσεις από μια εξωτερική μονάδα και θα τις εφάρμοζε. Αρχικά, θα έλεγχε τις κινήσεις για την εγκυρότητα τους και έπειτα θα τις εφάρμοζε.

Συμφώνα με τους κανόνες του παιχνιδιού, αν η τοποθέτηση πέτρας σε ένα Eye καταλήγει στην αιχμαλώτιση αντίπαλων πετρών τότε επιτρέπεται η τοποθέτηση της. Συνεπώς, υπάρχει περίπτωση επανατοποθέτησης πέτρας - πετρών σε θέση - θέσεις του πλαισίου παιχνιδιού, πράγμα που δυσκολεύει τον καθορισμό του τέλους του παιχνιδιού. Για αυτό, συνήθως, απαγορεύεται η επανατοποθέτηση πέτρας σε θέση Eye.

Οπότε σύμφωνα με την προηγούμενη διπλωματική εργασία αντικανονικές θεωρούνται οι εξής ενέργειες:

- Τοποθέτηση πέτρας εκτός ορίων του πλαισίου παιχνιδιού.
- Τοποθέτηση πέτρας σε ήδη κατειλημμένη θέση.
- Τοποθέτηση πέτρας σε θέση Eye.
- Κίνηση αυτοκτονίας.

Με τον όρο κίνηση αυτοκτονίας ορίζεται η τοποθέτηση πέτρας σε θέση που έχει μια ελευθερία και περιβάλλεται από αντίπαλου χρώματος πέτρες.



εικόνα 49 : Κίνηση Αυτοκτονίας

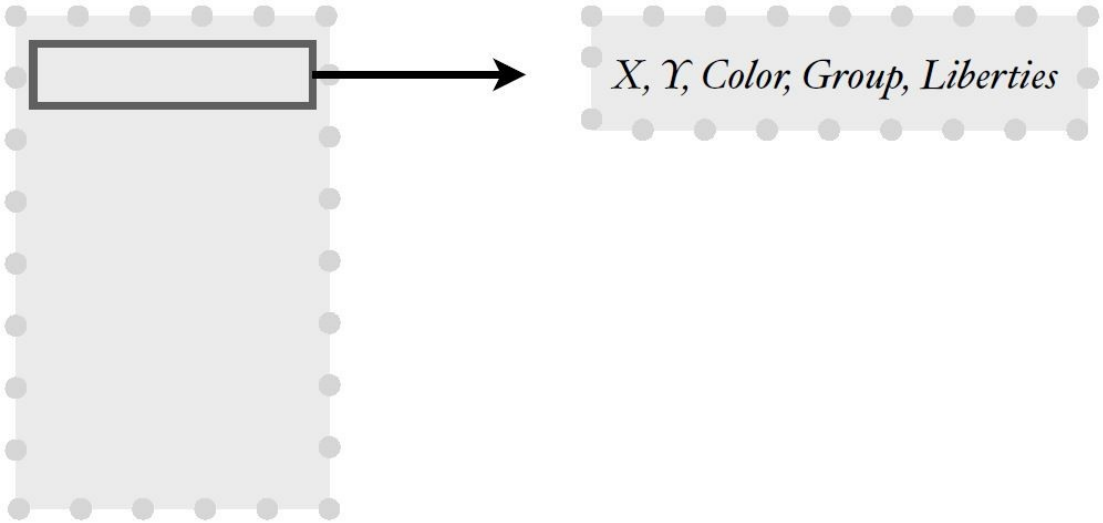
Όπως φαίνεται και στην παραπάνω εικόνα, η τοποθέτηση τις άσπρης πέτρας αποτελεί κίνηση αυτοκτονίας, διότι ο παίχτης με τις μαύρες μπορεί άμεσα να την αιχμαλωτίσει.

Η σχεδίαση είχε την δυνατότητα:

- να τοποθετεί πέτρες στις θέσεις του πλαισίου παιχνιδιού και να ανακτά ανά πάσα στιγμή πληροφορίες για αυτές.
- να αναγνωρίζει και να αφαιρεί τις αιχμάλωτες πέτρες.
- να ειδοποιεί για αντικανονικές κινήσεις.
- να ενημερώνει για το τέλος του παιχνιδιού.

Κάθε νέα θέση, λοιπόν, που πληροί τις προϋποθέσεις τοποθετείται στο πλαισίου παιχνιδιού και τροποποιεί έναν καταχωρητή - μια θέση μνήμης ενημερώνοντας με τα στοιχεία της. Τα κύρια χαρακτηριστικά, που πρέπει να είναι αποθηκευμένα σε κάθε θέση του πλαισίου παιχνιδιού είναι:

- οι συντεταγμένες της θέσης.
- το χρώμα της πέτρας που βρίσκεται στη θέση.
- η ομάδα στην οποία ανήκει η τοποθετημένη πέτρα.
- ο αριθμός των ελευθεριών για κάθε πέτρα



εικόνα 50 : Στιγμιότυπο μιας θέσης του πλαισίου παιχνιδιού, πηγή: [22]

Οπότε, κάθε νέα τοποθέτηση οφείλει να ενημερώνει για την επιρροή που ασκεί στο πλαισίου παιχνιδιού αλλά και στις γειτονικές πέτρες.

3.2.2 Αρχική Υλοποίηση

Η πρώτη σχεδίαση, που υλοποιήθηκε περιέγραφε την υλοποίηση ενός παραμετροποιήσιμου πλαισίου παιχνιδιού ανάλογα με τις επιθυμητές διαστάσεις. Είτε 9x9, είτε 13x13, είτε 19x19. Κάθε θέση του πλαισίου παιχνιδιού περιλαμβάνει κατάλληλο χώρο μνήμης για την αποθήκευση των απαραιτήτων πληροφοριών αλλά και μια μονάδα, που αναλαμβάνει το ρόλο ενός "επεξεργαστή". Αυτή η μονάδα είναι υπεύθυνη για δύο ενέργειες:

- να δέχεται και να επεξεργάζεται κάθε εισερχόμενο μήνυμα, είτε αφορά τοποθέτηση νέας πέτρας, είτε τροποποίηση κάποιας από τις τιμές που είναι αποθηκευμένες, εξαιτίας κάποιας τοποθέτησης που ασκεί επιρροή.
- να διαβάζει την κατάσταση των τεσσάρων γειτόνων και να εκπέμπει προς όλες τις υπόλοιπες θέσεις - επεξεργαστές τα ανάλογα μηνύματα.

Οι βασικές ενέργειες, που πρέπει να καλυφτούν είναι η τοποθέτηση πέτρας σε μια νέα θέση, η ενημέρωση των γειτόνων ανάλογα με την ασκούμενη επιρροή(δημιουργία ομάδας πετρών, αιχμαλώτιση, μείωση αριθμού ελευθεριών) αλλά και η εκτύπωση των πληροφοριών κάποιας θέσης οποτεδήποτε χρειαστεί.

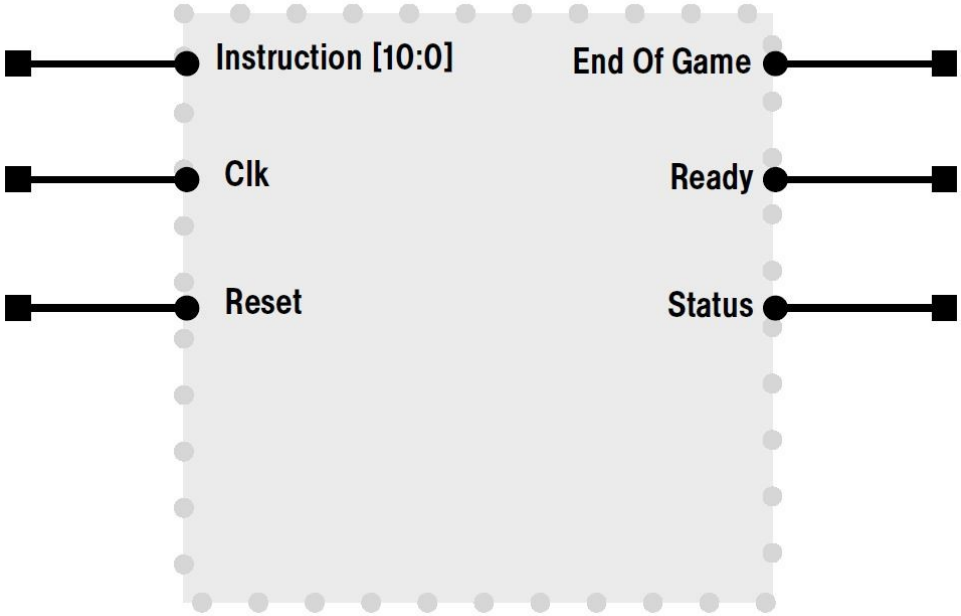
Έτσι, λοιπόν, δημιουργήθηκε ένα σύνολο 6 εντολών(Instruction Set), ώστε να καλυφτούν όλες οι ανάγκες όπως περιγράφεται στον παρακάτω πίνακα.

Instruction	Description	Example
reset	resets the entity to an empty position state. comes from external module	reset
new move	sends a new move tuple to all entities. comes from external module	[newmove, 3 , 3 , black]
merge	the group receiving this message changes its group id to the message's group id	[merge , 1 , 2 , 6] (group 1 will become 2 with 6 liberties)
modify	modifies all attributes of an entity	[modify , 3 , 3 , 1 , 10] (modify entity 3, 3 to group 1 and 10 liberties)
modify liberties	increase or decrease the liberties of a group by a number	[modify , 11 , -1] (reduce group's 11 liberties by one)
print	prints the entity's attributes to the standard output	[print , 3 , 3]

εικόνα 51 : Πίνακας εντολών

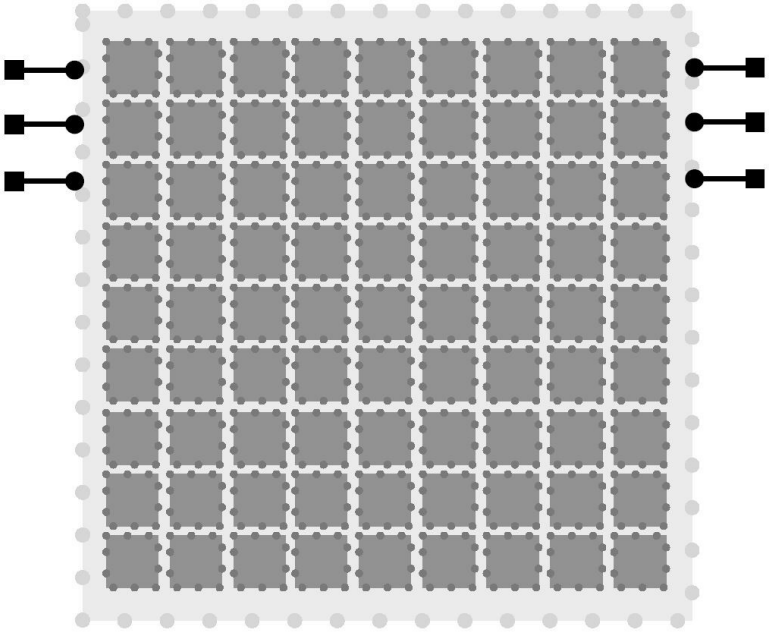
Συνεπώς, το μόνο, που πρέπει να προσέξουμε και να συντονίσουμε είναι το σήμα που θα στέλνεται στην μονάδα δημιουργίας επόμενης κίνησης να φτάνει αφού έχουν πραγματοποιηθεί και όλες οι αλλαγές αλλά και τα bit που απαιτούνται για την αποθήκευση κάθε πληροφορίας. Οι εξισώσεις υπολογισμού τους περιγράφονται ιδανικά από τον Σ. Κόκκαλη[22].

Η παραπάνω σχεδίαση υλοποιήθηκε με το εργαλείο Xilinx IDE σε γλώσσα VHDL με το Ανώτατο Επίπεδο της αρχιτεκτονικής να είναι:



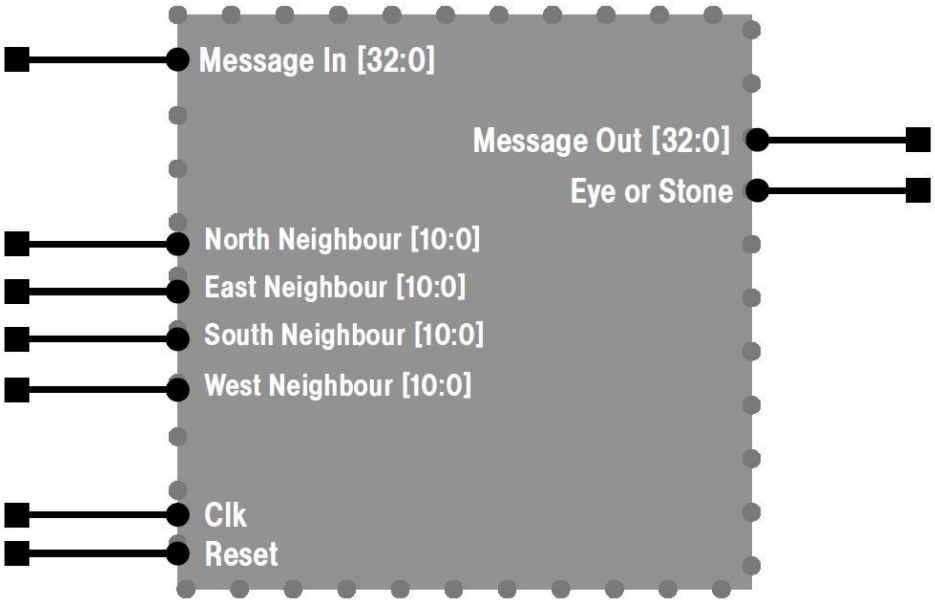
εικόνα 52 : Ανώτατο Επίπεδο αρχικής σχεδίασης Σ. Κόκκαλη, πηγή: [22]

και για το πλαίσιο παιχνιδιού διαστάσεων 9x9 το εσωτερικό του Ανώτερου Επιπέδου ήταν κάπως έτσι:



εικόνα 53 : Οι 81 θέσεις στο εσωτερικό του πλαισίου παιχνιδιού, πηγή: [22]

Με το γενικευμένο Σχεδιαστικό Διάγραμμα(Block Diagram) κάθε θέσης του πλαισίου παιχνιδιού να είναι το παρακάτω:



εικόνα 54 : Σχεδιαστικό Διάγραμμα της θέσης στο πλαίσιο παιχνιδιού, πηγή: [22]

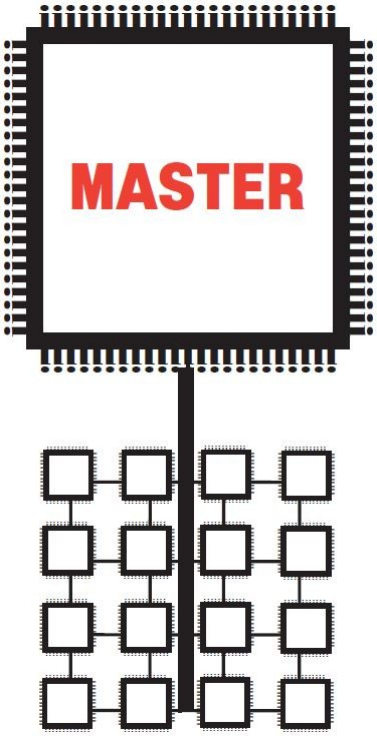
όπου παρατηρούμε της πληροφορίες που λαμβάνει από τις τέσσερις γειτονικές θέσεις ώστε να διαμορφώσει το κατάλληλο μήνυμα προς εκπομπή.

Η ύπαρξη, όμως, της μονάδας ελέγχου επεξεργασίας και εκπομπής μηνυμάτων σε κάθε θέση του πλαισίου παιχνιδιού δημιούργησε χωρικό πρόβλημα. Ενώ η σωστή λειτουργία της σχεδίασης επιβεβαιώθηκε με το κατάλληλο τεστ παρατηρήθηκε ότι η μικρότερου μεγέθους σχεδίαση(για το 9x9 πλαίσιο παιχνιδιού) κατέλαβε το 240% των πόρων μιας μεσαίου μεγέθους FPGA(Virtex 5 110T). Συνεπώς, πάρα την αναμενόμενα γρήγορη υλοποίηση ενός πλαισίου παιχνιδιού Go, που αντικειμενικά θα προσέφερε η παραπάνω σχεδίαση έπρεπε να τροποποιηθεί.

3.2.3 Τελική Σχεδίαση - Υλοποίηση

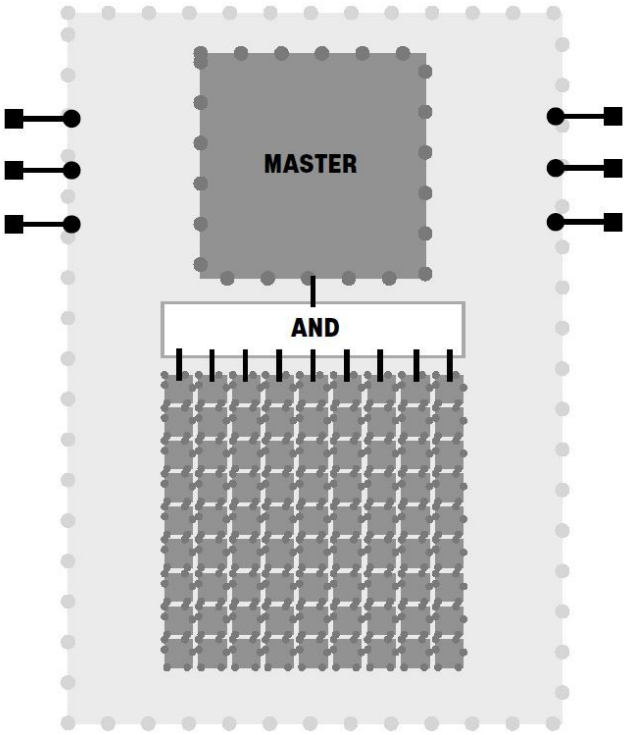
Έχοντας σχεδιάσει, υλοποιήσει αλλά και ελέγξει μια μονάδα, η οποία ανάλογα τα γειτονικά δεδομένα εκπέμπει τα σωστά μηνύματα για την ανανέωση του πλαισίου παιχνιδιού δεν υπήρχε λόγος να αλλάξει ή να δημιουργηθεί κάτι καινούριο.

Απομονώνοντας την παραπάνω μονάδα από κάθε θέση, λόγω του χωρικού προβλήματος που δημιουργούσε, και εφαρμόζοντας την σκέψη να υπάρχει μια, η οποία να εξυπηρετεί όλες τις θέσεις ανάλογα με την ζήτηση πρόεκυψε μια νέα αρχιτεκτονική. Η νέα σχεδίαση θα μπορούσε κάλλιστα να χαρακτηριστεί Master - Slave αρχιτεκτονική



εικόνα 55 : Master - Slave Αρχιτεκτονική, πηγή: [22]

με την μονάδα επεξεργασίας και εκπομπής των μηνυμάτων να είναι ο Master - Επεξεργαστής και οι θέσεις του πλαισίου παιχνιδιού να αποτελούν τις μονάδες - Slaves. Συνεπώς το νέο εσωτερικό του πλαισίου παιχνιδιού θα είναι έτσι:



εικόνα 56 : Το αναθεωρημένο(Master - Slave) εσωτερικό του πλαισίου παιχνιδιού, πηγή: [22]

Όπως ήταν λογικό το πρόβλημα χώρου λύθηκε, μάλιστα με εντυπωσιακά αποτελέσματα. Πλέον το 1/4 των πόρων της FPGA ήταν αρκετό για τις διαστάσεις 9x9, ενώ το πλαίσιο παιχνιδιού των 361 θέσεων απαιτούσε μόνο 15% επιπλέον χώρο. Βέβαια, όσο αφορά τις επιδώσεις είναι βέβαιο ότι θα ήταν τουλάχιστον δύο φορές πιο αργό. Αν η προηγούμενη σχεδίαση απαιτούσε ένα κύκλο ώστε ο επεξεργαστής να αποφασίσει τι μήνυμα θα εξέπεμπε, τώρα απαιτείται ένας κύκλος να φτάσουν τα δεδομένα μέσω του διαύλου στον Master - επεξεργαστή και άλλο ένα να επιστρέψει το κατάλληλο μήνυμα. Επιπλέον, αν αναλογιστούμε ότι όλη η πληροφορία από τις θέσεις - slaves φτάνει στη "κεντρική" μονάδα επεξεργασίας και εκπομπής μηνυμάτων δημιουργώντας αρκετό Latency αναμένεται επιπλέον μείωση της συχνότητας λειτουργιάς.

Αναλύοντας την προσέγγιση του Σ. Κόκκαλη και προσχεδιάζοντας τις βελτιώσεις που σκόπευα να εφαρμόσω το μοντέλο Master - Slave είναι εκείνο στο οποίο βασίστηκα για την σχεδίαση μου.

3.3 Ανάλυση και Διαφοροποίηση της Υλοποιημένης Αρχιτεκτονικής

Τα τρία κύρια χαρακτηριστικά, τα οποία προστεθήκαν ήταν:

- Γεννήτρια Τυχαίων Αριθμών
- Μονάδα Μέτρησης Σκορ
- Μονάδα Αναγνώρισης Πρότυπων Κινήσεων 3x3

με αρχικό και κύριο στόχο την βελτίωση και ενίσχυση της διαδικασίας προσομοίωσης τυχαίων παιχνιδιών αλλά και δευτερεύον την υλοποίηση μιας μονάδας, η οποία θα ήταν ικανή να απαντά με κίνηση σε μια υπάρχουσα κατάσταση του πλαισίου παιχνιδιού.

Έτσι, λοιπόν υλοποιήθηκε οτιδήποτε από την αρχή, δημιουργώντας με μικροδιαφορές σε κάποιες μονάδες ένα πλαίσιο παιχνιδιού μεγέθους 9x9, το οποίο βρίσκει δύο υποψήφιες κινήσεις και παίζοντας ένα τυχαίο παιχνίδι για την καθεμία επιλέγει εκείνη με το υψηλότερο σκορ στο τέλος του παιχνιδιού. Χρησιμοποιώντας ως σύμμαχο την προηγούμενη σχεδίαση αλλά και το αυξανόμενο καθημερινό ενδιαφέρον για το παιχνίδι υλοποιήθηκε μια σχεδίαση, η οποία υπακούει σε όλους του κανόνες -πλην εκείνου του ko- του παιχνιδιού αλλά και τους περιορισμούς, που έχουν τεθεί για την λίγο "εξυπνότερη" λειτουργία. Με χρήση του εργαλείου Xilinx ISE και με την δημιουργία τεστ ελέγχτηκε η ορθότητα της σχεδίασης. Επίσης είναι πολύ σημαντικό να τονιστεί ότι είναι εξαιρετικά δύσκολο να χαρακτηριστεί με πλήρης επιτυχία η πρώτη προσπάθεια πάνω σε κάτι, διότι μέχρι την επίτευξη του

στόχου πρέπει να λυθούν και να αντιμετωπιστούν αρκετά προβλήματα, που προκύπτουν αναπάντεχα για να επιβεβαιώσουν την δήλωση του Bouzy παραπάνω.

3.3.1. Γεννήτρια Τυχαίων Αριθμών

Ο λόγος υλοποίησης και ύπαρξης αυτής της μονάδας είναι προφανής. Στόχος της είναι να αναλαμβάνει το ρόλο των παιχτών στην τυχαία προσομοίωση του παιχνιδιού. Σύμφωνα με την μέθοδο Monte Carlo η κατανομή που θα πρέπει να ακολουθείται είναι κανονική, ώστε να μην αμφισβητείται η τυχαιότητα.

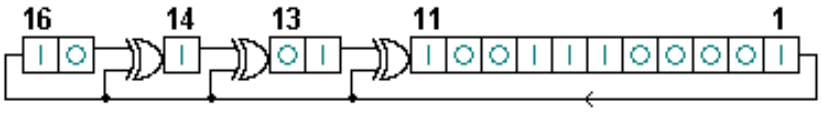
Όμως, η τυχαιότητα μπορεί κάποιες στιγμές να δημιουργεί πρόβλημα. Στην προκειμένη περίπτωση, εφόσον οι παίκτες παίζουν εναλλάξ το μόνο που απαιτείται από την στιγμή τοποθέτησης μιας πέτρας είναι η τυχαία επιλογή της επόμενης θέσης στην οποία θα τοποθετηθεί πέτρα του αντίθετου χρώματος από εκείνο της προηγούμενης. Δηλαδή, δύο αριθμούς που να καθορίζουν πως θα κινηθούμε στον άξονα x και y αντίστοιχα. Άρα, μια γεννήτρια, η οποία θα ακλουθούσε κανονική κατανομή γεννώντας νούμερα από τον 1 έως το 9 για την x διάσταση και μια με παρόμοιο τρόπο για την y διάσταση θα ήταν αρκετές. Δυστυχώς, όμως, έτσι μένει εκτός εύρους τιμών και περιπτώσεων η κίνηση "πάσο" που έχει και αύτη πιθανότητα εμφάνισης αλλά και νόημα επιλογής κάποιες στιγμές του παιχνιδιού. Ακόμη, ο παραπάνω τρόπος υπάρχει δυνατότητα να δημιουργήσει κάποιες συντεταγμένες, οι όποιες είναι ήδη κατειλημμένες. Ιδανική περίπτωση θα ήταν εκείνη κατά την οποία θα επιλέγονταν τυχαία και ακλουθώντας φυσικά κανονική κατανομή οι εναπομείναντες διαθέσιμες θέσεις του πλαισίου παιχνιδιού.

Στην περίπτωση ενός πλαισίου παιχνιδιού 81 θέσεων μια γεννήτρια κανονικής κατανομής μεταβλητού εύρους τιμών το οποίο να μειώνεται κατά μια θέση για κάθε επιλεγμένη κίνηση αλλά και η ύπαρξη μιας μονάδας, που να λειτουργεί ως λίστα, ώστε κάθε επιλεγμένη κίνηση απλά να αφαιρούνταν χωρίς να αφήνει κενή θέση θα έδιναν την λύση. Όμως, πόσο γρήγορη σε σχέση με την παραπάνω υλοποίηση θα ήταν αυτή; Πόσο μεγαλύτερο είναι το κόστος σε πόρους και η πολυπλοκότητα της μονάδας ελέγχου και απόρριψης μη έγκυρων κινήσεων, από την ιδανική λύση. Επειδή, ο έλεγχος εάν μια τιμή βρίσκεται εντός κάποιων ορίων είναι απλός, αλλά και ο έλεγχος αν μια θέση είναι κατειλημμένη εξαρτάται από ένα bit προτιμήθηκε η πρώτη λύση.

Ο έλεγχος για την ένταξη της τιμής της συντεταγμένης εντός κάποιων ορίων πρόεκυψε, διότι κατά την μεταφορά της σχεδίασης στο Hardware αυτό, που υπήρχε δυνατότητα να υλοποιηθεί και έγινε ήταν μια γεννήτρια κανονικής κατανομής όλων των δυνατών συνδυασμών που δημιουργούνται από n bits. Έτσι, όταν απαιτούνται 4 για την απεικόνιση του αριθμού 9, της μεγαλύτερης τιμής συντεταγμένης, η γεννήτρια θα δημιουργεί αριθμούς από το 0 → "0000" μέχρι το 15 → "1111", 7 επιπλέον τιμές, που δεν αντιπροσωπεύουν θέση του πλαισίου παιχνιδιού.

Η υλοποίηση της γεννήτριας τυχαίων αριθμών έγινε με την σχεδίαση ενός Linear Feedback Shift Register(L.F.S.R.). Ο L.F.S.R. είναι ένας καταχωρητής ολίσθησης, του οποίου το bit εισόδου αποτελεί γραμμική συνάρτηση της προηγούμενης κατάστασης. Η πιο συνηθισμένη και ευρέως χρησιμοποιούμενη έκδοση είναι εκείνη στην οποία η είσοδος στον καταχωρητή "καθοδηγείται" από το XOR ορισμένων συγκεκριμένων bit. Η λειτουργία του καταχωρητή είναι περιοδική και ντετερμινιστική. Εφόσον η τωρινή κατάστασή του εξαρτάται από την προηγούμενη κατάσταση και εφόσον οι δυνατές προηγούμενες καταστάσεις είναι ένας μετρίσιμος αριθμός, άρα και οι επόμενες θα είναι εξίσου μετρίσιμος και συγκεκριμένος. Έτσι με χρήση κατάλληλης Συνάρτησης Ανατροφοδότησης(Feedback Function), δηλαδή σωστή επιλογή των bit που θα περνούν μαζί με την έξοδο μέσα από XOR πύλες για να ανατροφοδοτούν την είσοδο, μπορεί να επιτευχτεί μέγιστη περίοδος και έτσι παρέχεται η δυνατότητα χρήσης του καταχωρητή σαν pseudo - random Number Generator.

Η περίοδος ενός L.F.S.R., που χρησιμοποιεί κατάλληλη Συνάρτηση Ανατροφοδότησης, εξαρτάται και από το μήκος n απο την σχεση $2^n - 1$. Έτσι χρησιμοποιήθηκε Register 168 bit, ο μεγαλύτερος δυνατός για τον οποίο υπάρχει κατάλληλη συνάρτηση. Συγκεκριμένα υλοποιήθηκε L.F.S.R. τύπου Galois όπως αυτός των 16 - bit της εικόνας



εικόνα 57 : 16 - bit Galois L.F.S.R., πηγή: L.F.S.R. Wiki

με τις XOR πύλες να τοποθετούνται μετά από τα bit 168, 166, 153, 151 και χρησιμοποιώντας τυχαία 8 bit του καταχωρητή για τις συντεταγμένες x, y.

3.3.2 Μονάδα Μέτρησης Σκορ

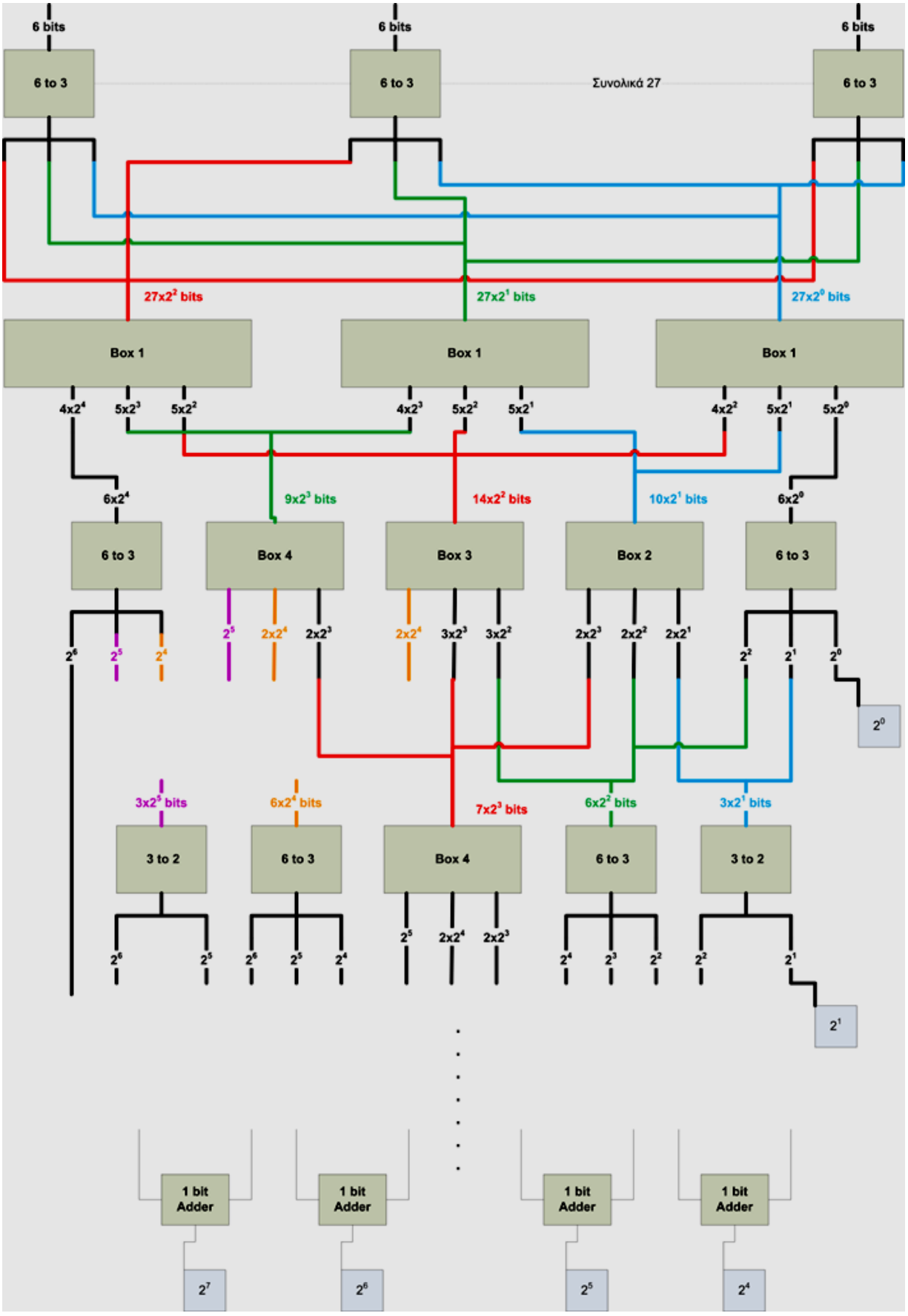
Η μονάδα μέτρησης σκορ, που δημιουργήθηκε έχει δύο λόγους ύπαρξης. Αρχικά, αναλαμβάνει την μέτρηση του σκορ στο τέλος του παιχνιδιού, ώστε να ανακοινωθεί το τελικό αποτέλεσμα και δευτερευόντως αποτελεί μέρος της Συνάρτησης Αξιολόγησης του προγράμματος. Στο τέλος της τυχαίας συνέχισης του παιχνιδιού, με δεδομένη τοποθέτηση πέτρας στις υποψήφιες θέσεις καταμετρά το σκορ για κάθε υποψήφια κίνηση, το οποίο είναι κριτήριο για την επιλογή της καλύτερης ή της λιγότερο κακής κίνησης. Για το λόγο αυτό κωδικοποιήθηκαν οι πιθανές καταστάσεις κάθε θέσης του πλαισίου παιχνιδιού με τον εξής τρόπο:

Κατάσταση	Κωδικοποίηση
Άδειο	"10"
Άσπρη Πέτρα	"11"
Μαύρη Πέτρα	"00"

εικόνα 58 : Πίνακας κωδικοποίησης πιθανών καταστάσεων κάθε πέτρας

Η παραπάνω κωδικοποίηση προσφέρει την δυνατότητα χρήσης Reduction Tree για τον υπολογισμό του σκορ. Δεδομένου ότι στην αρχή του παιχνιδιού έχουμε 81 άσσους και ότι η τοποθέτηση άσπρης πέτρας αυξάνει τον αριθμό των άσσων κατά 1 και μαύρης τον μειώνει εξίσου, στο τέλος του παιχνιδιού ο αριθμός, ο οποίος θα προκύψει υποδηλώνει το σκορ. Αν είναι μεγαλύτερος του 81 σημαίνει περισσότερες άσπρες πέτρες όση η διάφορα, ενώ αν είναι μικρότερος ισχύει το ίδιο για της μαύρες. Σε περίπτωση ίσου αριθμού, προφανώς το αποτέλεσμα είναι 81.

Η δομή του Reduction Tree που χρησιμοποιήθηκε βασίζεται στο γεγονός ότι το μέγιστο μήκος bits των L.U.T. της επιλεγμένης FPGA ήταν 6 bits. Συνεπώς χρησιμοποιήθηκαν "αποκωδικοποιητές" 6 to 3 και 3 to 2. Η δομή αυτή πλεονεκτεί σε ταχύτητα και πολυπλοκότητα, απέναντι σε εκείνη χρήσης αθροιστών, ώστε να μετρηθούν οι άσσοι, όμως απαιτεί προσοχή στην επιλογή των bits μετά από κάθε στάδιο μείωσης(reduction). Πρέπει να ομαδοποιηθούν προς αποκωδικοποίηση τα bits μαζί με εκείνα της ίδιας τάξης μεγέθους. Κάθε "αποκωδικοποιητής", απλά ανάλογα των αριθμό άσσων στην είσοδο δίνει στην έξοδο των αριθμό αυτό.



εικόνα 59 : Reduction Tree

Όπως φαίνεται και στην εικόνα στην περίπτωση, που τα bits προς αποκωδικοποίηση είναι περισσότερα των 6 χρησιμοποιούμε την μονάδα Box. Η κάθε μονάδα Box αποτελείται απλά από τον ελάχιστο αριθμό 6 to 3 L.U.T. και 3 to 2 L.U.T. και από τον βέλτιστο συνδυασμό αυτών ώστε να αποκωδικοποιηθούν. Επιπλέον, την στιγμή που το δένδρο φτάνει στο επίπεδο, το οποίο απαιτεί μόνο 3 to 2 L.U.T. για αποκωδικοποίηση στο επόμενο επίπεδο τοποθετείται ο απαραίτητος αριθμός αθροιστών 1 bit και έτσι σχηματίζεται το αποτέλεσμα. Προφανώς, από το σχήμα έχουν αφαιρεθεί, εκτός από κάποιες ίδιες μονάδες και κάποια αυτονόητα επίπεδα, και οι ενδιάμεσοι καταχωρητές που υπάρχουν για την βελτίωση της συχνότητας του ρολογιού.

3.3.3 Μονάδα Αναγνώρισης Προτύπων Κινήσεων 3x3

Έχουν συλλεχτεί και αξιολογηθεί πολλών μεγεθών και σχημάτων πρότυπες κινήσεις από παιχνίδια υψηλού επιπέδου παιχτών. Αυτές με την μεγαλύτερη επιτυχία και αποτελεσματικότητα είναι του μεγέθους 3x3 όσο αφορά πλαίσιο παιχνιδιού διαστάσεων 9x9. Η απεικόνιση μιας πρότυπης κίνησης γίνεται συνήθως κάπως έτσι:

```
202121202          TAG
 1| . O .
 2| # _ #
 3| . O .
pat number  1:  19 16 score = 84.210526
```

Το TAG απεικονίζει κάθε σειρά της πρότυπης κίνησης την μια δίπλα στην άλλη, όχι όμως με σύμβολα αλλά με τους αριθμούς με τους οποίους έχει κωδικοποιηθεί κάθε δυνατό σύμβολο. Συνεπώς το TAG μπορεί να θεωρηθεί και το σημαντικότερο κομμάτι μιας πρότυπης κίνησης, αφού βοηθά στην αναγνώριση και ταύτισή της. Μετά το TAG ακολουθεί η σχηματική απεικόνιση της 3x3 πρότυπης κίνησης πάλι με τα κατά σύμβαση σύμβολα που αντιπροσωπεύουν δυνατές πραγματικές καταστάσεις. Τέλος, υπάρχει η αξιολόγηση κάθε πρότυπης κίνησης, που αποτελείται από τον αριθμό που εμφανίστηκε, τον αριθμό που επιλέχτηκε και το ποσοστό επιλογής.

Το όριο που επιλέχτηκε σαν κατώτερο ποσοστό επιλογής πρότυπης κίνησης είναι αυτό του 30%. Συνεπώς από το αρχείο των 7570 πρότυπων κινήσεων, στο οποίο προφανώς περιλαμβάνονται και οι 8 δυνατές απεικονίσεις κάθε κίνησης, επιλέχτηκαν 309 αποτελέσματα που ξεπερνούσαν το όριο.

Στη συνέχεια έπρεπε να βρεθεί ένας τρόπος ώστε να γίνεται ταύτιση αποφεύγοντας να ελέγχουμε οκτώ περιμετρικές θέσεις για κάθε μια από τις 81 του πλαισίου παιχνιδιού κάθε φορά. Η πρώτη σκέψη ήταν να χαρακτηριστεί η κάθε θέση του πλαισίου παιχνιδιού με το TAG της πρότυπης κίνησης που την περιβάλλει και έπειτα αυτό το TAG να ελέγξουμε αν ταυτίζεται με κάποιο από τα 309. Όμως, 309 κατά το

μέγιστο έλεγχοι για κάθε θέση είναι αρκετοί, για αυτό χρησιμοποιήθηκε διαφορετική τακτική και τεχνική ώστε να μειωθεί αυτός ο αριθμός, η οποία, όμως, συνέβαλε στην μείωση της συχνότητας λειτουργιάς της FPGA, εμφανίζοντας και κάποια σημεία δυσλειτουργίας.

Ο τρόπος που χρησιμοποιήθηκε ήταν να κωδικοποιηθούν όλες οι πιθανές καταστάσεις που μπορεί να περιγράφει μια πρότυπη κίνηση ως εξής:

- Μαύρη πέτρα → 2
- Άσπρη πέτρα → 3
- Τοίχος/Τέλος του πλαισίου παιχνιδιού → 5
- Άδεια θέση → 7

Αυτό, διότι οι παραπάνω αριθμοί είναι πρώτοι και έχουν την ιδιότητα όταν πολλαπλασιαστούν να δίνουν μοναδικό γινόμενο. Συνεπώς, μπορούμε να χαρακτηρίσουμε κάθε θέση του πλαισίου παιχνιδιού από το γινόμενο των στοιχείων, που την περιβάλλουν. Αν η κωδικοποίηση των στοιχείων είναι η παραπάνω θα μπορούμε να γνωρίζουμε από τι περιβάλλεται.

Δυστυχώς, ο τρόπος αυτός δεν μας εξασφαλίζει και την ακριβή γνώση της σειράς με την οποία περιβάλλουν τα στοιχεία κάθε θέση. Το ανεχόμαστε όμως αυτό, διότι με τον παραπάνω τρόπο οι έλεγχοι για ταύτιση από 309 έγιναν 21. Αυτό, γιατί και τα 8 στιγμιότυπα μιας πρότυπης κίνησης έχουν ίδιο αποτέλεσμα αλλά και κάποιες κινήσεις τυγχάνει να δημιουργούν το ίδιο γινόμενο. Επιπλέον, για να βοηθήσουμε την μετάβαση αυτής της τεχνικής στο Hardware και σε bits αντικαταστήσαμε τον αριθμό 7 με εκείνον του 1, χωρίς να επηρεάζεται η ιδιότητα του γινομένου, απλά γλιτώνοντας κάποια bits αποφεύγοντας ένα (τον αναγκαστικό, λόγω κενής μεσαίας θέσης) ή περισσότερους πολλαπλασιασμούς με το 7.

- Μαύρη πέτρα → 2
- Άσπρη πέτρα → 3
- Τοίχος/Τέλος του πλαισίου παιχνιδιού → 5
- Άδεια θέση → 1

Λόγω τη δυσλειτουργίας της μεθόδου αυτής, αναγκαστικά, υποβάλλονται και οι κινήσεις, που προκύπτουν από την εκάστοτε ταύτιση σε ένα τακτικό έλεγχο, ώστε να μην επιλεχτούν εκείνες -οποίες ίσως δεν είναι πρότυπες- που είναι τακτικά κακές. Αυτό κατά κάποιο τρόπο μετριάζει το πρόβλημα, χωρίς να σημαίνει ότι δεν θα υπάρχουν περιπτώσεις, όπου μια κίνηση -που δεν είναι πρότυπη, αλλά περνά τον τακτικό έλεγχο- δεν θα διαλεχτεί, έναντι κάποιας άλλης, σταματώντας την περεταίρω αναζήτηση εύρεσης πρότυπης κίνησης.

3.3.4 Λοιπές Διαφοροποιήσεις - Αλλαγές

Παραπάνω, έγινε αναφορά στην γεννήτρια αριθμών, που χρησιμοποιήθηκε και τον τρόπο με τον οποίο μας δίνει τις δύο (x,y) συνταγμένες αλλά και την περίπτωση να τροφοδοτήσει το σύστημα με θέσεις εκτός πλαισίου παιχνιδιού ή κατειλημμένες. Επειδή η γεννήτρια έχει μια συγκεκριμένη, αν και μεγάλη, περίοδο δεν είναι ότι καλύτερο σε κάθε περίπτωση λανθασμένης θέσης να την επαναενεργοποιούμε μειώνοντας τις εναπομείναντες καταστάσεις της. Για αυτό το λόγο, δημιουργήθηκε μετά την γεννήτρια μια μονάδα, αντιστοίχησης κάθε κίνησης της γεννήτριας σε 4 διαφορετικές.

- **Περίπτωση 1:** Παραμένουν οι συντεταγμένες ανεπηρέαστες (x,y).
- **Περίπτωση 2:** Αντιστρέφονται οι άξονες (y,x).
- **Περίπτωση 3:** Ελέγχεται ο αριθμός που δημιουργείται από την ένωση $z = xy$ των αποτελεσμάτων την γεννήτριας. Αν ο αριθμός είναι μεταξύ των ορίων $1 \leq z \leq 81$ τότε μετατρέπεται στις αντίστοιχες συντεταγμένες του πλαισίου παιχνιδιού. Ο αριθμός 1 αντιστοιχεί στην θέση (1,1), ο αριθμός 2 στην (1,2) και ο 81 στην (9,9). Οι αριθμοί εκτός των ορίων κωδικοποιούνται στην κίνηση (0,0).
- **Περίπτωση 4:** Σε αυτή την περίπτωση, το πλαίσιο παιχνιδιού χωρίζεται σε 9 ισομεγέθη τετράγωνα των 9 θέσεων το καθένα. Κάθε τετράγωνο, έχει συγκεκριμένη x συντεταγμένη και για τις 9 θέσεις του. Η συντεταγμένη y είναι τα νούμερα από το 1 έως το 9 που αντιπροσωπεύουν την κάθε θέση αντίστοιχα με τρόπο αρίθμησης όμοιο με αυτό της Περίπτωσης 3, που εφαρμόζεται σε όλο το πλαίσιο παιχνιδιού.

0100	**0110**	**0101**
1000	**1010**	**1001**
0000	**0010**	**0001**

  Ο τρόπος με τον οποίο επιλέχτηκαν τα τέσσερα bits που καθορίζουν την συντεταγμένη x κάθε τετραγώνου είναι τέτοιος, ώστε να ελέγχονται διαφορετικές περιοχές από την Περίπτωση 1.

Αφού, απορριφτεί κάθε περίπτωση τότε ελέγχεται η επομένη. Αν απορριφτούν και οι τέσσερις τότε επαναενεργοποιείται η γεννήτρια για να δώσει δύο καινούριους αριθμούς.

Μια ακόμη ουσιαστική αλλαγή, σε αυτή την σχεδίαση, είναι η προσπάθεια να εφαρμοστούν όσο το δυνατόν καλυτέρα οι κανόνες του παιχνιδιού. Δηλαδή, εν αντιθέσει με την σχεδίαση του Σ. Κόκκαλη, επιτρέπεται η τοποθέτηση πετράς σε περιοχή που έχει αιχμαλωτιστεί, ή σε κάποιο Eye. Στη δεύτερη περίπτωση επιτρέπεται εάν και μόνο αν αυτή συνεπάγεται αιχμαλωσία πέτρας/πετρών αντιπάλου.

Για το λόγο αυτό διαμορφώθηκε και ορίστηκε και διαφορετικά το τέλος του παιχνιδιού. Εννοείται, ότι μετά από δύο διαδοχικές κινήσεις πάσου, μια από κάθε αντίπαλο το παιχνίδι τελειώνει. Σε διαφορετική περίπτωση, όμως, το παιχνίδι τερματίζει έπειτα από συγκεκριμένο αριθμό κινήσεων. Σύμφωνα με έρευνες ένα παιχνίδι Go 9x9 διαρκεί κατά μέσο όρο 100 περίπου κινήσεις. Οπότε μόλις το παιχνίδι μετρήσει 100 κινήσεις ασχέτως ποιός έκανε την τελευταία και ποιός την πρώτη τερματίζει, ώστε να μην υπάρχει η περίπτωση να συνεχιστεί αέναα.

Επιπλέον, σχεδιάστηκε και χρησιμοποιείται ένα πλαίσιο παιχνιδιού κλώνος, ώστε να υλοποιούνται οι προσομοιώσεις Monte Carlo. Αυτό συμβαίνει, διότι αν θέλαμε να χρησιμοποιούμε το ίδιο πλαίσιο παιχνιδιού θα έπρεπε να αποθηκεύουμε σε κάθε θέση την κατάσταση της πριν την τυχαία προσομοίωση ολοκλήρωσης του παιχνιδιού. Θα έπρεπε, δηλαδή, να αποθηκεύεται το χρώμα της πέτρας που υπάρχει, την ομάδα στην οποία ανήκει, ο αριθμός ελευθεριών που έχει, αν γειτονεύει ή όχι με αιχμάλωτη περιοχή, αν είναι αιχμάλωτη περιοχή. Σχεδόν ότι πληροφορία κρατείται σε κάθε θέση, άρα δεν είχε νόημα να μην δημιουργηθεί ένα πλαίσιο παιχνιδιού κλώνος.

Ακόμη, υπάρχει ανώτατο όριο και στις διαθέσιμες αναζητήσεις για κίνηση που κάνει σε κάθε παίχτη. Είτε κατά την διάρκεια αναζήτησης κίνησης για τον παίχτη που προσομοιώνει το σύστημα που υλοποιήθηκε, είτε και για τους δύο κατά την διάρκεια της τυχαίας συνέχισης του παιχνιδιού ο αριθμός των αναζητήσεων για τον καθένα δεν ξεπερνά τις 50. Μετά την τελευταία επιτρεπτή αναζήτηση ο παίχτης πάει πάσο. Με τον τρόπο αυτό, από την μια εισάγεται η κίνηση πάσο σαν μια πιθανή κίνηση στο παιχνίδι κάτι που το κάνει πιο ρεαλιστικό, από την άλλη όμως υπάρχει μεγάλη πιθανότητα το παιχνίδι να τελειώσει με δύο συνεχόμενα πάσο. Όσο γεμίζει το πλαίσιο παιχνιδιού τόσο πιο δύσκολο είναι για την γεννήτρια να βρει διαθέσιμη κίνηση.

Επίσης, πρέπει να τονιστεί ότι η μονάδα που κάνει ταύτιση πρότυπων κινήσεων, λειτουργεί μετά την $22^{η}$ διαδοχική κίνηση από την αρχή του παιχνιδιού, ώστε να έχει τοποθετηθεί ένας λογικός αριθμός πετρών στο πλαίσιο παιχνιδιού.

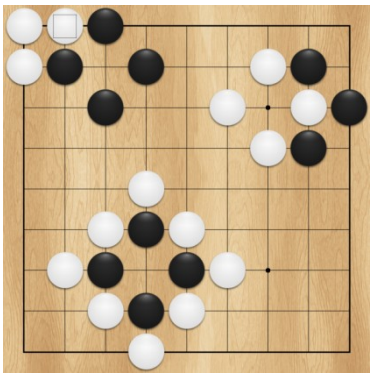
Ακόμη, επειδή κάθε, πετρά, ή ομάδα, μπορεί να αιχμαλωτιστεί από τόσες διαφορετικές αντίπαλες ομάδες, όσα τα συνολικά σημεία ελευθερίας καθιστώντας δύσκολο και αργό να τα εντοπίσουμε και να τα επισημάνουμε και αυτά αλλά και τις επιπλέον ελευθερίες που κερδίζει το καθένα μετά από μια αιχμαλωσία υιοθετήθηκε μια διαφορετική τακτική. Προστέθηκε μια μονάδα σε κάθε θέση του πλαισίου παιχνιδιού, η οποία απλά "ακούει" και "εκπέμπει" σε γείτονες -αν πρέπει- "ότι γειτονεύει με αιχμάλωτη περιοχή". Συνεπώς κάθε αιχμάλωτη θέση που της αφαιρείται η πέτρα εκπέμπει ένα μήνυμα στους τέσσερις γείτονες μέχρι να επανατοποθετηθεί πέτρα σε εκείνη τη θέση, ώστε να σταματήσει. Κάθε πέτρα, που ακούει το μήνυμα αυτό το επανεκπέμπει υπό προϋποθέσεις. Αν είναι αιχμάλωτη θέση το επανεκπέμπει προς όλους, αν είναι πέτρα διαφορετικής ομάδας ή άδεια θέση σταματά η εκπομπή πριν φτάσει στα σημεία αυτά. Με τον τρόπο αυτό, εξασφαλίζεται ότι Group, που έχουν συμμετάσχει σε αιχμαλώτιση αντίπαλων πετρών δεν θα αιχμαλωτιστούν έως

ότου καλυφτούν όλα τα σημεία ελευθερίας τους αλλά και τοποθετηθούν πέτρες στις θέσεις που εκπέμπουν κατάλληλο μήνυμα αιχμαλωσίας, προς αυτά.

Η τελευταία μονάδα, που διαφοροποιείται σημαντικά από εκείνη της προηγούμενης σχεδίασης είναι εκείνη του ελέγχου των αποδεκτών κινήσεων. Όπως έχει προαναφερθεί, η νέα μονάδα αποδέχεται κινήσεις σε θέσεις Eye - αιχμάλωτες, με την προϋπόθεση ότι είναι κινήσεις που αιχμαλωτίζουν αντίπαλες πέτρες. Επιπλέον, η μονάδα περιορίζει κάποιες τακτικά κακές κινήσεις, μόνο όσο αφορά το παιχνίδι του παίχτη, που προσομοιώνει, όχι και του αντιπάλου, στον οποίο απλά απαγορεύει κινήσεις σύμφωνα με τους κανόνες.

Κινήσεις που απαγορεύονται:

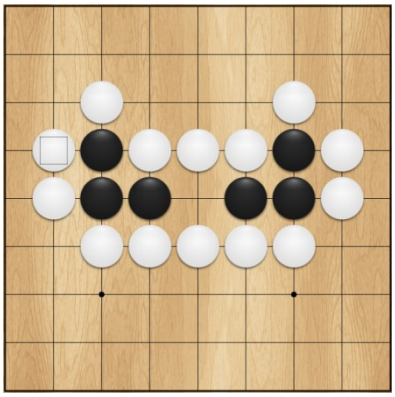
- Τοποθέτηση πέτρας εκτός ορίων του πλαισίου παιχνιδιού.
- Τοποθέτηση πέτρας σε ήδη κατειλημμένη θέση.
- Τοποθέτηση πέτρας σε θέση Eye, χωρίς να συνεπάγεται αιχμαλώτιση αντίπαλων πετρών.



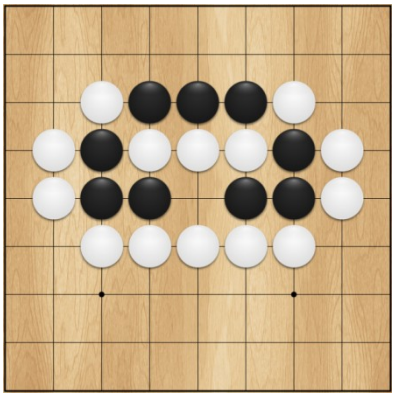
εικόνα 60 : Απαγορεύεται η τοποθέτηση άσπρης πέτρας στο πάνω αριστερά Eye. Στα άλλα δύο επιτρέπεται πέτρα χρώματος που προκαλεί αιχμαλώτιση

- Κίνηση αυτοκτονίας.
- Κινήσεις μείωσης ελευθεριών χωρίς, να συνεπάγεται αιχμαλωσία αντίπαλων πετρών, ή χωρίς να αφορά την ενδυνάμωση κάποιας ομάδας.

Ο τελευταίος κανόνας που είναι και ο διαφορετικός, περιγράφει όλες τις δυνατές περιπτώσεις, που μπορούν να υπάρξουν, όπου η σύνδεση δύο ομάδων πετρών, δεν θα έχει σαν αποτέλεσμα το άθροισμα των ελευθεριών, άρα και την αύξησή τους, αλλά την μείωση και ίσως μια κίνηση αυτοκτονίας. Προφανώς δεν απαγορεύεται πλήρως μια τέτοια πράξη, εφόσον αιχμαλωτίζει στον αντίπαλο.

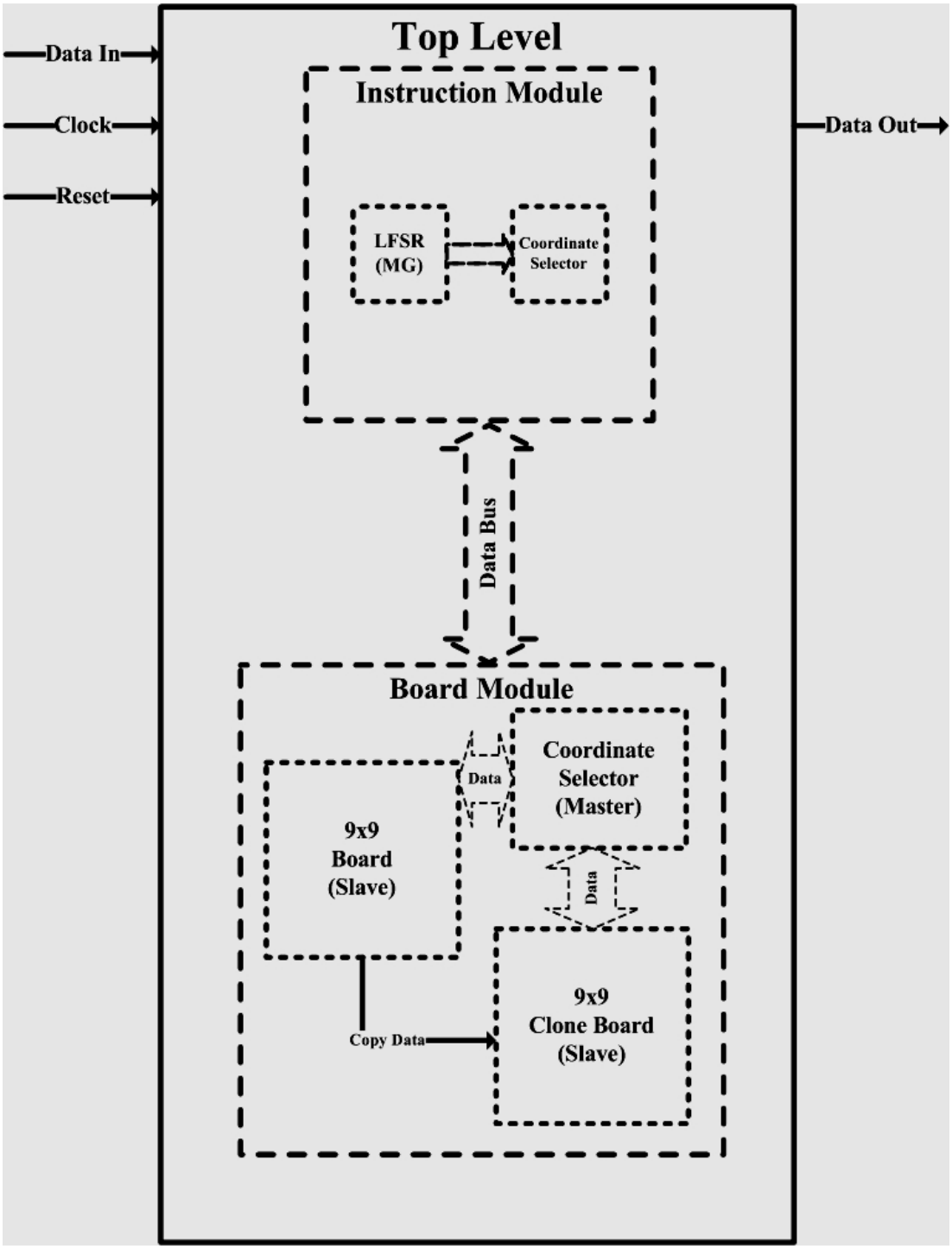


εικόνα 61 : Απαγορεύεται η τοποθέτηση μαύρης πέτρας, που ενώνει τις δύο ομάδες



εικόνα 62 : Επιτρέπεται η τοποθέτηση μαύρης πέτρας, που ενώνει τις δύο ομάδες

Συνεπώς το Ανώτατο Επίπεδο της νέας σχεδίασης θα είναι έτσι:



εικόνα 63: Ανώτατο Επίπεδο

Θα αναλυθεί καλύτερα και βαθύτερα στο επόμενο Κεφάλαιο 4.

Κεφάλαιο 4

Σχεδίαση και Υλοποίηση

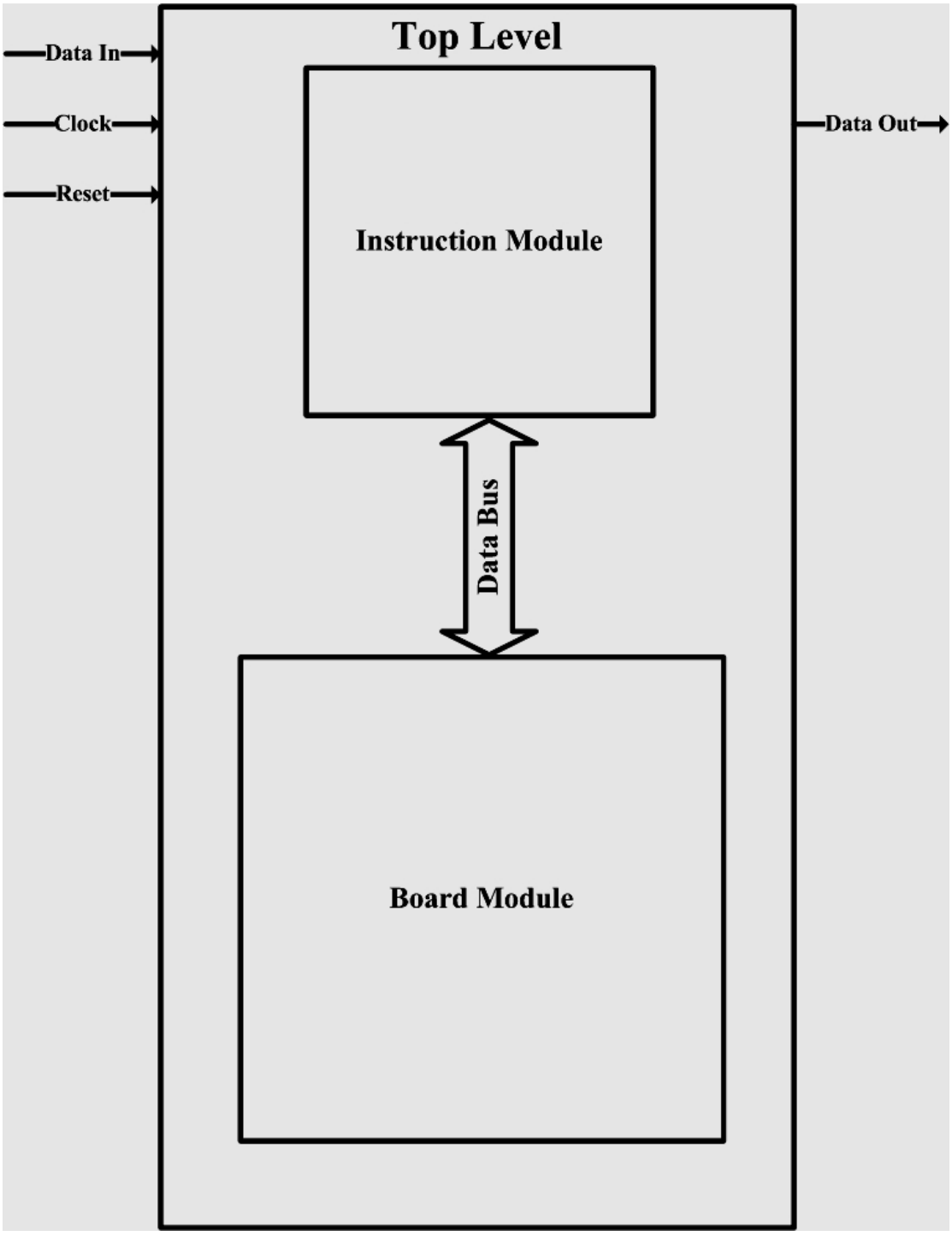
4.1 Εισαγωγή

Πριν την σχεδίαση και υλοποίηση των καινούριων και διαφορετικών μονάδων ή την τροποποίηση κάποιων υπαρχόντων υλοποιήθηκε η αρχιτεκτονική, του Σ. Κόκκαλη από το μηδέν και ελέγχτηκε η ορθή λειτουργία της με το κατάλληλο τεστ. Στην συνέχεια, τροποποιήθηκε με σκοπό την τακτική ενίσχυση η μονάδα ελέγχου κάθε νέας κίνησης πράγμα, που επέφερε αλλαγές και προσθέσεις μονάδων και σε κάθε θέση του πλαισίου παιχνιδιού. Η λειτουργία του συστήματος ελέγχτηκε με τη χειροκίνητη τοποθέτηση πετρών μέσω τεστ ώστε να ελεγχτούν όλες οι καινούριες περιπτώσεις που προέκυψαν. Στη συνέχεια προστέθηκε στο σύστημα η τυχαία γεννήτρια αριθμών αλλά και η μονάδα αντιστοίχησης κάθε σετ αριθμών στις τέσσερις διαφορετικές δυάδες συντεταγμένων. Ένα στάδιο πριν το τέλος προστέθηκαν οι μονάδες ταύτισης πρότυπων κινήσεων και μέτρησης σκορ αντίστοιχα. Τέλος, υλοποιήθηκε το πλαίσιο παιχνιδιού κλώνος και με κατάλληλο τεστ, παρατηρήθηκε μέσω του iSim η φυσιολογική και σωστή συμπεριφορά του συστήματος.

Ας παρατηρήσουμε όμως την σχηματική απεικόνιση του συστήματος μέσω σχηματικών διαγραμμάτων(Block Diagrams), σχολίων, συμπερασμάτων και αποτελεσμάτων.

4.2 Σχηματικά Διαγράμματα Μονάδων Σχεδίασης

4.2.1 Ανώτατο Επίπεδο



εικόνα 64 : Top Level

Η δομή των εισόδων και των εξόδων είναι η παρακάτω:

Data In(8 bits):

X Coordinate(4 bits)	**Y Coordinate**(4 bits)
(7 downto 4)	(3 downto 0)

ή

Color(1 bit)
(0)

Data Out(8 bits):

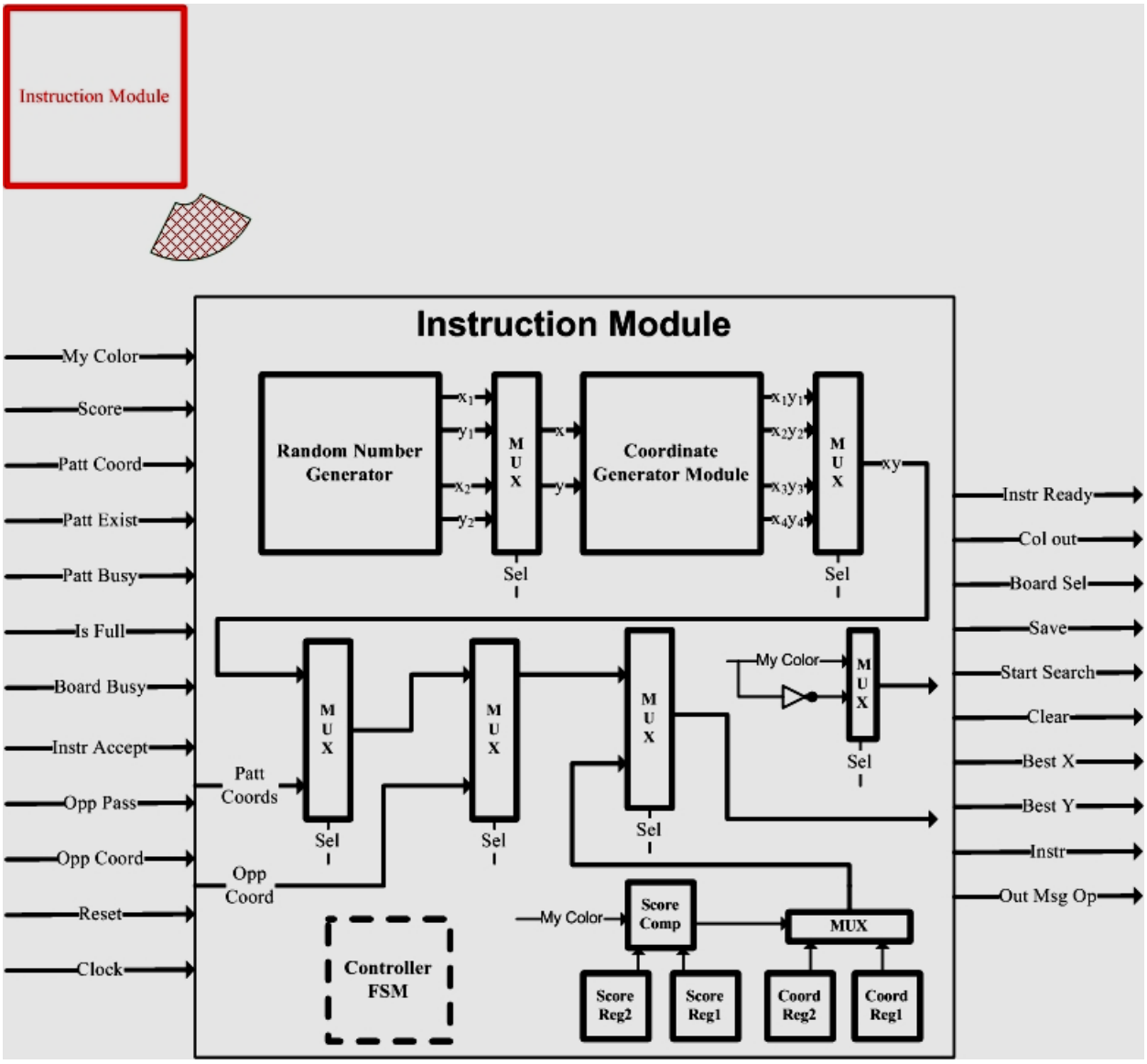
X Coordinate(4 bits)	**Y Coordinate**(4 bits)
(7 downto 4)	(3 downto 0)

ή

Score(8 bits)
(7 downto 0)

Όπως φαίνεται, αρχικά τροφοδοτούμε το σύστημα με το χρώμα μας και έπειτα με τις κινήσεις του αντιπάλου και εκείνο ανταποκρίνεται με την κίνηση, την οποία πρέπει να παίξουμε, ή με το σκορ στο τέλος του παιχνιδιού.

4.2.2 Μονάδα Εντολών



εικόνα 65 : Instruction Module

Όπως φαίνεται και από το σχεδιαστικό διάγραμμα η παραπάνω μονάδα είναι εκείνη που αναλαμβάνει την διαχείριση και την εξέλιξη του παιχνιδιού. Από τη γεννήτρια τυχαίων αριθμών διαλέγουμε συνολικά 16 από τα 168 bits του L.F.S.R. ώστε να δημιουργούνται δύο σετ συντεταγμένων. Αυτό, διότι κάθε φορά που τελειώνει η προσομοίωση για το πρώτο σετ συντεταγμένων, που αποθηκεύονται στον αντίστοιχο καταχωρητή, και κάνουμε Reset στον L.F.S.R. λόγω της ντετερμινιστικής φύσης του θα μας έδινε το ίδιο σετ προς σύγκριση με το προηγούμενο. Συνεπώς, σε εκείνο το σημείο η FSM, που αναλαμβάνει όλα τα σήματα, διαλέγει την δεύτερη 8-άδα. Η FSM είναι επίσης υπεύθυνη για την εκπομπή σήματος προς αντιγραφή ή εκκαθάριση του πλαισίου παιχνιδιού κλώνος, προς έναρξη αναζήτησης για ταύτιση πρότυπης κίνησης, όπως και για την έκδοση κατάλληλου κωδικού Out Message, για τον διαχωρισμό της εξόδου του σκορ και των ιδανικών συντεταγμένων του

συνολικού συστήματος. Επίσης, υπάρχει μια σειρά πολυπλεκτών, των οποίων τα σήματα επιλογής(Select) διαχειρίζεται η FSM καθορίζοντας τις συντεταγμένες και το χρώμα στην εντολή(Instruction) την οποία εκδίδει, επιλέγοντας μεταξύ, των εξωτερικών συντεταγμένων από το χρήστη, εκείνων που έρχονται από την μονάδα αναγνώρισης πρότυπων κινήσεων, αυτών από την μονάδα Coordinate Generator και των καλύτερων μετά από την σύγκριση των προσομοιώσεων Monte Carlo.

Η δομή της εντολής που αποστέλλεται στη μονάδα πλαισίου παιχνιδιού(Board) είναι έτσι:

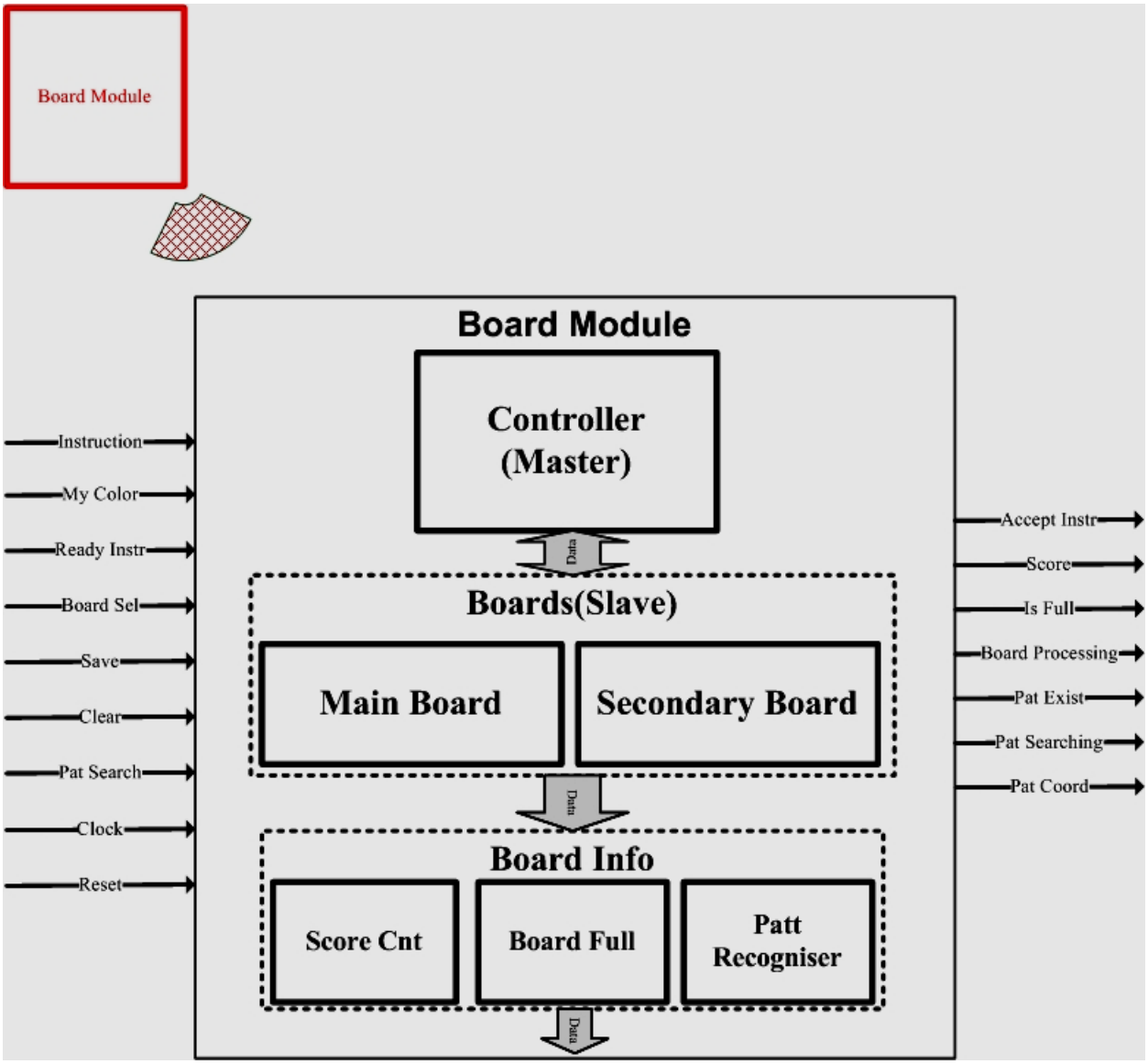
Instruction(12 bits):

Opcode(3 bits)	**Color**(1 bit)	**Coordinates**(8 bits)
(11 downto 9)	(8)	(7 downto 0)

Και μπορεί να είναι τριών ειδών:

Opcode	**Meaning**
"000"	Do Nothing
"001"	New Move
"100"	Count Points

4.2.3 Μονάδα Πλαισίου Παιχνιδιού



εικόνα 66 : Board Module

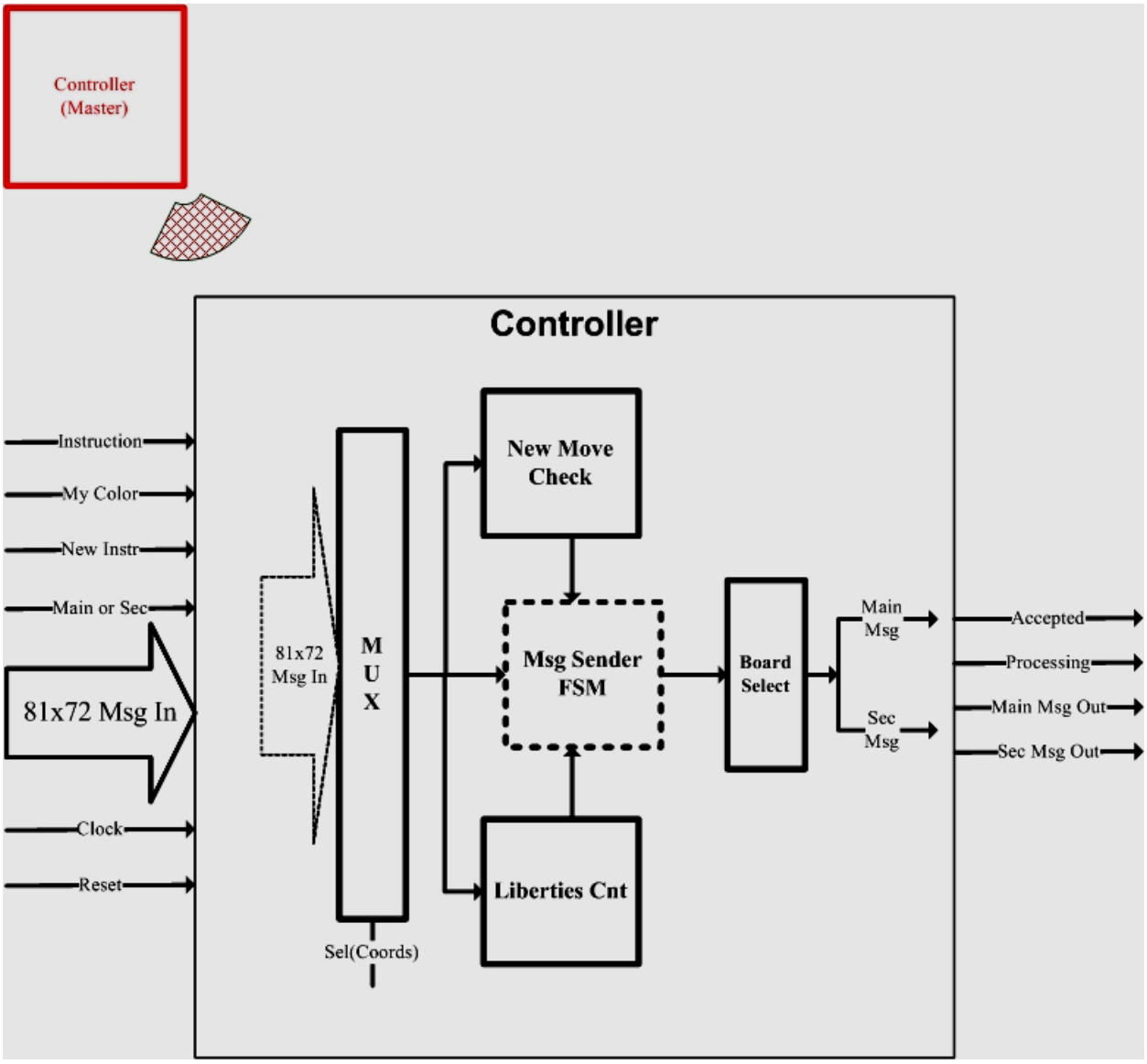
Παρατηρώντας το παραπάνω σχεδιαστικό διάγραμμα, η μονάδα Controller παίζει το ρόλο του επεξεργαστή των εντολών, που φτάνουν από την παραπάνω μονάδαInstruction, αλλά και εκείνο του εκπομπού των κατάλληλων μηνυμάτων προς όλες τις κατάλληλες κατευθύνσεις. Για να αποσταλούν, όμως τα κατάλληλα μηνύματα προς τις σωστές κατευθύνσεις απαιτείται η επεξεργασία των κατάλληλων πληροφοριών. Για το λόγο αυτό $81 \times 72 = 5832$ bits αποστέλλονται από τη μονάδα Boards και άλλα 23 bits από την Master μονάδα στη Slave. Ανάλογα με τις συντεταγμένες της νέας κίνησης που φτάνει ελέγχονται τα αντίστοιχα bits και διαμορφώνονται τα μηνύματα των 23 bits. Η μονάδα Boards επίσης τροφοδοτεί την μονάδα που παρέχει πληροφορίες για το σκορ με $81 \times 2 = 162$ bits, την μονάδα ελέγχου πληρότητας του πλαισίου παιχνιδιού με $81 \times 1 = 81$ bits -περιλαμβάνονται στον αρχικό δίαυλο, άρα και αντλούνται από εκεί- και την τελευταία που

αναλαμβάνει το την ταύτιση των πρότυπων θέσεων με $81 \times 33 = 2673$ bits -σε αντίθεση με πριν εδώ μόνο τα 81 είναι κοινά με τα παραπάνω 5832 bits-.

Συνεπώς παρατηρούμε ότι μέσω των διαύλων δεδομένων μεταφέρονται αρκετά bits, τα οποία συνεισφέρουν στη δημιουργία αρκετού Latency, το οποίο είναι αντιστρόφως ανάλογο μέγεθος της συχνότητας λειτουργιάς της FPGA.

Εν κατακλείδι, παρατηρούμε, πως η μονάδα Board είναι υπεύθυνή για την ενημέρωση την μονάδας Instruction για την ορθότητα μιας εντολής, για την πληρότητα του πλαισίου παιχνιδιού αλλά και την λειτουργιά του(δηλαδή την περίοδο τοποθέτησης ή αιχμαλωσίας πετρών), όπως και για τα αποτελέσματα και την λειτουργία της μονάδας για την αναγνώριση πρότυπων κινήσεων. Τέλος, τροφοδοτεί τον χρήστη και όποιον το ζητά με το σκορ στο πέρας του παιχνιδιού, ή κάποιας προσομοίωσης Monte Carlo.

4.2.4 Μονάδα Ελεγκτή - Ρυθμιστή



εικόνα 67 : Controller Module

Όπως προκύπτει και από το σχήμα το ρόλο του Επεξεργαστή - Ρυθμιστή έχει η κεντρική μονάδα Msg Sender, η οποία υλοποιήθηκε με μια FSM. Η FSM ανάλογα με τις πληροφορίες, που συλλέγει από τον έλεγχο εγκυρότητας της νέας θέσης προς τοποθέτηση, τον μετρητή των ελευθεριών, που θα έχει κατά την αρχική τοποθέτηση η νέα πέτρα, αλλά και τα δεδομένα των γειτονικών σημείων από το σετ συντεταγμένων στέλνει τα απαραίτητα μηνύματα. Τα μηνύματα, έπειτα, ανάλογα το σήμα ελέγχου που φτάνει από ανώτερη μονάδα κατευθύνονται, είτε προς το πρωτο - κύριο πλαίσιο παιχνιδιού, είτε προς το δευτερεύον - κλώνο πλαίσιο παιχνιδιού,. Βέβαια, απαραίτητη προϋπόθεση για όλα τα παραπάνω είναι η επεξεργασία των σωστών πληροφοριών. Ένας πολυπλέκτης 81 to 1 με επιλογέα(select) τις συντεταγμένες της εισόδου Instruction αναλαμβάνει να κατευθύνει το σωστό Msg In. Επιπλέον, πρέπει να τονίσουμε ότι κύριο χαρακτηριστικό της μονάδας New Move Check, εκτός του

τυπικού -βάσει των κανόνων- ελέγχου εγκυρότητας κάθε νέας κίνησης, είναι οι πολλές συγκρίσεις και οι πολλοί έλεγχοι, που κάνει, ώστε να απορρίψει τακτικά ασθενείς κινήσεις.

Η δομή των Msg In είναι η παρακάτω:

Msg In(72 bits):

del (1 bit)	**no remove** (1 bit)	**EmptyCol** (2 bits)	**West info** (17 bits)	**North info** (17 bits)	**East info** (17 bits)	**South info** (17 bits)
(71)	(70)	(69 downto 68)	(67 downto 51)	(50 downto 34)	(33 downto 17)	(16 downto 0)

Neighbor Info(17 bits):

Del (1 bit)	**no remove** (1 bit)	**EmptyCol** (2 bits)	**group_id** (6 bits)	**libs** (7 bits)
(16)	(15)	(14 downto 13)	(12 downto 7)	(6 downto 0)

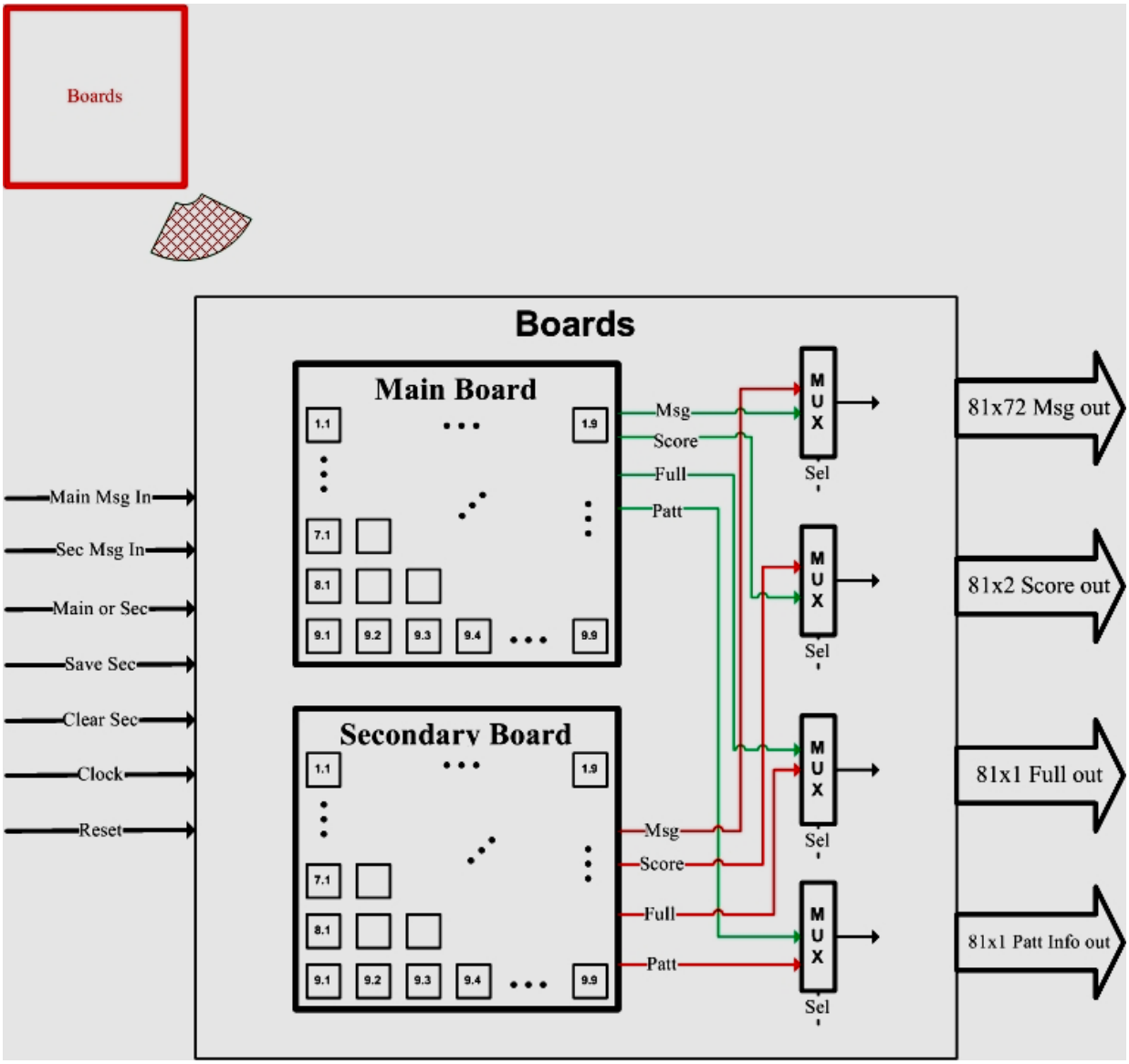
Ενώ η δομή του Msg out είναι η ακόλουθη:

Msg Out(24 bits):

	Opcode (2 bits)	**x** (4 bits)	**y** (4 bits)	**color** (1 bit)	**group** (6 bits)	**libs** (7 bits)
new move	**"00"** (23 downto 22)	(21 downto 18)	(17 downto 14)	(13)	(12 downto 7)	(6 downto 0)
merge	**"10"** (23 downto 22)	d.c. (21 downto 20)	group_1 (19 downto 14)	(13)	group_2 (12 downto 7)	(6 downto 0)
decrease	**"01"** (23 downto 22)	d.c.	d.c.	(13)	(12 downto 7)	d.c.
delete	**"11"** (23 downto 22)	d.c.	d.c.	(13)	(12 downto 7)	d.c.

Η εντολή "new move" τοποθετεί μια πέτρα στις (x,y) συντεταγμένες να ανήκει σε συγκεκριμένη ομάδα και με συγκεκριμένο αριθμό ελευθεριών. Εκείνη, που αφορά την συνένωση των ομάδων πετρών και εκτελείται δύο φορές για κάθε "merge" τροποποιεί αυτές που ανήκουν στο group_2 κάνοντάς τες να ανήκουν στο group_1 με συγκεκριμένο αριθμό ελευθεριών. Ο λόγος, που η ένωση δύο ομάδων διαρκεί 2 κύκλους είναι, διότι πρώτα πρέπει να "τοποθετήσουμε" την νέα πέτρα στη γειτονική ομάδα και στη συνέχεια να τροποποιήσουμε τις ελευθερίες της ανανεωμένης ομάδας. Οι δύο τελευταίες εντολές είναι και οι πιο απλές, μειώνοντας κατά 1 την ελευθερία της συγκεκριμένης ομάδας και αφαιρώντας, όλες τις πέτρες, που ανήκουν σε συγκεκριμένη ομάδα. Η διαδικασία αποστολής μηνυμάτων επαναλαμβάνεται και για τους 4 γείτονες αντίστοιχα.

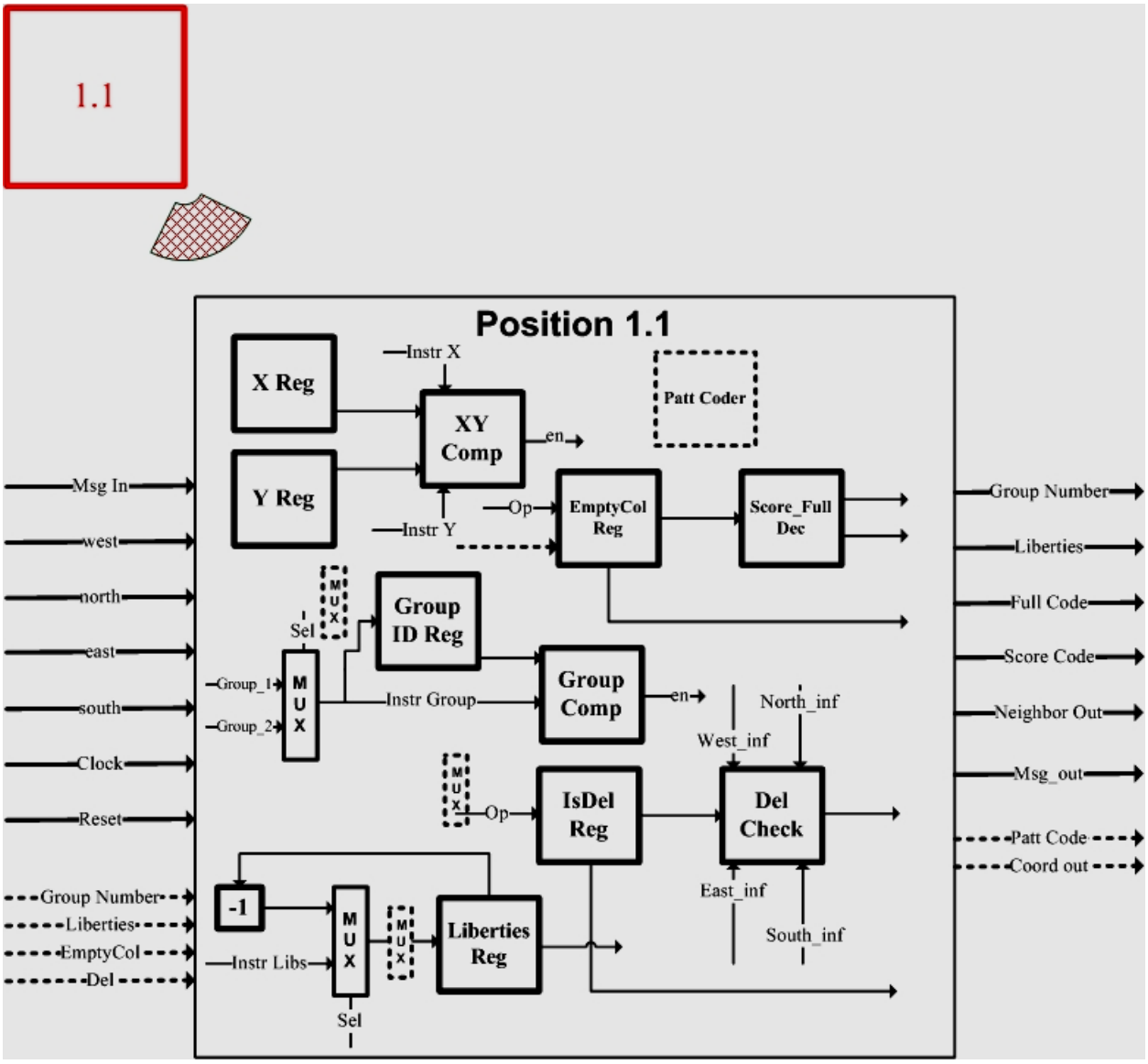
4.2.5 Μονάδα Πλαισίων Παιχνιδιού



εικόνα 68 : Boards Module

Το παραπάνω σχεδιαστικό διάγραμμα, ίσως, είναι το πιο απλό. Αποτελείται από τα Main Board και Secondary Board. Το Main, στο οποίο τοποθετούνται οι βασικές κινήσεις και η καλύτερη εκ των δύο προτεινόμενων, οι οποίες αξιολογούνται με την τυχαία ολοκλήρωση του παιχνιδιού από την κατάσταση στην οποία βρίσκεται εκείνη την στιγμή μέχρι τέλους. Η κατάσταση αντιγράφεται στο Secondary Board, όπου και πραγματοποιείται η προσομοίωση Monte Carlo. Αναλόγως, λοιπόν, ποιό από τα δύο Board χρησιμοποιείται, κατευθύνονται και τα αντίστοιχα δεδομένα ως έξοδοι από την μονάδα αυτή.

4.2.6 Μονάδα Θέσης



εικόνα 69 : Position Module

Ίσως το πολυπλοκότερο, το πιο σύνθετο αλλά και το πιο απαιτητικό στην υλοποίηση μέρος της σχεδίασης ήταν η θέση του πλαισίου παιχνιδιού. Κάθε μια από αυτές περιλαμβάνει δύο καταχωρητές, που έχουν αποθηκευμένες τις εκάστοτε συντεταγμένες. Αυτές συγκρίνονται με εκείνες κάθε νέας εντολής ώστε, εάν υπάρχει ισότητα να καταλάβει η συγκεκριμένη θέση ότι η νέα εντολή την αφορά και να ενεργοποιήσει για τον επόμενο κύκλο τα κατάλληλα σήματα που επιτρέπουν την εγγραφή δεδομένων στους καταχωρητές. Εφόσον, τοποθετηθεί πέτρα στη θέση τότε εκείνη αποκτά τιμή και στον καταχωρητή Group. Και η τιμή του Group αποτελεί έλεγχο για το στόχο κάθε νέας εντολής. Έτσι, λοιπόν, αν οι συγκριτές ομάδας ή συντεταγμένων επιστρέψουν τιμή 1, τότε τα σήματα εγγραφής καταχωρητών αποθήκευσης πληροφορίας ενεργοποιούνται για ένα κύκλο. Αυτοί οι καταχωρητές είναι:

- **Group ID Reg**
- **Liberties Reg**
- **isDel Reg**
- **EmptyCol Reg**

Ο καταχωρητής αποθήκευσης ελευθεριών, δέχεται είτε την τιμή της εντολής, είτε την υπάρχουσα -1, σε περίπτωση "decrease" εντολής.

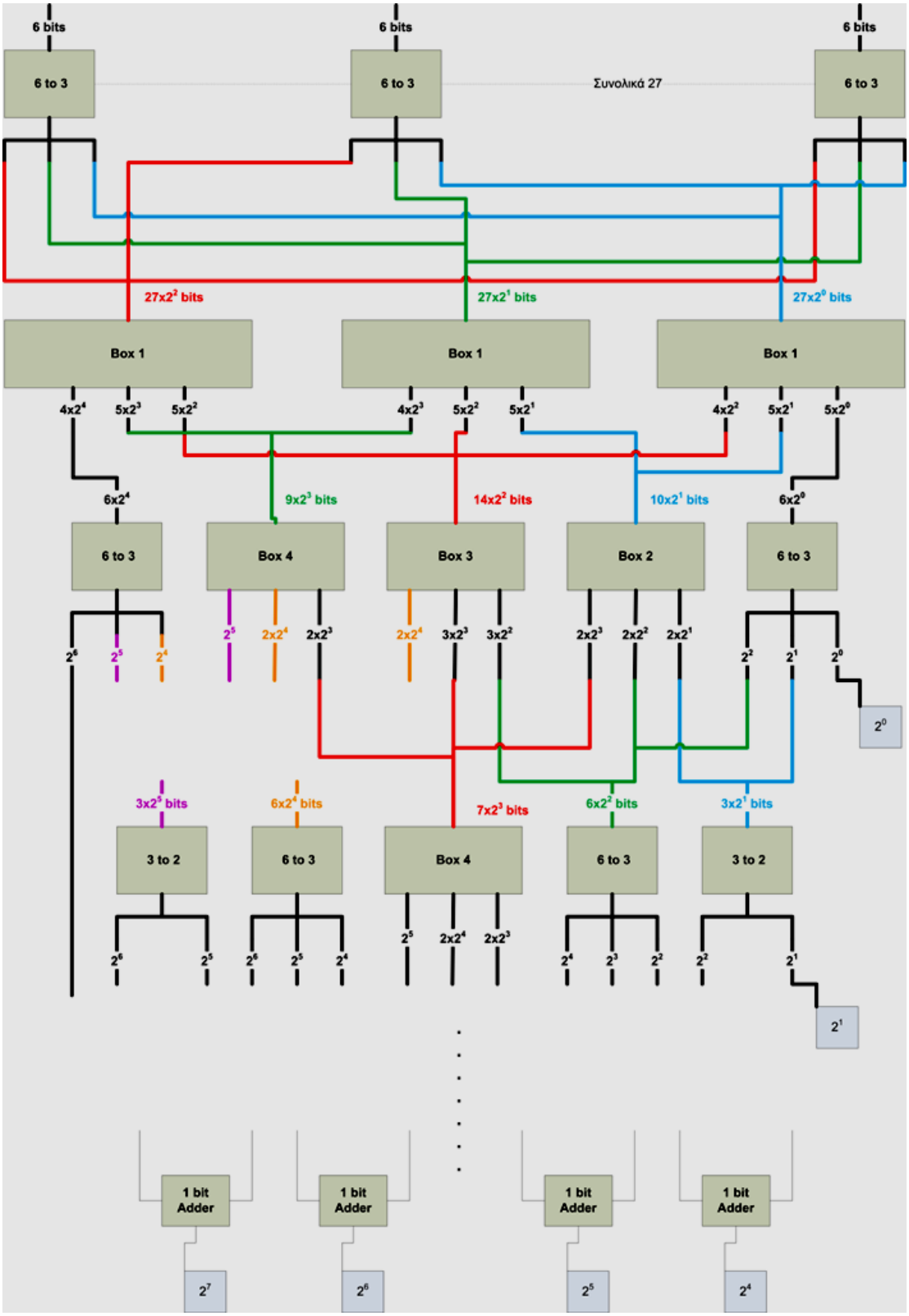
Εκείνος της αποθήκευσης πληροφορίας της ομάδας τροφοδοτείται με την τιμή από την εκάστοτε εντολή. Βέβαια, σε περίπτωση εντολής "merge", ελέγχεται στον ένα κύκλο η ομοιότητα με τον ένα αριθμό ομάδας, που περιέχει η εντολή και στον επόμενο με τον άλλο.

Ο καταχωρητής EmptyCol όταν ενεργοποιηθεί αποθηκεύει το χρώμα της νέας πέτρας, και αλλάζει την τιμή, που φανερώνει αν η θέση είναι ελεύθερη η κατειλημμένη. Έπειτα αποκωδικοποιείται για να σταλούν τα ανάλογα μηνύματα στις μονάδες Πληρότητας και Μέτρησης σκορ.

Τελευταία, ο καταχωρητής isDel αν η εντολή είναι εκείνη της διαγραφής πετρών και ενεργοποιηθεί τότε εξωτερικεύει την τιμή, που φανερώνει ότι η περιοχή είναι διεγραμμένη περιοχή. Ανάλογα με την τιμή αυτή αλλά και αντίστοιχες γειτονικές προκύπτει η τιμή, η οποία καθορίζει αν μπορεί να διαγραφεί ή όχι η θέση.

Με διακεκομμένες γραμμές, σχεδιάστηκαν οι είσοδοι, οι έξοδοι αλλά και τα στοιχεία που χρησιμοποιήθηκαν σε κάθε κλώνο θέση. Αρχικά, όλη η αποθηκευμένη πληροφορία μεταβιβάζεται από κάθε κύρια θέση στην δευτερεύουσα. Για αυτό χρειάζονται κάποιοι πολυπλέκτες, που χρησιμοποιούνται μόνο κατά την αντιγραφή -από το κύριο πλαίσιο παιχνιδιού, στο δευτερεύον- ουσιαστικά. Τέλος, η μονάδα κωδικοποίησης για την εύρεση πρότυπων κινήσεων δημιουργεί τα κατάλληλα σήματα προς την αντίστοιχη μονάδα. Για το λόγο αυτό, προέκυψαν και δύο ακόμη έξοδοι, μια για την παραπάνω κωδικοποιημένη τιμή και μια για τις συντεταγμένες της θέσης που ακλουθούν την κωδικοποιημένη τιμή, έτσι ώστε σε περίπτωση εύρεσης πρότυπης κίνησης να γνωρίζουμε σε ποιά θέση αναφέρεται.

4.2.7 Μονάδα Μέτρησης Σκορ



εικόνα 70 : Score Counter Module

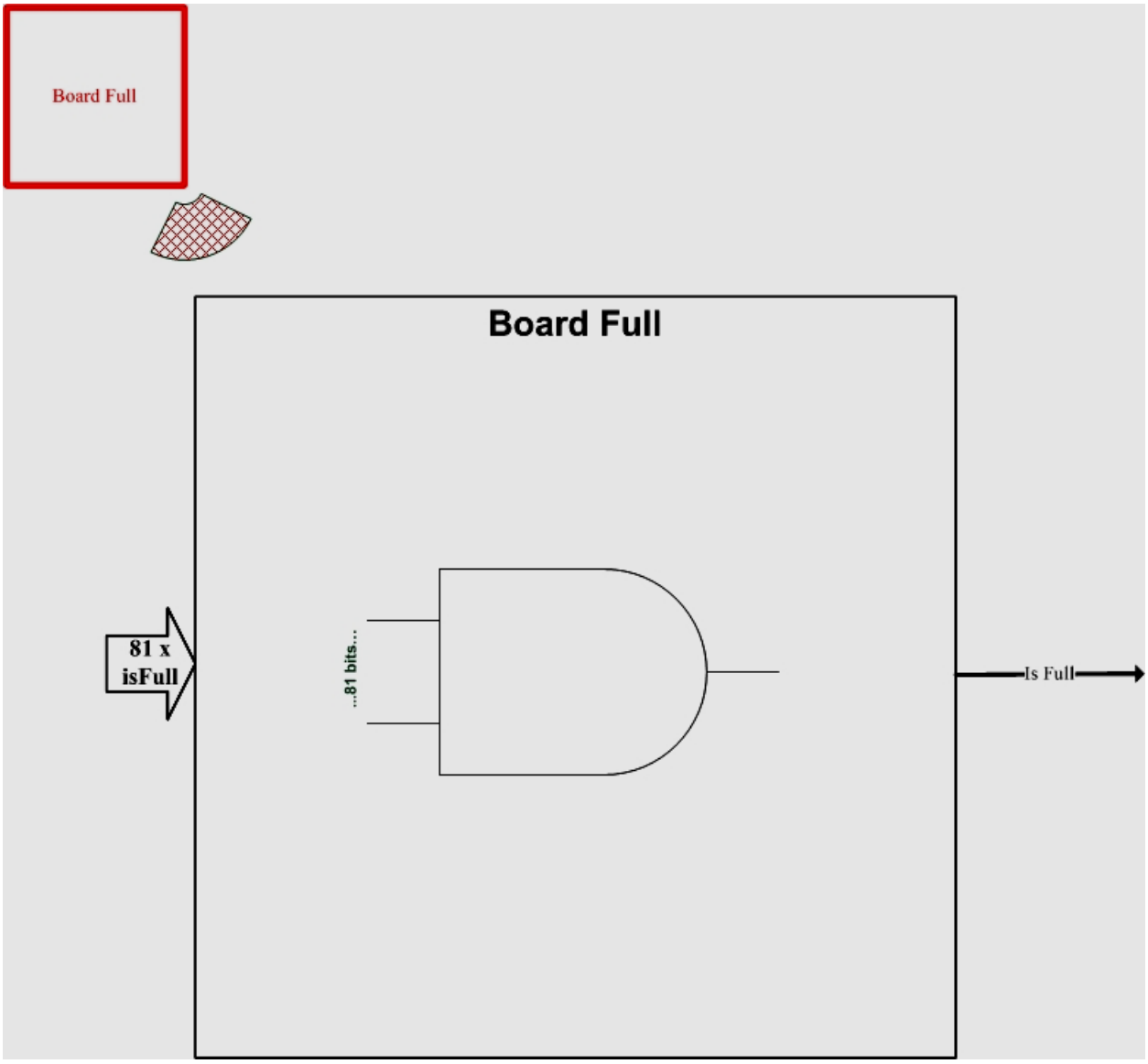
Η μονάδα που καταμετρά το σκορ αναλύθηκε στο προηγούμενο κεφάλαιο εκτενώς. Χρησιμοποιεί τις κωδικοποιημένες τιμές από όλες τις θέσεις

EmptyCol	**Κωδικοποιημένη Τιμή**
"10"(άδειο)	"10"
"01"(άσπρη πέτρα)	"11"
"00"(μαύρη πέτρα)	"00"

εικόνα 71 : Πίνακας Κωδικοποίησης Κατάστασης Θέσης

και εν συνεχεία μετράει τους άσσους σε μια μεγάλη ακολουθία 162 ψηφίων.

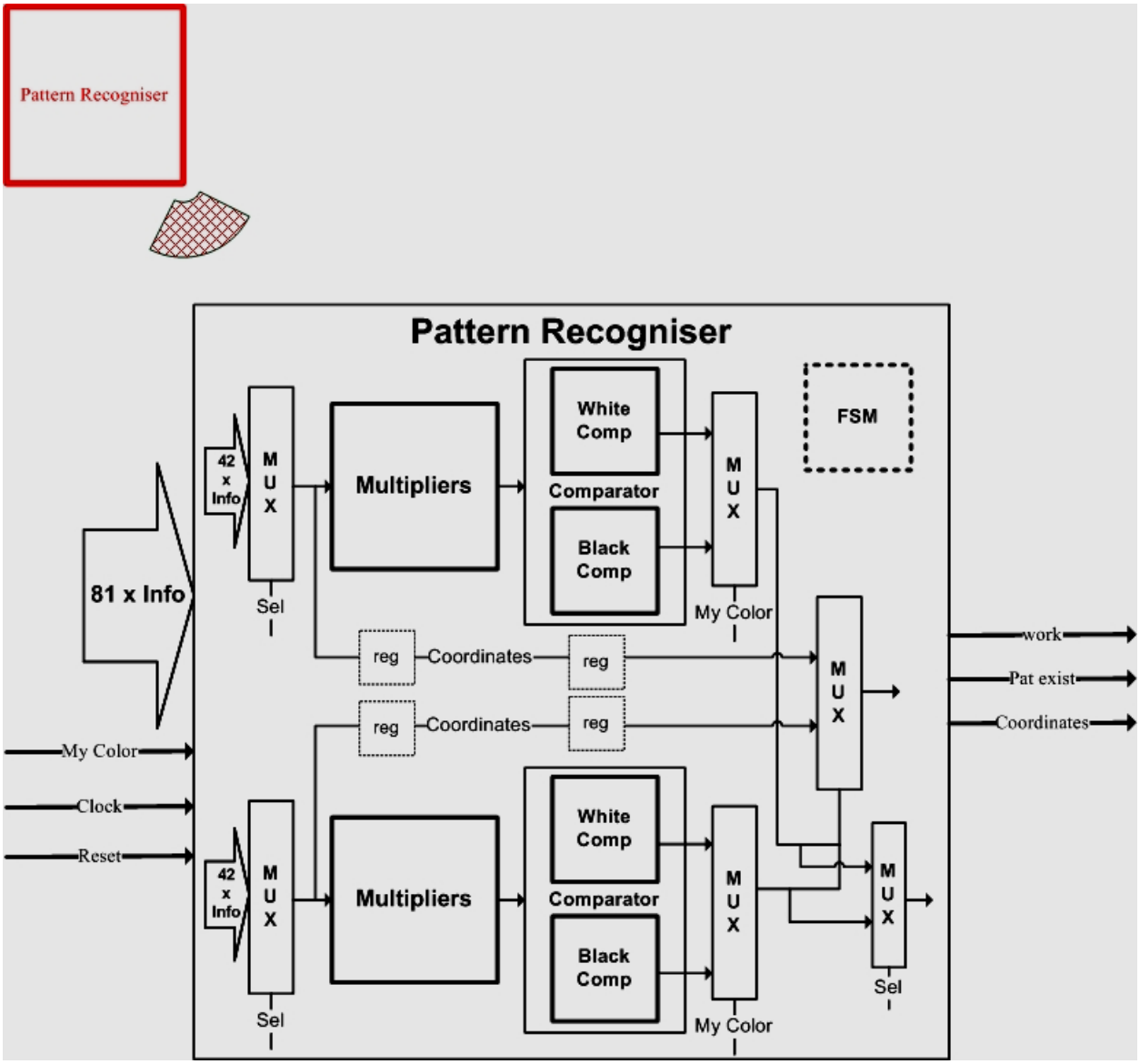
4.2.8 Μονάδα Ελέγχου Πληρότητας του Πλαισίου Παιχνιδιού



εικόνα 72 : Board Full Module

Το εργαλείο τις Xilinx, ανέλαβε την δημιουργία της παραπάνω μονάδας σχεδόν εξ ολοκλήρου. Με την λογική ότι το πλαίσιο παιχνιδιού, είναι εντελώς γεμάτο αν και μόνο αν όλες οι θέσεις του έχουν καταλειφθεί από πέτρα, τοποθετηθήκαν όλα τα σήματα, που το φανερώνουν, και από τις 81 θέσεις σε μια πύλη AND. Προφανώς, η πραγματική υλοποίηση αποτελείται από ένα ιδανικό δένδρο AND πυλών μικρότερου Fan - In που δημιουργείται κατά την βελτιστοποίηση της σχεδίασης από το σύστημα.

4.2.9 Μονάδα Αναγνώρισης Πρότυπων Κινήσεων



εικόνα 73 : Pattern Recogniser Module

Η τελευταία μονάδα είναι εκείνη που αναλαμβάνει την ταύτιση προτύπων κινήσεων. Ο τρόπος με τον οποίο το επιτυγχάνει αυτό απεικονίζεται στο παραπάνω σχήμα. Τα 81 σήματα των 33 bits χωρίζονται σε δύο ομάδες των 42. Ουσιαστικά ξεκάνει η αναζήτηση από την αρχή και την μέση του πλαισίου παιχνιδιού, καταλήγοντας στο κέντρο και το τέλος του αντίστοιχα. Την διαδικασία συντονίζει μια FSM, που κάθε κύκλο τροποποιεί τον επιλογέα των πολυπλεκτών ώστε να προχωρά η αναζήτηση και σταματά μόλις βρεθεί η πρώτη ταύτιση. Η διαδικασία είναι pipelined, αφού οι πολλαπλασιαστές έχουν δημιουργηθεί από το εργαλείο με σκοπό την βελτιστοποίηση σε σχέση με την ταχύτητα και είναι pipelined σε ιδανικό αριθμό σταδίων. Πρώτα πολλαπλασιάζονται οι οκτώ κωδικοποιημένες τιμές και με μορφή πυραμίδας συνεχίζει η πράξη. Λόγω της καθυστέρησης υπολογισμού του αποτελέσματος υλοποιήθηκαν οι κατάλληλοι καταχωρητές που μεταφέρουν

ταυτόχρονα με το αποτέλεσμα τις συντεταγμένες της πιθανής ταυτισμένης πρότυπης θέσης.

Κεφάλαιο 5

Επιβεβαίωση Λειτουργίας και Απόδοση

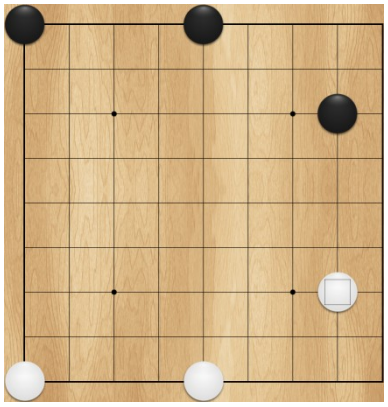
5.1 Εισαγωγή

Αρκετό χρόνο αλλά και μελέτη χρειάζεται και η επιβεβαίωση κάθε καινούριας σχεδίασης, εκτός από την υλοποίηση της. Όπως είναι φυσικό για να τεκμηριωθεί η πραγματικά ορθή λειτουργία και φυσιολογική συμπεριφορά κάθε μονάδας, που υλοποιήθηκε πρέπει να δημιουργηθούν και να πραγματοποιηθούν κατάλληλα τεστ. Οι δοκιμές, που πραγματοποιήθηκαν ήταν τέτοιες, ώστε να καλύπτουν όλες τις δυνατές περιπτώσεις, που πιθανώς θα εμφανιστούν κατά την διάρκεια πραγματικής λειτουργίας του συστήματος.

5.2 Ορθή Λειτουργία Πλαισίου Παιχνιδιού(Board) και Μονάδας Εντολών

Ξεκινώντας από την αρχή -χωρίς ύπαρξη προηγούμενων κωδίκων- την δημιουργία του συστήματος προσομοίωσης ενός παίχτη Go το πρωτο πράγμα, που υλοποιήθηκε και ελέγχτηκε ήταν η ορθή λειτουργία και συμπεριφορά του πλαισίου παιχνιδιού αλλά και της μονάδας, που δημιουργεί τις κατάλληλες εντολές ώστε να τοποθετηθούν καινούριες πέτρες στο πλαίσιο παιχνιδιού με σωστό αριθμό ελευθεριών, ή να επηρεαστούν κατάλληλα ήδη υπάρχουσες. Για να δημιουργηθούν κατάλληλες εντολές πρέπει η αντίστοιχη μονάδα να λαμβάνει και να επεξεργάζεται σωστά τις πληροφορίες από την νέα θέση και τους γείτονές της.

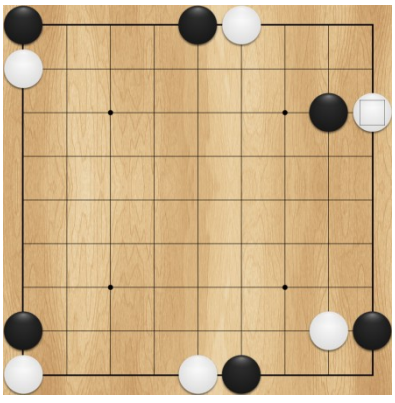
Το πρωτο τεστ περιελάμβανε την απλή τοποθέτηση και των δυο χρωμάτων πετρών σε σημεία που ποικίλουν διαθέσιμων ελευθεριών ώστε να ελεγχτεί η δημιουργία σωστών εντολών και απόδοσης σωστού αριθμού ελευθεριών αλλά και αριθμού ομάδας σε κάθε περίπτωση.



εικόνα 74 : Τεστ 1

Τοποθετήθηκαν, λοιπόν, πέτρες και των δυο χρωμάτων στις γωνίες του πλαισίου παιχνιδιού, σε σημείο όπου εφάπτονται από την μια μόνο πλευρά με όριο - τοίχο του πλαισίου παιχνιδιού και σε σημείο του πλαισίου παιχνιδιού όπου δεν υπάρχει επαφή με τίποτα και στις τέσσερις κατευθύνσεις. Όντως, οι καταχώρησες των ελευθεριών των γωνιακών πετρών είχαν αποθηκεύσει την τιμή δυο(2) σωστά. Αντίστοιχα, αυτές που μοιράζονται σημείο επαφής με τα όρια του πλαισίου παιχνιδιού είχαν τρεις(3) ελευθερίες, ενώ οι τελευταίες τέσσερις(4). Επίσης, κάθε πέτρα είχε διαφορετική έξοδο στον καταχωρητή αποθήκευσης αριθμού ομάδας με τιμές από το ένα(1), έως το έξι(6), όπως είναι φυσικό. Συνεπώς, η εντολή τοποθέτησης νέας πέτρας λειτουργεί σωστά.

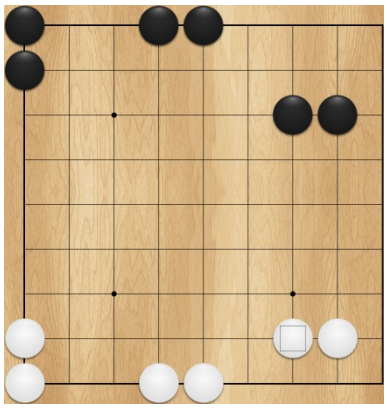
Η επόμενη δοκιμή αφορά την αλληλεπίδραση των διαφορετικών πετρών. Θα πρέπει κάθε διαφορετική πέτρα, η οποία τοποθετείται και έχει σημείο επαφής με αντίπαλη γειτονική, αφενός να λαμβάνει υπό όψιν την ύπαρξη της αντιπάλου και αφετέρου να μειώνει κατά μια τις ελευθερίες σε κάθε εφαπτόμενη αντίπαλη πέτρα.



εικόνα 75 : Τεστ 2

Για το λόγο αυτό, χρησιμοποιήθηκε το παραπάνω τεστ και στο ήδη δημιουργημένο μοτίβο, προστέθηκαν πέτρες αντίπαλου χρώματος εφαπτόμενες σε κάθε μια από τις ήδη υπάρχουσες. Πράγματι, όλες οι νέες πέτρες μέτρησαν δυο(2) ελευθερίες και διαφορετικό αριθμό ομάδας, ενώ μετά την εντολή τοποθέτησης, προέκυπτε εντολή μείωσης ελευθεριών για τις αντίπαλες πέτρες εφαπτόμενες πέτρες. Η εντολή τοποθέτησης, όπως και πριν, αφορούσε συγκεκριμένες συντεταγμένες, ενώ η εντολή μείωσης ελευθεριών συγκεκριμένο αριθμό ομάδας, όπου άνηκε η αντίπαλη πέτρα. Άρα, με αυτό τον τρόπο, επιβεβαιώθηκε και η ορθή λειτουργία της εντολής μείωσης ελευθεριών.

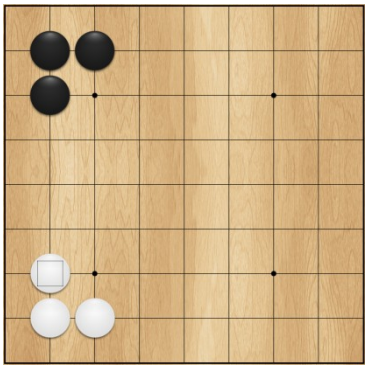
Επόμενη βασική λειτουργία που ελέγχτηκε ήταν εκείνη της ένωσης πετρών ίδιου χρώματος για την δημιουργία ομάδων πετρών με κοινό αριθμό ομάδας αλλά και νέο αριθμό ελευθεριών. Για αυτό το σκοπό, τοποθετήθηκαν στο αρχικό μοτίβο πέτρες ιδίου χρώματος παραπλεύρως από εκείνες που προϋπήρχαν.



εικόνα 76 : Τεστ 3

Το αποτέλεσμα ήταν ακριβώς όπως αναμενόταν. Οι νέες πέτρες τοποθετήθηκαν σαν καινούριες πέτρες με δικό τους αριθμό ομάδας και ελευθερίες. Αμέσως μετά, άλλαζε ο αριθμός ομάδας τους ώστε να ανήκουν στον ίδιο με την γειτονική πέτρα αλλά και για να ελευθερώσουν τον αριθμό αυτό ώστε να επαναχρησιμοποιηθεί. Έπειτα τροποποιείται ο αριθμός ελευθεριών της ομάδας στο άθροισμα ελευθεριών της γειτονικής πέτρας συν αυτών της καινούριας μείον 1. Έτσι δημιουργήθηκαν, σωστά, ομάδες με τρεις(3), τέσσερις(4) και έξι(6) ελευθερίες.

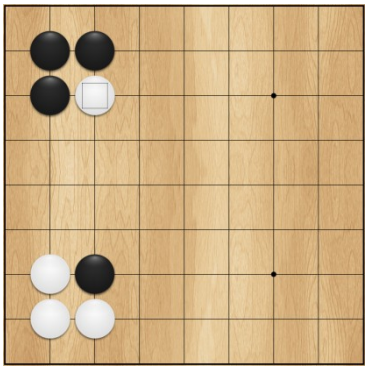
Ένα εύλογο ερώτημα - πρόβλημα που προκύπτει, είναι με τα σημεία τα οποία αποτελούν κοινά σημεία ελευθερίας μιας ομάδας. Αν υπάρχουν να μετριούνται δυο φορές ή κατά κάποιο τρόπο να γνωρίζουμε ποια είναι αυτά ώστε κατά την τοποθέτηση νέας πέτρας, πριν την ένωση να λαμβάνονται υπό όψιν; Επειδή, ο έλεγχος και η διαπίστωση αν ένα σημείο είναι κοινό σημείο ελευθερίας και οι επιπλέον ενέργειες θα υποχρέωναν την μονάδα εντολών σε αρκετούς ελέγχους αλλά και συγκρίσεις επιλέχτηκε να αγνοηθεί ασφαλώς αυτή η εξαίρεση.



εικόνα 77 : Τεστ 4

Σύμφωνα με το τεστ, που δημιουργεί τις παραπάνω ομάδες κάθε μια από αυτές μετράει οκτώ(8), αντί για επτά(7) ελευθερίες. Αυτό, όμως, δεν αποτελεί πρόβλημα, εφόσον η τοποθέτηση αντίπαλου χρώματος πέτρας στο κοινό σημείο διορθώνει την ανωμαλία, κάτι που είναι το ζητούμενο. Θα υπήρχε πρόβλημα με τον πλεονασμό ελευθεριών σε περίπτωση αιχμαλώτισης της ομάδας, διότι το κριτήριο είναι ο αριθμός ελευθεριών. Κάθε νέα τοποθέτηση πέτρας αποτελείται από πέντε βήματα, ένα για την αυτόνομη τοποθέτησή της στο σημείο που καταδεικνύουν οι συντεταγμένες(με τις κατάλληλες πληροφορίες ελευθεριών σύμφωνα με τα γειτονικά δεδομένα) και τέσσερα για την αλληλεπίδραση με τους γείτονες ανάλογα το χρώμα τους και την ύπαρξη τους.

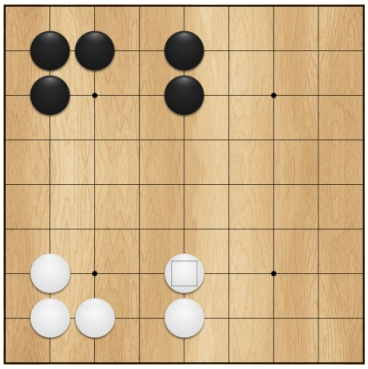
Συνεπώς, στη συνέχεια του τεστ με την τοποθέτηση αντίπαλου χρώματος πέτρας στο σημείο κοινών ελευθεριών



εικόνα 78 : Τεστ 5

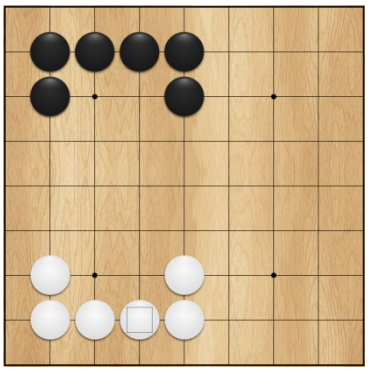
παρατηρούμε ότι εκτελείται δυο φορές η εντολή μείωσης ελευθεριών κατά μια, οπότε πλέον κάθε ομάδα έχει έξι(6) ελευθερίες ορθώς. Οπότε, αυτή η μικρή "δυσλειτουργία" δεν επηρεάζει καθόλου το σύστημα.

Πριν εξετάσουμε την εντολή αιχμαλώτισης πέτρας/πετρών υλοποιήθηκε και εκτελέστηκε ένα ακόμη τεστ για την δημιουργία νέας ομάδας πετρών, όχι όμως από μεμονωμένες πέτρες αλλά από την ένωση δυο ομάδων πετρών. Αρχικά τοποθετήθηκαν οι πέτρες ώστε να δημιουργηθούν οι κατάλληλες ομάδες



εικόνα 79 : Τεστ 6

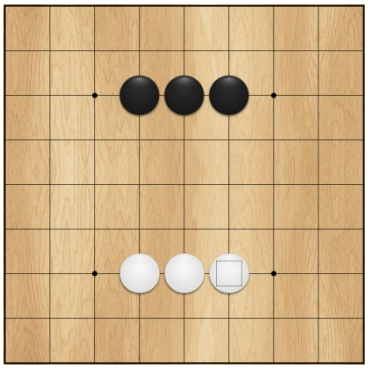
Στη συνέχεια τοποθετήθηκε η συνδετική πέτρα για να δημιουργηθεί μια μεγαλύτερη ομάδα για κάθε χρώμα.



εικόνα 80 : Τεστ 7

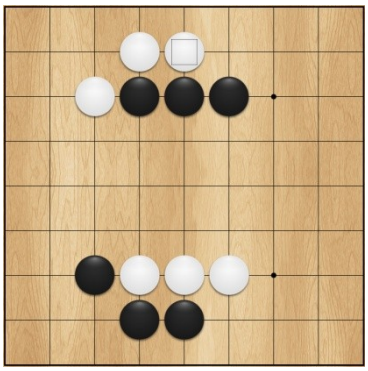
Η συνδετική πέτρα πρώτα ενώθηκε με την αριστερή ομάδα των τριών πετρών και δημιουργήθηκε μια νέα ομάδα τεσσάρων πετρών, η οποία έπειτα συνδέθηκε με την ομάδα των δυο κάθετων πετρών, των οποίων απόκτησε εκτός από τις ελευθερίες και τον αριθμό ομάδας. Πάντα σε ενώσεις πετρών η ομάδα που δημιουργείται ανήκει στην ομάδα στην οποία ενσωματώθηκε τελευταία. Λογικό, αφού η σειρά ελέγχου αλληλεπίδρασης είναι: Δύση → Βορράς → Ανατολή → Νότος. Ο αριθμός ελευθεριών για κάθε ομάδα είναι 14, λανθασμένος -προσωρινά- σε σχέση με την πραγματικότητα, χωρίς όμως αυτό να αποτελεί πρόβλημα.

Το τελευταίο τεστ, που πραγματοποιήθηκε, για την επιβεβαίωση ορθής υλοποίησης της προηγούμενης σχεδίασης του Σ. Κόκκαλη ήταν αυτό, που προσομοιώνει κατάσταση αιχμαλώτισης ομάδας πετρών όταν ο αριθμός των ελευθεριών μειώνεται στο μηδέν(0). Έτσι, δημιουργήθηκαν δυο όμοιες ομάδες, μια μαύρη και μια άσπρη με οκτώ(8) ελευθερίες.

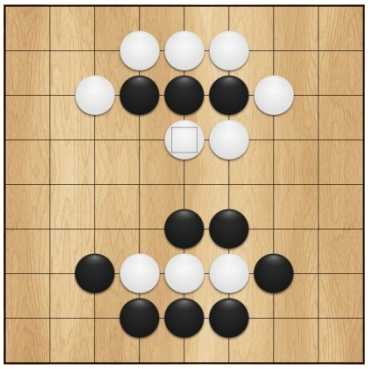


εικόνα 81 : Τεστ 8

Στη συνέχεια τοποθετούνταν πέτρες αντίπαλου χρώματος περιμετρικά ώστε να μειωθούν οι ελευθερίες στο ελάχιστο με την τοποθέτηση της 8^{ης} πέτρας.

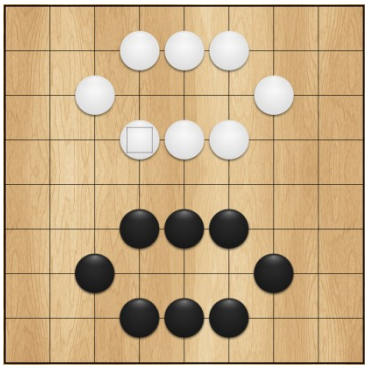


εικόνα 82 : Τεστ 9



εικόνα 83 : Τεστ 10

Ένα, βήμα πριν την τοποθέτηση της τελευταίας πέτρας αντίπαλου χρώματος για κάθε ομάδα οι καταχωρητές αποθήκευσης του αριθμού ελευθεριών έχουν σωστά έξοδο την τιμή ένα(1).

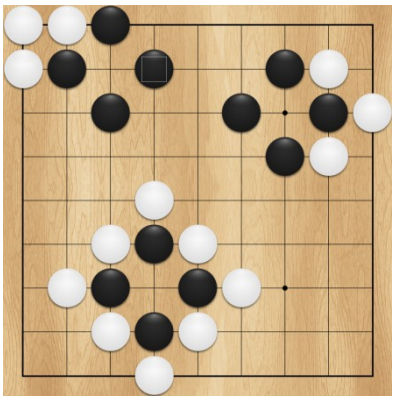


εικόνα 84 : Τεστ 11

Εφόσον, τοποθετηθούν και οι τελευταίες αναγκαίες πέτρες για την αιχμαλώτιση των ομάδων πετρών, τα σημεία στα οποία υπήρχαν οι αιχμάλωτες πέτρες, δείχνουν άδεια, όπως αναμενόταν, με την τιμή του καταχωρητή ελέγχου αν είναι άδεια στο '1', επιβεβαιώνοντας την σωστή λειτουργία αιχμαλώτισης πετρών.

5.3 Επιβεβαίωση Περιορισμών - Στρατηγικής Τοποθέτησης Πετρών

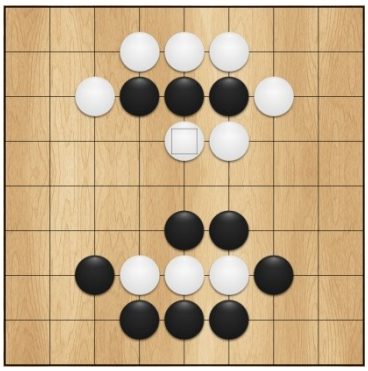
Όπως αναφέρθηκε και σε προηγούμενα κεφάλαια, η νέα σχεδίαση διαφοροποιείται λιγάκι στις αποδεκτές κινήσεις. Οι περιορισμοί είναι ίδιοι εκτός από έναν. Εκείνο, που επιτρέπει τοποθέτηση πέτρας σε Eye, εφόσον έχει νόημα. Δηλαδή, όπως έχει αναλυθεί στο 3^{o} κεφάλαιο, μόνο εφόσον είναι η τελευταία αναγκαία πέτρα για την αιχμαλώτιση αντίπαλης ομάδας πετρών. Για την επιβεβαίωση τοποθετήθηκαν μέσω κατάλληλου τεστ οι πέτρες όπως φαίνονται στο παρακάτω σχήμα.



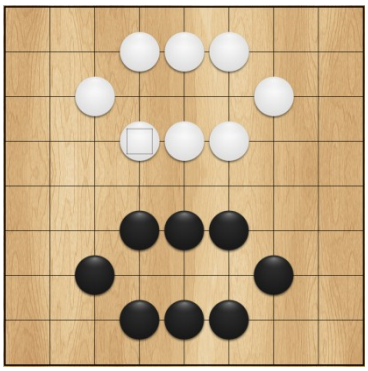
εικόνα 85 : Τεστ 12

Στην συνέχεια, δόθηκαν οι συντεταγμένες των τριών Eye για τοποθέτηση άσπρων πετρών. Λειτουργώντας το σύστημα σωστά, επέτρεψε την τοποθέτηση μόνο στα δυο δεξιότερα Eyes.

Η επέκταση του κανόνα - στρατηγικής οδήγησε στην αντιμετώπιση της περίπτωσης αιχμαλώτισης ομάδας πετρών και τοποθέτησης πέτρας/πετρών στο σημείο όπου υπήρχαν οι αιχμάλωτες πέτρες, οι οποίες αφαιρέθηκαν. Η μέθοδος αντιμετώπισης της κατάστασης αυτής σχολιάστηκε στο 3^{ο} κεφάλαιο. Οι θέσεις που περιείχαν αιχμάλωτες πέτρες εκπέμπουν ένα ειδικό σήμα προς τους εφαπτόμενους γείτονες μέχρι να καταλειφθούν από καινούριες πέτρες, όπου σταματά η εκπομπή. Για το λόγο αυτό χρησιμοποιήθηκε και επεκτάθηκε το απλό τεστ αιχμαλωσίας, ώστε να επιβεβαιωθεί η σωστή λειτουργία της μεθόδου.

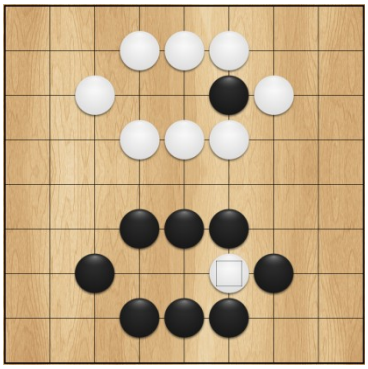


εικόνα 86 : Τεστ 13



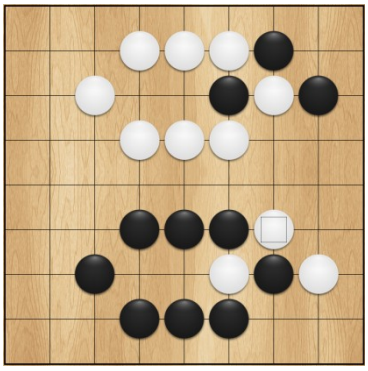
εικόνα 87 : Τεστ 14

Μετά την αιχμαλώτιση, των πετρών, δεν επηρεάστηκε ο αριθμός ελευθεριών των περιμετρικών ομάδων, αλλά το κατάλληλο σήμα (το οποίο υποδηλώνει ότι μια πέτρα γειτονεύει με περιοχή όπου αιχμαλωτίστηκε ή ανήκει σε ομάδα, η οποία είχε συνεισφορά στην αιχμαλωσία και δεν μπορεί να αιχμαλωτιστεί) ενεργοποιήθηκε.

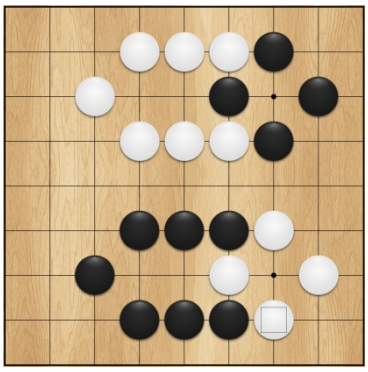


εικόνα 88 : Τεστ 15

Τοποθετώντας πέτρα αντιθέτου χρώματος, από εκείνες που αιχμαλωτίστηκαν, στη δεξιότερη "αιχμάλωτη" θέση επιβεβαιώνουμε μέσω του τεστ την σωστή απενεργοποίηση του κατάλληλου σήματος στην δεξιότερη μεμονωμένη πέτρα καθιστώντας την δυνατή προς αιχμαλωσία.



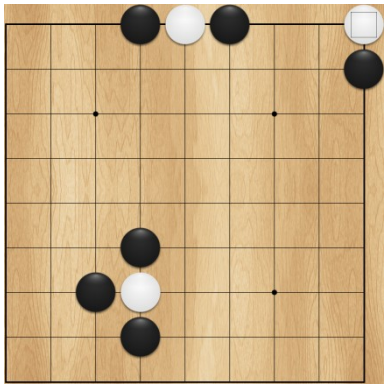
εικόνα 89 : Τεστ 16



εικόνα 90 : Τεστ 17

Για επιπλέον επιβεβαίωση τοποθετήθηκαν και οι υπόλοιπες πέτρες μέχρι την αιχμαλωσία.

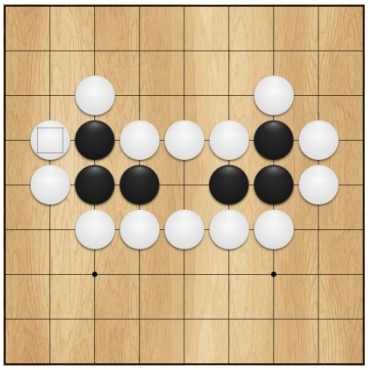
Επιπλέον, δημιουργήθηκε ένα απλό τεστ που επιβεβαίωσε την αποτροπή κάθε δυνατής κίνησης έμμεσης αυτοκτονίας



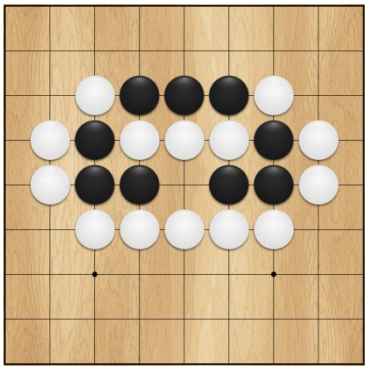
εικόνα 91 : Τεστ 18

καθώς δεν επέτρεψε την τοποθέτηση των άσπρων πετρών που φαίνονται στη παραπάνω εικόνα.

Ακόμη υλοποιήθηκαν ακόμη δυο αλληλουχίες κινήσεων για τον έλεγχο σωστής συμπεριφοράς της τακτικής τοποθέτησης πέτρας σε σημεία με μηδενικό αριθμό ελευθερίας. Σύμφωνα με την οποία αποτρέπεται η τοποθέτηση πετρών, οι οποίες έχουν αρνητικό αντίκτυπο στον αριθμό ελευθεριών γειτόνων ιδίου χρώματος και επιτρέπεται μόνο εφόσον συνεπάγεται αιχμαλώτιση αντίπαλων πετρών.



εικόνα 92 : Τεστ 19



εικόνα 93 : Τεστ 20

Η επιβεβαίωση της καλής λειτουργίας όλων των περιορισμών ολοκληρώθηκε, αφού στο πρωτο τεστ δεν επιτράπηκε η τοποθέτηση μαύρης πέτρας για την ένωση των δύο

μαύρων ομάδων. Ενώ στο δεύτερο, μετά την τοποθέτησή της ενωθήκαν οι δύο ομάδες των τριών μαύρων πετρών, αφαιρέθηκε σωστά και η ομάδα των τριών άσπρων και ενεργοποιήθηκαν τα κατάλληλα σήματα όπως αναμενόταν.

5.4 Επιβεβαίωση Λειτουργίας Μονάδας Μέτρησης Σκορ

Ένα επιπλέον καινούριο και διαφορετικό στοιχείο της νέας αρχιτεκτονικής είναι η μονάδα, η οποία είναι υπεύθυνη για την καταμέτρηση σκορ στο τέλος κάθε παιχνιδιού. Όπως αναλύθηκε και σε προηγούμενο κεφάλαιο η μονάδα, ουσιαστικά, μετρά του άσσους σε μια ακολουθία 162 bits χρησιμοποιώντας ένα Reduction Tree από αποκωδικοποιητές και απλούς αθροιστές των 2 bits στο τελευταίο επίπεδο.

Η πρώτη επιβεβαίωση λειτουργίας του έγινε πριν ενσωματωθεί στο συνολικό σύστημα. Μέσω ενός τεστ δόθηκαν αρκετές τιμές μήκους 162 bits με διαφορετικό αριθμό άσσων σε διαφορετικές θέσεις της ακολουθίας. Σε όλες τις περιπτώσεις το αποτέλεσμα της μονάδας μέτρησης συνέπεσε σωστά με τον αριθμό των άσσων.

Στη συνέχεια, αφού ενσωματώθηκε η μονάδα στη σχεδίαση, προσθέσαμε σε κάθε προηγούμενο τεστ δυο κινήσεις πόσου, ώστε να τελειώσει το παιχνίδι και να μετρηθούν οι πόντοι. Όπως και προηγουμένως, η μέτρηση ήταν ακριβής με την μονάδα να έχει αποτέλεσμα 81 + (αριθμός άσπρων πετρών - αριθμός μαύρων πετρών) σε κάθε περίπτωση.

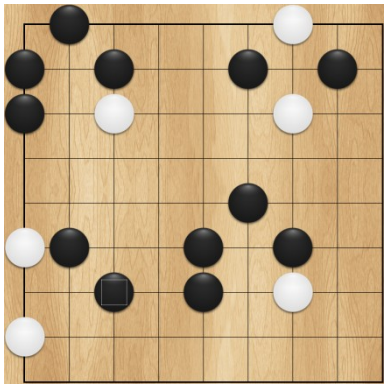
5.5 Επιβεβαίωση Λειτουργίας Μονάδας Αναγνώρισης Προτύπων 3X3

Όπως και με την προηγούμενη μονάδα, έτσι και τώρα πρώτα υλοποιήθηκε και ελέγχτηκε η μονάδα αυτόνομα και έπειτα ενσωματώθηκε στο σύστημα όπου και ελέγχτηκε ξανά. Όπως αναλύθηκε στο $3^{ο}$ κεφάλαιο η μονάδα δέχεται σαν είσοδο το γινόμενο των κωδικών των 8 περιμετρικών θέσεων κάθε θέσης που ελέγχεται. Αν αυτό το γινόμενο ταυτίζεται με κάποιο από τα συγκεκριμένα γινόμενα των προτύπων και έχει σειρά να τοποθετηθεί το προτεινόμενο, από τα πρότυπα, χρώμα τότε εμφανίζεται σήμα ταύτισης αλλά και οι συντεταγμένες τοποθέτησης νέας πέτρας.

Η πρώτη δοκιμή, η οποία έγινε, περιελάμβανε είσοδο γινομένων και χρώματος που έχει σειρά να παίξει στο σύστημα, τα οποία εναλλάξ δίνουν εμφάνιση προτύπου.

Όντως η μονάδα λειτούργησε σωστά ενεργοποιώντας το σήμα εύρεσης κάθε δεύτερη φορά.

Στη συνέχεια υλοποιήθηκε η F.S.M. ελέγχου και συντονισμού του συστήματος και ένα αντίγραφο της μονάδας αναγνώρισης προτύπων, ώστε να μοιραστεί το πλαίσιο παιχνιδιού και ο έλεγχος στα δυο, ενσωματώθηκε η μονάδα στο σύστημα. Συμφώνα με την F.S.M. η μια μονάδα ελέγχει το πάνω μισό του πλαισίου παιχνιδιού από την θέση (1,1) έως την (5,5) και η άλλη το υπόλοιπο από την (5,5) έως την (9,9). Το τεστ, που δημιουργήθηκε περιελάμβανε την τοποθέτηση τεσσάρων "οκτάδων" πετρών σε αντίστοιχα σημεία στα δυο μέρη του πλαισίου παιχνιδιού.



εικόνα 94 : Τεστ 21

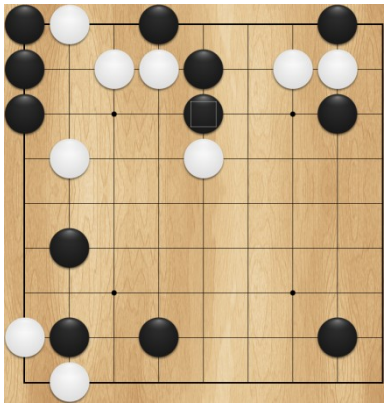
Η δοκιμή που έγινε επιβεβαίωσε τη μερικώς ορθή λειτουργία της μονάδας αλλά και την μικρή δυσλειτουργία που παρουσιάζει. Δηλαδή, ξεκινώντας από τις θέσεις (1,1) και (5,5) απορρίπτονται οι πρώτες οκτάδες, ενώ οι δεύτερες επιβεβαιώνονται. Εκείνη της θέσης (2,7) σωστά, διότι είναι όντως πρότυπη, αλλά εκείνη της θέσης (7,2) λανθασμένα αφού απλά πληροί την προϋπόθεση του γινομένου. Η δυσλειτουργία αυτή είχε προβλεφτεί, οπότε δεν αποτελεί πρόβλημα. Επίσης παρατηρήθηκε ότι η μονάδα ορθώς είχε στην έξοδο της τις συντεταγμένες (2.7), αφού σε περίπτωση κοινής εύρεσης επιλέγονται εκείνες της πρώτης μονάδας ταύτισης.

5.6 Επιβεβαίωση Λειτουργίας Συνολικού Συστήματος

Εφόσον ελέγχτηκαν και ενσωματώθηκαν όλες οι καινούριες μονάδες και τροποποιήσεις στην αρχική σχεδίαση δημιουργήθηκε ένα τεστ για τον έλεγχο όλου του συστήματος συνολικά. Για το λόγο αυτό, δημιουργήθηκε μια απλή γεννήτρια "τυχαίων" συντεταγμένων, η οποία όποτε της ζητηθεί, γεννά συντεταγμένες προσομοιώνοντας έναν απλό αντίπαλο παίχτη.

Το τεστ ξεκινάει τροφοδοτώντας το σύστημα με το χρώμα μας. Αυτό είναι το άσπρο, οπότε δεν χρειάζεται να εισάγουμε συντεταγμένες, διότι ο παίχτης που έχει το

μαύρο χρώμα ξεκινά. Όπως ορίστηκε σε παραπάνω κεφάλαιο κατά την διάρκεια των πρώτων 22 κινήσεων δεν γίνεται έλεγχος για εύρεση πρότυπης κίνησης. Μέχρι τότε το παιχνίδι έχει την εξής μορφή:

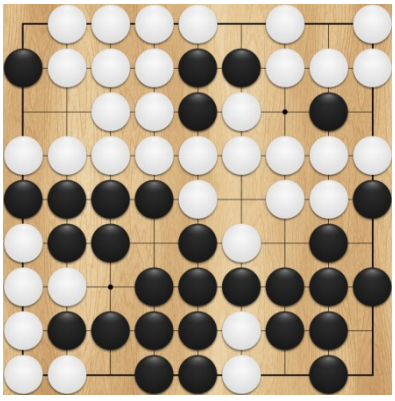


εικόνα 95 : Τεστ 22

Παρατηρούμε ότι υπάρχουν δυο παραπάνω μαύρες πέτρες, διότι σύμφωνα με την τακτική, που χρησιμοποιήθηκε, μετά από 50 άστοχες συντεταγμένες ο εικονικός παίχτης πάει πάσο. Λόγω την ντετερμινιστικής "τυχαίας" συμπεριφοράς της γεννήτριας, όσες φορές εκκινήσει το παιχνίδι με την αντίπαλη κίνηση να προσομοιώνεται πρώτη θα έχουμε το ίδιο αποτέλεσμα στις πρώτες 22 κινήσεις.

Στη συνέχεια, με την ενεργοποίηση και της αναζήτησης πρότυπης κίνησης το σύστημα αντιγράφει το υπάρχον πλαίσιο παιχνιδιού στον κλώνο, όπου και συνεχίζεται η τυχαία προσομοίωση μέχρι την 100 κίνηση. Μόλις φτάσει αυτό τον αριθμό κινήσεων καταμετράται το σκορ αδειάζει εντελώς το πλαίσιο παιχνιδιού κλώνος, ξανά αντιγράφεται το πρότυπο και πλέον γίνεται η ίδια διαδικασία για μια δεύτερη κίνηση. Αφού μετρηθεί και εκείνης το σκορ έπειτα, διαλέγεται η καλύτερη και τοποθετείται στο πρότυπο πλαίσιο παιχνιδιού.

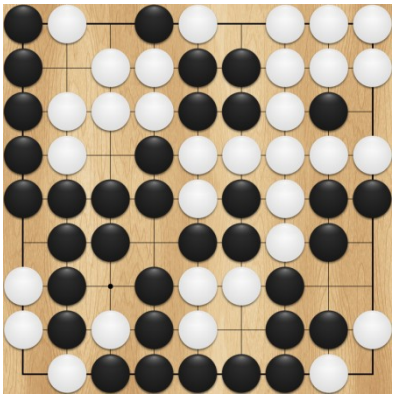
Η πρώτη κίνηση που προσομοιώνεται είναι η τοποθέτηση άσπρης πέτρας στο σημείο (4,6). Μετά το τέλος της προσομοίωσης το πλαίσιο παιχνιδιού έχει αυτή την μορφή:



εικόνα 96 : Τεστ 23

Κατά την διάρκεια της προσομοίωσης παρατηρήθηκαν αιχμαλωτίσεις, τοποθετήσεις άσπρων πετρών σε πρότυπες θέσεις αλλά και σε θέσεις όπου ανήκαν σε αιχμάλωτες πέτρες. Το σκορ, που καταμετρήθηκε στο τέλος της προσομοίωσης έδειχνε σωστά +7 άσπρες πέτρες. Πρέπει να σημειωθεί η αδυναμία, που έχει η μονάδα ελέγχου των κινήσεων να εφαρμόζει "τακτικό" έλεγχο μόνο στις κινήσεις που αφορούν το δικό της χρώμα και όχι του αντιπάλου. Για το λόγο αυτό επιτράπηκαν και κάποιες κακές αντίπαλες κινήσεις, που οδηγούσαν στην αιχμαλωσία - αυτοκτονία μαύρων πετρών. Επίσης, σύμφωνα με την σχεδίαση, έπειτα από 50 αποτυχημένες αναζητήσεις για ένα χρώμα θεωρητικά ο παίχτης πάει πάσο και ελέγχεται το άλλο χρώμα. Δυο διαδοχικές κινήσεις πόσου τερματίζουν το παιχνίδι σύμφωνα με τους κανόνες. Αυτό κατά την διάρκεια της προσομοίωσης δεν εφαρμόζεται. Μετά από 50 αποτυχημένες αναζητήσεις τοποθέτησης άσπρης πετράς ο παίχτης πάει πάσο, αυξάνεται ο αριθμός των κινήσεων κατά 1 και συνεχίζεται η αναζήτηση τοποθέτησης αντιθέτου χρώματος. Το παιχνίδι τελειώνει αμέσως μόλις ο αριθμός κινήσεων ξεπεράσει το νούμερο 100.

Στη συνέχεια το πρόγραμμα προσομοιώνει την κίνηση (1,5), με λιγότερες "κακές" και άναρχες κινήσεις από τον αντίπαλο,



εικόνα 97 : Τεστ 24

ο οποίος κερδίζει για 3 πέτρες αυτή την φορά. Και στην προσομοίωση αυτή υπήρξαν αιχμαλωσίες πετρών και τοποθετήσεις πετρών σύμφωνα με τα πρότυπα.

Η διαδικασία που ακολουθεί είναι η διαγράφη του πλαισίου παιχνιδιού κλώνος και η τοποθέτηση της κίνησης με το καλύτερο σκορ στο πρότυπο πλαίσιο παιχνιδιού. Όντως, το σύστημα διαλέγει την πρώτη προτεινόμενη κίνηση την τοποθετεί στο πλαίσιο παιχνιδιού και συνεχίζει επαναλαμβάνοντας την ίδια διαδικασία για τις επόμενες κινήσεις. Κάθε προηγούμενη προσομοίωση κράτησε περίπου 19000 κύκλους ρολογιού - περιόδους. Αυτό, αποτελεί και το ανώτατο όριο, καθώς όσο γεμίζει το πλαίσιο παιχνιδιού τόσο θα ελαττώνεται ο απαιτούμενος χρόνος προσομοίωσης.

Η ενίσχυση της σχεδίασης με την μέτρηση του σκορ και η ρεαλιστικότερη προσομοίωση του παιχνιδιού με την ελεγχόμενη τοποθέτηση πετρών σε περιοχές Eye ή σε πρότερα σημεία αιχμάλωτων πετρών λειτουργούν σωστά. Επιπλέον,

δοκιμαστικά τοποθετήθηκε και η μονάδα ταύτισης πρότυπων κινήσεων προσπαθώντας να επιλεχτούν οι καλύτερες, σημαντικότερες και πιο χρήσιμες πρότυπες κινήσεις προς ταύτιση. Παρά την μικρή δυσλειτουργία, η οποία υπάρχει και είχε προβλεφτεί η μονάδα αυτή πιθανών να ενισχύει την αξιολόγηση κάθε κίνησης. Σίγουρα, δεν την αποδυναμώνει, διότι ακόμη και οι πρότυπες προτεινόμενες κινήσεις ελέγχονται πριν τοποθετηθούν στο πλαίσιο παιχνιδιού.

5.7 Αποτελέσματα - Συμπεράσματα

Η FPGA η οποία χρησιμοποιήθηκε ως πρότυπο για την μέτρηση του κόστους της σχεδίασης αλλά και τον υπολογισμό της συχνότητας λειτουργίας της είναι μια Virtex 5 XC5VLX110T. Η σχεδίαση έγινε εξ' ολοκλήρου χρησιμοποιώντας το εργαλείο Xilinx ISE και πιο συγκεκριμένα την έκδοση 14.4 με speed grade -1. Η σχεδίαση έφτασε μέχρι το επίπεδο Place & Route, από όπου και θα σας παρουσιάσω τα αποτελέσματα παρακάτω. Το εργαλείο με το οποίο ελέγχτηκαν τα Test Benches, που υλοποιήθηκαν για την επιβεβαίωση λειτουργίας της σχεδίασης είναι το ISim.

5.7.1 Κόστος Συνολικής Σχεδίασης και Επιμέρους Μονάδων

Συνολικά η νέα σχεδίαση καταλαμβάνει λίγο περισσότερο από τον διπλάσιο χώρο από ότι η προηγούμενη του Σ. Κόκκαλη. Βέβαια, περιλαμβάνει ένα πλαίσιο παιχνιδιού κλώνο και κάποιες επιπλέον μονάδες. Όπως θα παρατηρήσουμε και παρακάτω τα αποτελέσματα είναι όπως τα περιμέναμε καθώς το πλαίσιο παιχνιδιού είναι εκείνο που απαιτεί τους περισσότερους πόρους.

Γενική Κατανομή Σχεδίασης στην FPGA

Logic Utilization:

Slice Logic	Used/ Available	FPGA Percentage
Slice Registers	3506/69120	**5%**
Slice LUTs	14061/69120	**20%**
Slice LUTs used as Logic	14046/69120	**20%**
Slice LUTs used as Memory	15/17920	**0%**

Logic Distribution:

Slice Logic	Used	
LUT Flip Flop pairs	16524	
LUT Flip Flop Pairs	**Number/ Used**	**Flip Flop Pairs Percentage**
Pairs with an unused Flip Flop	13018/16524	**78%**
Pairs with an unused LUT	2463/16524	**14%**
Fully used LUT-FF pairs	1043/16524	**6%**

IO Utilization:

Number of IOs	69	
Bonded IOs	**Used/ Available**	**IOBs Percentage**
IOBs	45/640	**7%**

Αν και δεν είναι μεγάλος ο αριθμός, των I/O, που χρησιμοποιεί το σύστημα, η μονάδα αυτή δεν αποτελεί πρόβλημα. Αυτό, γιατί το σύστημα λειτουργεί αυτόνομα και ανεξάρτητα από κάποιον επεξεργαστή, χωρίς να απαιτεί επικοινωνία και ανταλλαγή πληροφοριών. Επιπροσθέτως, το ποσοστό των BRAM είναι ελάχιστο, διότι η συγκεκριμένη αρχιτεκτονική υλοποιήθηκε με λογική στην οποία και στηρίζεται, μη απαιτώντας μνήμη.

Κατανομή Πόρων Σχεδίασης

Πιο αναλυτικά όσο αφορά την συνολική σχεδίαση οι απαιτήσεις σε συγκεκριμένους πόρους στοιχεία της FPGA είναι:

Module	**Registers**	**Adders/ Subtractors**	**Comparators**	**Multiplexers**	**XOR Gates**
Ανώτατο Επίπεδο (**Top Module**)	3574	182	1206	50	35

Αναλυτικότερα, οι επιμέρους μονάδες, των οποίων τα σχηματικά διαγράμματα παρουσιάστηκαν στο προηγούμενο κεφάλαιο χρησιμοποιούν συγκεκριμένο αριθμό στοιχείων - πόρων όπως προκύπτει από τους παρακάτω πίνακες.

Επιμέρους Μονάδες του Top Module	**Registers** (% Συνολικά) (% Top Module)	**Adders/ Subtractors** (% Συνολικά) (% Top Module)	**Comparators** (% Συνολικά) (% Top Module)	**Multiplexers** (% Συνολικά) (% Top Module)	**XOR Gates** (% Συνολικά) (% Top Module)
Μονάδα Εντολών (**Instruction Module**)	432(12%)(12%)	3(2%)(2%)	8(1%)(1%)	-	6(12%)(12%)
Μονάδα Πλαισίου Παιχνιδιού (**Board Module**)	3142(88%)(88%)	179(98%)(98%)	1198(99%)(99%)	50(100%)(100%)	29(82%)(82%)

Επιμέρους Μονάδες του Board Module	**Registers** (% Συνολικά) (% Board Module)	**Adders/ Subtractors** (% Συνολικά) (% Board Module)	**Comparators** (% Συνολικά) (% Board Module)	**Multiplexers** (% Συνολικά) (%Board Module)	**XOR Gates** (% Συνολικά) (%Board Module)
Μονάδα Ελεγκτή - Ρυθμιστή (**Controller**)	108(3%)(3%)	16(8%)(9%)	63(5%)(5%)	-	25(71%)(86%)
Μονάδα Πλαισίων Παιχνιδιού (**Boards**)	2592(72%)(82%)	162(90%)(91%)	1134(94%)(95%)	-	-
Μονάδα Μέτρησης Σκορ (**Score Counter**)	196(5%)(6%)	-		50(100%)(100%)	4(11%)(14%)
Μονάδα Αναγνώρισης Πρότυπων Κινήσεων(**Pattern Recogniser**)	246(8%)(9%)	1(0%)(0%)	1(0%)(0%)	-	-

Επιμέρους Μονάδες του Boards	**Registers** (% Συνολικά) (% Boards)	**Adders/ Subtractors** (% Συνολικά) (% Boards)	**Comparators** (% Συνολικά) (% Boards)	**Multiplexers** (% Συνολικά) (% Boards)	**XOR Gates** (% Συνολικά) (% Boards)
Μονάδα Θέσης (**Position**)	16(0%)(0%)	1(0%)(0%)	7(0%)(0%)	-	-

Παρατηρούμε ότι το μεγαλύτερο ποσοστό πόρων καταλαμβάνει η υλοποίηση των δύο πλαισίων παιχνιδιού με την συνεισφορά κάθε θέσης να είναι μηδαμινή αν εξεταστεί ανεξάρτητα από το σύνολο . Οι υπόλοιπες μονάδες, οι οποίες τροφοδοτούν το σύστημα με χρήσιμες πληροφορίες για να λαμβάνει αποφάσεις καταλαμβάνουν σχετικά πολύ μικρό ποσοστό. Άρα, με προσεκτική σχεδίαση θα μπορούσαν να γίνουν πιο ισχυρές και πιο πολύπλοκες παρέχοντας καλύτερες πληροφορίες, ενισχύοντας την διαδικασία απόφασης επόμενης κίνησης. Βέβαια, τόσο το ποσοστό των πλαισίων παιχνιδιού, όσο και αυτό της συνολικής σχεδίασης δεν είναι τόσο απαιτητικό σε πόρους.

Η συχνότητα λειτουργίας που υπολογίστηκε από το εργαλείο είναι ελάχιστα πάνω από τα 10 MHz. Η τιμή αυτή για σχεδίαση σε μια FPGA είναι αρκετά χαμηλή. Δικαιολογείται όμως, εν μέρη, λόγω του πολύ μεγάλου Latency που δημιουργείται. Τα bits, που μεταφέρονται από τα πλαίσια παιχνιδιού προς και από την μονάδα Controller(Master) αλλά και εκείνα που αποστέλλονται στη γενικότερη μονάδα που υπολογίζει το σκορ, την πληρότητα του πλαισίου παιχνιδιού και τις πρότυπες κινήσεις είναι πάρα πολλά και σε μεγάλο βαθμό διαφορετικά μεταξύ τους. Έχοντας τόσο μεγάλο αριθμό bits, που διαβιβάζονται από μονάδα σε μονάδα το Latency αυξάνεται εξίσου μειώνοντας, δυστυχώς, την συχνότητα λειτουργίας. Προσπαθώντας

να μειώσουμε η συνδυαστική έκρηξή του παιχνιδιού Go, πληρώνουμε σε πληροφορίες που μεταφέρονται για να εξυπηρετηθούν έλεγχοι, οι οποίοι έχουν σκοπό να αποκλείσουν μελλοντικά αρνητικά αποτελέσματα που θα προκαλέσουν κάποιες κινήσεις. Όπως είναι φυσικό, η πολυπλοκότητα αυξάνεται και συμβάλλοντας και εκείνη αρκετά στην μείωση στης συχνότητας του ρολογιού και την απόδοση του συστήματος, βελτιώνοντας όμως τις τακτικές επιδόσεις.

Παρά την συνολική επίτευξη του στόχου της διπλωματικής εργασίας, ο οποίος ήταν η "εξέλιξη" και η πιστοποιημένη λειτουργικότητα της προηγούμενης σχεδίασης - ώστε να δημιουργηθεί ένα πλαίσιο παιχνιδιού τυχαίων προσομοιώσεων το οποίο συνδυάζοντας τεχνική και τακτική να επιλέγει καλύτερα την επόμενη κίνηση σε κάθε στάδιο - η αντιμετώπιση της νέας σχεδίασης έγινε κάτω από έναν άξονα βελτίωσης. Η μονάδα καταμέτρησης του σκορ και η μονάδα αναγνώρισης πρότυπων κινήσεων είναι πλήρως pipelined. Όσο αφορά τη μονάδα παραγωγής κινήσεων δεν θα μπορούσε κάθε κύκλο να δημιουργεί μια νέα έγκυρη κίνηση. Προφανώς, είναι εφικτό κάθε κύκλο να γεννά καινούριες συντεταγμένες, και να ελέγχονται προς εγκυρότητα εκείνες που ταυτίζονται με την χρονική στιγμή αναζήτησης νέας κίνησης. Βέβαια, αυτό δεν θα είχε νόημα και αντίκρισμα, αντιθέτως, επειδή ο L.F.S.R. είναι "περιοδικός", κάτι τέτοιο μπορεί να οδηγούσε στον ταχύτερο επανέλεγχο ίδιων κινήσεων, που είχαν απορριφτεί προηγουμένως. Από την άλλη, δεν είναι εφικτή η δημιουργία έγκυρης κίνησης κάθε νέο κύκλο ρολογιού. Η εγκυρότητα κάθε κίνησης εξαρτάται από την κατάσταση του πλαισίου παιχνιδιού πριν την τοποθέτηση της. Έχοντας δεδομένο ότι η τοποθέτηση μια νέας πέτρας αποτελεί έναν αριθμό εντολών εξαρτώμενο από τις γειτονικές πέτρες σημαίνει ότι η δημιουργία κάθε νέου σετ εντολών πρέπει να συνεπάγεται ελέγχου των πιο ενημερωμένων γειτονικών πληροφοριών. Επίσης, θα μπορούσε αν γινόταν πιο πολύπλοκος ο έλεγχος και αυξάνονταν το μήκος της εντολής τοποθέτησης πέτρας, σχεδόν 4 φόρες, να διαρκεί ένα κύκλο η τοποθέτηση της, άρα να είναι πιο νωρίς έτοιμες οι πιο ενημερωμένες γειτονικές πληροφορίες αυξάνοντας σίγουρα το Latency και την πολυπλοκότητα. Στο παραπάνω διμερές trade - off επιλέχτηκε η λύση των 5 κύκλων(1 για την τοποθέτηση, 4 για την αλληλεπίδραση με τους γείτονες) για την τοποθέτηση κάθε νέας πέτρας και έπειτα αίτημα προς την γεννήτρια για δημιουργία νέας κίνησης.

Ένα ακόμη πρόβλημα, το οποίο δημιούργησε η τελική συχνότητα λειτουργίας είναι η απόρριψη της ιδέας δημιουργίας ενός demo. Η χαμηλή συχνότητα της σχεδίασης απαιτεί προσθήκη "ρυθμιστών ρολογιού"(D.C.M.), ώστε και η συχνότητα λειτουργίας της FPGA που επιλέχτηκε να ταυτιστεί. Επίσης, επειδή σχεδιάστηκε ο χρηστής να τροφοδοτεί το σύστημα μέσω πληκτρολογίου για τις κινήσεις του αντιπάλου αλλά και για άλλες πληροφορίες και να λαμβάνει τα αποτελέσματα μέσω μιας οθόνης, αυτό δημιουργεί περεταίρω εργασία και τροποποιήσεις. Αρκετή, δουλειά θα πρέπει να γίνει για το συγχρονισμό των διαύλων(F.S.L.) μεταφοράς πληροφοριών από και προς την αρχιτεκτονική στο συνολικό σύστημα.

Λαμβάνοντας υπό όψιν την χειρότερη και πιο καθυστερημένη των περιπτώσεων η ολοκλήρωση ενός τυχαίου παιχνιδιού διαρκεί περίπου 27.000 κύκλους κάνοντας το σύστημα να μπορεί να παίξει περίπου 370 "βαριά" παιχνίδια/sec. Οπότε έχοντας ένα εύλογο χρονικό όριο(timeout) για να απαντήσει το σύστημα, θα μπορεί να παίζει πάνω από 1000 παιχνίδια για κάθε μια από τις δύο κινήσεις που ελέγχει. Επίσης, όσο γεμίζει το πλαισίου παιχνιδιού τόσο λιγότερο χρόνο θα απαιτεί η ολοκλήρωση ενός παιχνιδιού, άρα θα αυξάνονται οι προσομοιώσεις MC.

5.7.2 Συσχέτιση και Σύνδεση με την Προηγούμενη Σχεδίαση

Η προηγούμενη σχεδίαση του κ. Κόκκαλη, αποτέλεσε το έναυσμα αλλά και την βάση για το σύστημα το οποίο υλοποιήθηκε στην εργασία αυτή. Η υλοποίησή της έγινε με στόχο την βελτίωση και επιτάχυνση του πιο απαιτητικού μέρους ενός κώδικα. Ο κώδικας αυτός ήταν κομμάτι του GNUGo, ενός από τα πιο επιτυχημένα προγράμματα στο τομέα Computer Go. Το παραπάνω κομμάτι αφορούσε την συνάρτηση, ή οποία χρησιμοποιούνταν κατά την προσομοίωση τυχαίων παιχνιδιών. Έπειτα από κατάλληλο profiling, διαπιστώθηκε ότι τον περισσότερο χρόνο απαιτούσε η τοποθέτηση πέτρας στο πλαίσιο παιχνιδιού και η αλληλεπίδρασή της με τις γειτονικές. Για το λόγο αυτό, δημιουργήθηκαν οι κατάλληλες δομές, που υλοποιούσαν ένα πλαίσιο παιχνιδιού και την τοποθέτηση πετρών σε αυτές με ταχύτερο τρόπο από ότι το software.

Η νέα σχεδίαση πέτυχε να κάνει πιο ρεαλιστική την λειτουργία του πλαισίου παιχνιδιού επιτρέποντας κινήσεις σε περιοχές(Eye, captured territory) που απέφευγε η προηγούμενη σχεδίαση. Επίσης, δημιουργήθηκε η πολύ σημαντική μονάδα Μέτρησης Σκορ στο τέλος κάθε παιχνιδιού, αλλά και η Γεννήτρια Συντεταγμένων. Συνεπώς, η νέα σχεδίαση μπορούσε να υλοποιήσει προσομοίωση τυχαίου παιχνιδιού. Ακόμη, έγινε προσπάθεια να ενισχυθεί η μονάδα Απόφασης Επόμενης Κίνησης από ένα σύνολο πρότυπων κινήσεων αλλά και κάποιους ευριστικούς κανόνες. Άρα, με βάση την αρχιτεκτονική του πλαισίου παιχνιδιού υλοποιήθηκε ένα σύστημα – παίχτης, το οποίο έχει την δυνατότητα να προσομοιώνει δυο τυχαίες κινήσεις και να επιλέγει την καλύτερη, κάνοντας τις λιγότερο κακές επιλογές κατά την προσομοίωση.

Προφανώς, η περαιτέρω εξέλιξη της εργασίας με σκοπό την απτή υλοποίηση της σε μια FPGA είναι επιτακτική. Αυτό σημαίνει μελέτη και υλοποίηση pipeline καταχωρητών όπου είναι εφικτό για την βελτίωση της συχνότητας ή οποιαδήποτε τροποποίησή της με τον παραπάνω στόχο. Έτσι ώστε, να υπάρξει η δυνατότητα καθορισμού χρονικού ορίου(timeout) για τον υπολογισμό των δυνατών προσομοιώσεων παιχνιδιών/sec.

5.7.3 Σύγκριση υλοποίησης με Software

Τα προγράμματα τα οποία έχουν υλοποιηθεί μέχρι στιγμής αποτελούν μέρος μελετών και επιστημονικών ερευνών αρκετών χρόνων. Ως εκ τούτου καθένα από αυτά θα μπορούσε να χαρακτηριστεί state of the art. Επίσης, η δουλεία χρόνων και ανθρώπινου δυναμικού η οποία έχει γίνει για την δημιουργία και εξέλιξη κάθε προγράμματος δεν θα μπορούσε εύκολα να συγκριθεί με την σχεδίαση αυτής της εργασίας. Επιπροσθέτως, η πλειοψηφία του software, που χρησιμοποιεί τεχνικές Monte Carlo για την αξιολόγηση των κινήσεων υλοποιείται σε πολύ – πύρηνα συστήματα των οποίων η απόδοση αγγίζει τα 15 Teraflops και η συχνότητα λειτουργίας ξεπερνά τα 4.5 GHz. Πιο συγκεκριμένα το πρόγραμμα MoGo κατάφερε να κερδίσει έναν επαγγελματία παίχτη με μόλις 1,5 πόντο, έχοντας λάβει και Handicap 9 πετρών. Αυτό συνέβη όταν το MoGo "έτρεχε" στο Huygens cluster(25 nodes, 800 cores, 4 cpus per node with each core running at 4.7 GHz to produce 15 Teraflops) του Άμστερνταμ.

Μια ποσότητα, η οποία μπορεί να αποτελέσει μέτρο σύγκρισης είναι οι προσομοιώσεις/sec με τα παραπάνω προγράμματα να υλοποιούν στα συγκεκριμένα συστήματα από 30.000 μέχρι 65.000 προσομοιώσεις/sec τοποθετώντας πέτρες μόνο στα διαθέσιμα σημεία, εκτός "αιχμάλωτων περιοχών" και "Eye" κάνοντας τα προγράμματα πολύ απλά και γρήγορα. Όπως αναφέρθηκε και παραπάνω η συγκεκριμένη σχεδίαση έχει την δυνατότητα να παίξει λίγες περισσότερες από 370 προσομοιώσεις/sec. Με κατάλληλη τροποποίηση και εξέλιξη, βέβαια, μπορεί να ξεπεράσει τις 1000 προσομοιώσεις/sec, αριθμός κατάλληλος για την σημαντική ελάττωση της αβεβαιότητας των τυχαίων παιχνιδιών και εμφάνισης της ισχύος της στατιστικής. Όμως δεν μπορούμε να ισχυριστούμε ότι αυτό είναι αναγκαίο.

Είναι γεγονός, όμως, τα παραπάνω συστήματα υλοποιούν 175 φορές περισσότερες προσομοιώσεις/sec καταναλώνοντας, όμως, εξαιρετικά περισσότερη ενέργεια από μια Virtex 5. Επιπλέον, οι εντελώς τυχαίες κινήσεις των συστημάτων αυτών σε σχέση με την ενεργειακή κατανάλωσή τους κάνουν την ιδέα χρήσης ενός αριθμού FPGA των 370 προσομοιώσεων/sec, οι οποίες θα υλοποιούν για την ίδια κίνηση διαφορετικές προσομοιώσεις με τους υπάρχοντες ευριστικούς κανόνες και την υπάρχουσα βάση πρότυπων κινήσεων, αρκετά ελκυστική.

5.7.4 Παραλληλισμός

Έπειτα από τον υπολογισμό των χωρικών απαιτήσεων της σχεδίασης, είναι εύκολος ο ισχυρισμός, δημιουργίας επιπλέον Μονάδων Πλαισίου Παιχνιδιού, Γεννητριών Κινήσεων, αλλά και Μονάδων Ελεγκτή - Ρυθμιστή στην ίδια FPGA Οι μονάδες αυτές θα μπορούν να λειτουργούν παράλληλα χωρίς κανένα πρόβλημα, αξιολογώντας μέσω τυχαίων προσομοιώσεων περάτωσης παιχνιδιών επιπλέον κινήσεις. Συνεπώς, υπάρχει ή δυνατότητα ύπαρξης περισσότερων πλαισίων

παιχνιδιού, που θα λειτουργούν ταυτόχρονα και ανεξάρτητα. Επιπροσθέτως, η Μονάδα Θέσης, υλοποιείται στην συγκεκριμένη σχεδίαση 81 φορές εξαρτώμενη μόνο από τους γείτονες της. Αυτό σημαίνει, ότι είναι άμεσα παραμετροποιήσιμο το σύστημα ώστε να υλοποιηθεί μεγαλύτερο πλαίσιο παιχνιδιού. Ακόμη, η συγκεκριμένη υλοποίηση θέσης ευνοεί την δημιουργία συστημάτων, τα οποία για τακτικούς σκοπούς μπορούν να σαρώνουν είτε κατά σειρά, είτε κατά στήλη όλο το πλαίσιο παιχνιδιού.

Στην συγκεκριμένη σχεδίαση έχει υλοποιηθεί η Μονάδα Αναγνώρισης Πρότυπων Κινήσεων, η οποία αποτελείται από δυο ελεγκτές που διασχίζουν επιτυχώς ταυτόχρονα τα δυο μισά του πλαισίου παιχνιδιού για εύρεση πρότυπης κίνησης. Η υλοποίηση μεγαλύτερου πλαισίου παιχνιδιού, ή περισσότερων πλαισίων δεν θα αυξήσει τον μέγιστο χρόνο που απαιτείται για την εύρεση πρότυπης κίνησης. Θα χρειαστεί, μόνο, η υλοποίηση περισσότερων μονάδων ελέγχου.

Γενικά, ο παραλληλισμός μπορεί να εμφανιστεί στην επαναλαμβανόμενη χρήση του κυρίους μέρους της συγκεκριμένης σχεδίασης. Οι μονάδες, που είναι υπεύθυνες για την προσομοίωση τυχαίων παιχνιδιών έως το τέλος, αποτελούν το κύριο μέρος. Στο επόμενο κεφάλαιο αναφέρονται κάποιες προτάσεις χρήσης του σε παράλληλα συστήματα, που εξυπηρετούν τον έλεγχο όσο δυνατόν περισσότερων κινήσεων παράλληλα.

Κεφάλαιο 6

Μελλοντική Επέκταση

Όπως είναι προφανές η μελλοντική επέκταση ενός συστήματος, συνεπάγεται με την βελτίωσή του στον τομέα της ταχύτητας, της κατανάλωσης ενέργειας, των χρησιμοποιούμενων πόρων, της συνολικής αποδοτικότητας.

Παρά τη χαμηλή συχνότητα λειτουργίας του συστήματος σαν πρώτη μελλοντική επέκταση θα πρότεινα τον διπλασιασμό των απαραίτητων μονάδων ώστε να υπάρχουν δύο πλαίσια παιχνιδιού κλώνοι, τα οποία να λειτουργούν ταυτόχρονα, ώστε είτε να προσομοιώνεται η ιδία κίνηση και στα δύο πλαίσια παιχνιδιού, είτε να ελέγχονται οι δύο προτεινόμενες κινήσεις ταυτόχρονα, είτε να ελέγχονται σχεδόν στον ίδιο χρόνο με τώρα τέσσερις αντί για δύο κινήσεις. Βέβαια απαιτείται προσοχή συγχρονισμού στην περίπτωση της παράλληλης λειτουργίας, ώστε θεωρητικά να τελειώνουν μαζί τα παιχνίδια και να αξιολογούνται οι κινήσεις. Κάτι τέτοιο θα διπλασιάσει των αριθμό των προσομοιώσεων/sec για μια κίνηση ή θα μειώσει τον χρόνο ελέγχου των δύο κινήσεων ή θα ελέγξει στον ίδιο χρόνο διπλάσιο αριθμό κινήσεων αντίστοιχα.

Δεδομένου ότι ο έλεγχος αν το πλαίσιο παιχνιδιού είναι γεμάτο και ο υπολογισμός του σκορ στο τέλος είναι αναπόφευκτα και αναπόσπαστα κομμάτια της σχεδίασης το Latency που δημιουργούν δεν μπορούμε να το αποφύγουμε. Επίσης, είναι απαραίτητο στην μονάδα Master να μεταβαίνουν όλες οι απαραίτητες πληροφορίες αναγκαστικά για την έκδοση σωστών μηνυμάτων. Συνεπώς κάποια bits τα οποία διαβιβάζονται μεταξύ των μονάδων είναι απαραίτητα. Για αυτό, μια μελλοντική προσέγγιση θα ήταν η αξιολόγηση των πρότυπων κινήσεων. Εύκολα θα μπορούσε και το υπάρχον σύστημα αντί για δεύτερη προτεινόμενη κίνηση να προσομοιώνει την ίδια χωρίς ύπαρξη μονάδας ταύτισης πρότυπων κινήσεων. Έπειτα να συγκρίνει και να αξιολογεί, την συνεισφορά των πρότυπων κινήσεων.

Αν είναι αναγκαία η ύπαρξη τους τότε, θα μπορούσε να μοντελοποιηθεί και να χρησιμοποιηθεί κάποιος συγκεκριμένος αλγόριθμος Pattern Recognition αλλάζοντας και απλοποιώντας φυσικά την κωδικοποίηση κάθε δυνατής κατάστασης κάθε θέσης. Αυτό θα μείωνε σίγουρα σε 2×81 τα bits που θα απαιτούσε η μονάδα εύρεσης πρότυπης κίνησης, διότι κάθε θέση θα είχε τέσσερις δυνατές καταστάσεις. Έτσι θα μειωθεί σίγουρα το Latency, όμως θα αυξηθεί κατά πολύ ο αριθμός των ελέγχων. Αν, όμως, ο αλγόριθμος, που θα χρησιμοποιηθεί χρήζει παραλληλοποίησης, τότε θα έχουμε θετικά αποτελέσματα.

Από την άλλη αν η σχεδίαση με το σετ των προτύπων κινήσεων υστερεί της σχεδίασης χωρίς αυτές, τότε λαμβάνοντας υπό όψιν πως, όσο περισσότερες προσομοιώσεις MC υλοποιηθούν για μια κίνηση τόσο περισσότερο μειώνεται ο παράγοντας της αβεβαιότητας μια επέκταση με γνώμονα την ταχύτητα και μόνο θα ήταν ιδανική. Για αυτό, αφαιρώντας τις μονάδες πλαίσιο παιχνιδιού κλώνος, μετρητής σκορ αλλά και την γεννήτρια κινήσεων ώστε να μείνουν μόνο οι Controller(Master) και Board(slave) θα δημιουργούσε ένα αρκετά γρήγορο πλαίσιο παιχνιδιού. Η υλοποίηση αυτή θα δέχεται κινήσεις από μια εξωτερική γεννήτρια και στο τέλος κάθε προσομοίωσης θα κωδικοποιεί το πλαίσιο παιχνιδιού με τον ελάχιστο απαιτούμενο αριθμό από bits και θα το βγάζει σαν έξοδο, ώστε να μετριέται το σκορ και να αξιολογείται η αρχική κίνηση κάθε τυχαίας προσομοίωσης.

Μια τέτοια σχεδίαση θα χωρούσε τουλάχιστον 3 μονάδες προσομοιώσεων MC σε μια μέτρια FPGA και θα υλοποιούσε σεβαστό αριθμό τυχαίων υλοποιήσεων/sec. Οπότε αν υπάρχει η δυνατότητα κάθε μια από τις τέσσερις FPGA ενός Convey System να υλοποιήσει διαφορετική σχεδίαση, τότε τρεις θα αποτελούσαν το λιγότερο 9 πλαίσια παιχνιδιού προσομοιώσεων MC και μια θα αναλάμβανε τον ρόλο της γεννήτριας κινήσεων αλλά και υπολογισμού σκορ. Η παραπάνω σχεδίαση θα μπορεί να χρησιμοποιηθεί είτε για την καλύτερη και γρηγορότερη αξιολόγηση μιας κίνησης, η για αξιολόγηση διαφορετικού αριθμού κινήσεων ταυτοχρόνως.

References

[1] B. Brügmann, “Monte Carlo Go”, (1993).

[2] B. Bouzy, “The INDIGO program”, *in Proceedings of the 2nd Game Programming Workshop in Japan, Hakone*, (1995), pp. 197-206.

[3] B. Bouzy, “Spatial Reasoning in the game of Go”, *in Proceedings of the Workshop on Representations and Processes in Vision and Natural Language, ECAI*, (1996), pp. 78-80.

[4] B. Bouzy, “There are no wining moves except the last”, *in Proceedings IPMU-96, Canada, Spain*, (1996), pp. 197-202.

[5] B. Bouzy, T. Cazenave, “Using the object oriented paradigm to model context in computer Go”, *in Proceedings of the First International and Interdisciplinary Conference on Modeling and Using Context*, (1997).

[6] K. Chen, Z. Chen, “Static analysis of life and death in the game of Go”, *International Journal of Information Sciences-Informatics and Computer Science 121(1-2)*, (1999), pp. 113-134.

[7] B. Bouzy, T. Cazenave, “Computer Go: an AI oriented Survey”, *Artificial Intelligence 132(1)*, (2001), pp. 39-103.

[8] B. Bouzy, “Mathematical morphology applied to computer Go”, *International Journal of Pattern Recognition and Artificial Intelligence 17(2)*, (2003) pp. 257-268.

[9] B. Bouzy, “The Move Decision Strategy of Indigo”, *ICGA Journal 26(1)*, (2003), pp. 14-27.

[10] B. Bouzy, B. Helmstetter, “Monte-Carlo Go Developments”, *in Proceedings of the 10th Advances in computer Games Conference(AGS-10), eds. H.J. van den Herik, H. Iida and E.A. Heinz(Kluwer Academic)*, (2004), pp. 159-174.

[11] B. Bouzy, “Associating domain-dependent knowledge and Monte Carlo approaches within a Go program”, *Information Sciences 175(4)*, (2005), pp. 247-257.

[12] T. Cazenave, B. Helmstetter, “Combining Tactical Search and Monte-Carlo in the game of Go”, *in Proceedings of IEEE 2005 Symposium on Computational Intelligence and Games, Colchester, Essex, UK,* (2005), pp. 171-175.

[13] B. Bouzy, G. Chaslot, "Bayesian generation and integration of K-nearest-neighbor patterns for 19x19 go", *in Proceedings of IEEE 2005 Symposium on Computational Intelligence and Games, Colchester, Essex, UK,* (2005), pp. 176-181.

[14] B. Bouzy, "History and territory heuristics for Monte-Carlo Go", *New Mathematics and Natural Computation 2(2)*, (2006), pp. 1-8.

[15] G. Chaslot, J.-T. Saito, J. W. H. M. Uiterwijk, B. Bouzy, H. J. van den Herik, "Monte-Carlo Strategies for Computer Go", *in Proceedings of the 18th BeNeLux Conference on Artificial Intelligence, Namur, Belgium, eds. P. –Y. Schobbens, W. Vanhoof and G. Schwanen*, (2006), pp. 83-91.

[16] B. Bouzy, G. Chaslot, "Monte-Carlo Go Reinforcement Learning Experiments", *in Proceedings of IEEE 2006 Symposium on Computational Intelligence and Games, Reno, USA*, (2006), pp. 187-194.

[17] B. Bouzy, "Old-fashioned Computer Go vs. Monte-Carlo Go", *in Proceedings of IEEE 2007 Symposium on Computational Intelligence and Games, Honolulu, Hawaii, invited tutorial*, (2007).

[18] R. Coulom, "Efficient selectivity and backup operators in Monte-Carlo tree search", *in Proceedings of the 5th International Conference on Computers and Games, Lecture Notes in Computer Science(LNCS), eds. H. J. van den Herik, P. Ciancarini and H. H. L. M. Donkers (Springer-Verlang, Heidelberg, Germany)*, (2007), pp. 72-83.

[19] Y. Wang, S. Gelly, "Modifications of UCT and sequence-like simulation for Monte-Carlo Go", *in Proceedings of IEEE 2007 Symposium on Computational Intelligence and Games, Honolulu, Hawaii,* (2007), pp. 175-182.

[20] M. C. Herbordt, T. VanCourt, Y. Gu, B. Sukhwani, A. Conti, J. Model, D. DiSabello, "Achieving High Performance with FPGA-Based Computing", Computer 40(3), (2007), pp. 50-57.

[21] G. M. J.-B. Chaslot, M. H. M. Winands, H. Jaap Van Den Herik, J. W. H. M. Uiterwijk, B. Bouzy, "Progressive strategies for Monte-Carlo tree search", *New Mathematics and Natural Computation 4(3)*, (2008), pp. 343-357.

[22] S. Kokkalis, "Design & Architecture with Reconfigurable Logic of Data Structures for the Game of Go"-Diploma Thesis, *Technical University of Crete*, (2009).

[23] C.-S. Lee, M.-H. Wang, G. Chaslot, J.-B. Hoock, A. Rimmel, O. Teytaud, S.-R. Tsai, S.-C. Hsu, T.-P. Hong, "The Computational Intelligence of MoGo Revealed in Taiwan's Computer Go Tournaments", *in Proceedings of the*

IEEE Transactions on Computational Intelligence and AI in Games 1(1), 2009, pp. 73-89.

[24] G. Chaslot, C. Fiter, J.-B. Hoock, A. Rimmel, O. Teytaud, "Adding expert knowledge and exploration in Monte-Carlo Tree Search ", *in Proceedings of the 12th Advances in computer Games Conference(AGS-12),LNCS 6048*, (2010), pp. 1-13.

[25] A. Bourki, G. Chaslot, M. Coulm, V. Danjean, H. Doghmen, J.-H. Hoock, T. Herault, A. Rimmel, F. Teytaud, O. Teytaud, P. Vayssiere, Z. Yu, "Scalability and Parallelization of Monte-Carlo Tree Search", *in Proceedings of the 7th International Conference on Computers and Games, Kanazawa, Japan, Lecture Notes in Computer Science(LNCS)*, (2010), pp. 48-58.

[26] H. Gao, F. Wang, W. Lei, Y. Lin, "Monte-Carlo simulation of 9x9 Go game on FPGA", (2010).

[27] G. Chaslot, "Monte-Carlo tree search"-PhD thesis, *Maastricht University*, (2010).

[28] S. Gelly, D. Silver, "Monte-Carlo tree search and rapid action value estimation in computer go" *Artificial Intelligence 175(11),* (2011).