Κωδικός Ομάδας: LAB20343534
Ονοματεπώνυμο 1: Περίδης Γιάννης
Ονοματεπώνυμο 2: Σκλάβος Παναγιώτης

1.Σκοπός των ασκήσεων:

Ο Σκοπός εκπόνησης των ασκήσεων θα μπορούσε να εξεταστεί τόσο μακροσκοπικά, ώς προς το σύνολο των ασκήσεων, όσο και διακριτά για κάθε άσκηση ξεχωριστά. Κύριως στόχος των ασκήσεων αποτέλεσε η πλήρης εξοικείωση με την γλώσσα περιγραφής υλικού VHDL,και η επίλυση σύνθετων σχεδιαστικών προβλημάτων χωρίς την χρήση των μεθόδων που μας παρέχονται απο λογική σχεδίαση(π.χ πίνακες Karnaugh κλπα.), συνεπώς κάθε εργαστήριο ήρθε να συμπληρώσει ορισμένες απαραίτητες γνώσεις ώστε να επιτευχθή κατανόηση σε βάθος των δυνατοτήτων που μας παρέχονται. Ειδικότερα παρατηρώντας σειριακά τις ασκήσεις, βλέπουμε ότι στην πρώτη στόχος ήταν να κατανοηθεί πώς χρησιμοποιώντας το εργαλείο Xilinx Ise με VHDL, μπορούμε να σχεδιάσουμε απλά κυκλώματα των οποίων την λειτουργία μπορούμε να ελέγξουμε μέσω του Προσωμοιωτή. Έπειτα, η δεύτερη άσκηση, σχεδιάστηκε με τέτοιο τρόπο ώστε να κατανοήσουμε και να πραγματοποιήσουμε ιεραρχικές δομές με πολλαπλά αρχείο(δημιουργία 4-bit adder με σχεδίαση των υπομονάδων του). Συνεχίζοντας στο τρίτο εργαστήριο, ήταν επιθυμητή η κατανόση και η εκπόνηση behavioural τρόπου σχεδίασης, με την δημιουργία και την περιγρφή της συμπεριφοράς ενός Strange Counter, όπου γνωρίσαμε και πώς να ελέχουμε συνθηκές στην VHDL. Συνεχίζοντας σε behavioural συλλογιστικές στο επόμενο εργαστήριο έιδαμε πώς με κώδικα της VHDL μπορούμε να σχεδιάσουμε FSM, καθορίζοντας τις εξόδους και τις μεταβάσεις της. Έτσι φτάνοντας στο τελευταίο εργαστήριο, σκοπός ήταν ο συνδυασμός όλων των παραπάνω γνώσεων, ώστε να πραγματοποιήσουμε μια συνδυαστική σχεδίαση, με την auto-generated μνήμη που μας παρέχει η VHDL, οπότε σε αυτήν την φάση κατανοήσαμε πως να υλοποιούμε συστήματα που κάνουν χρήση της μνήμης.

2.Περιγραφή τεχνολογίας που χρησιμοποιήθηκε:

Τα εργαστήρια του μαθήματος (μικροεπεξεργαστών και υλικού MHL),υλοποιήθηκαν με τεχνολογία FPGA.Χρησιμοποιήθηκε για αυτήν την υλοποίηση μια από τις εκδόσεις του εργαλείου της Xilinix ,το ISE.Μέσω αυτού έγινε η μοντελοποίηση των διαφόρων εργαστηριακών ασκήσεων(κυκλωμάτων) με την την γλώσσα περιγραφής υλικού VHDL (VHSIC-HDL, Very High Speed Integrated Circuit- Hardware Description Language) η οποία χρησιμοποιείται για τις διαδικασίες synthesis, simulation και specification.Αναλύοντας την λειτουργία των FPGAs, πρόκειται για προγραμματιζόμενα τσίπ, τα οποία περιέχουν, μια συλλογή διαμορφώσιμων λογικών μπλοκ που περιβάλλονται απ'ο μπλοκ εισόδου/εξόδου που συνδυάζονται μέσω προγραμματιζόμενων πόρων διασύνδεσης για να γίνουν οποιοδήποτε είδος ψηφιακού κυκλώματος ή συστήματος.Μέσω αυτών, για την υλοποίηση, ακολουθείται η παρακάτω διαδικασία σχεδιασμού: Specification->VHDL description(vhd files),αποτέλεσμα το functional simulation->Synthesis, αποτέλεσμα το post-synthesis simulation->Implementation(Mapping, Placing & Routing),αποτέλεσμα το timing simulation->Configuration και τέλος η δοκιμή σε τσίπ.Στα συγκερκιμένα εργαστήρια μας ζητήθηκε μόνο η μοντελοποίηση του κυκλώματος και η περιγραφή του σε VHDL και όχι οι υπόλοιπες διαδικασίες.Τέλος, θα πρέπει να σημειωθεί πως επειδή σε τεχνολογία FPGA το ρολόι φτάνει με διαφορετικό χρόνο σε κάθε κύκλωμα ,είναι προτιμότερο να μην πειραματιζόμαστε με το ρολόι (σε αντίθεση με την τεχνολογία VLSI που το επιτρέπει σε κάποιον βαθμό).

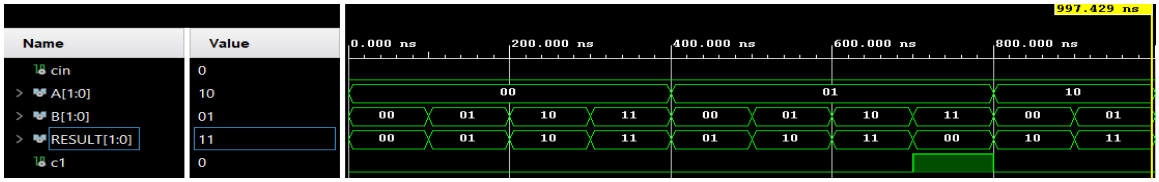
3. Περιγραφή της μεθοδολογίας επίλυσης των ασκήσεων:

Δύο βασικά μεθοδολογικά εργαλεία για την ψηφιακή σχεδίαση συστημάτων , είναι η αποσύνθεση και η κατάτμηση.Αναλύοντας, η αποσύνθεση είναι το «σπάσιμο» ενός πολύπλοκου συστήματος σε λειτουργικά κομμάτια, καθένα εκ των οποίων είναι μια ενότητα που μπορεί από μόνη της να αποδομηθεί περισσότερο με σταδιακή περαιτέρω αποσύνθεση,μέχρι να φτάσει στα βασικά στοιχεία που απαρτίζεται το σύστημα(«εξαρτήματα»).Ύστερα,η κατάτμηση ενός συστήματος είναι η απεικόνισή του στους φυσικούς πόρους υλοποίησής του.Στα συγκεκριμένα εργαστήρια,ακολουθήσαμε μια top-down σχεδίαση,δηλαδή εφαρμόστηκαν οι διαδικασίες της αποσύνθεσης και της κατάτμησης ,ιεραρχικά από πάνω προς τα κάτω.Αναλυτικότερα,ακολουθήθηκαν οι παρακάτω διαδικασίες με την σειρά αυτή:Απαιτούμενα(requirments)->Προδιαγραφές(specification)->Σχεδίαση(design)->Υλοποίηση(implementation)->Επιβεβαίωση λειτουργίας(validation).Στην σχεδίαση μας ακολουθήσαμε μια βασική θεωριτική σχεδιαστική ρόη(design flow) πάνω στην γλώσσα VHDL,με τα παρακάτω εξής βήματα(όπως έχει προαναφερθεί δεν μας ζητήθηκε η πραγματοποίηση όλων των βημάτων ,αλλά μέχρι πριν την σύνθεση):Συμπεριφεριακό μοντέλο(behavioral model)-> Προσομίωση (simulation)->Δομικό μοντέλο(structural)-> Προσομίωση (simulation)->Σύνθεση(synthesis)NETLIST->Προσομίωση(simulation)->Τοποθέτηση&Διασύνδεση(place&route)-> Προσομίωση (simulation)->Κατασκευή σε πλακέτα(download).Αναλύοντας το ζήτημα από μια πιο πρακτική όψη , η μεθοδολογία σε κάθε εργαστήριο ήταν να καταφέρουμε μόνο με την χρήση του block diagram που μας δινόταν για ένα κύκλωμα (ή με τις οδηγίες που μας δίνονταν να το κατασκευάζαμε μόνοι μας),να το απεικονίσουμε και αν το αναπαραστήσουμε σε γλώσσα VHDL(και φυσικά ισχύει και το ανάποδο).Δηλαδή,με την χρήση της μεθοδολογίας που αναφαίρθηκε παραπάνω,να φτάσουμε στην επίλυση της άσκησης χώρις να συμπεριλάβουμε , μεθοδολογίες και τεχνικές επίλυσης της λογικής σχεδίασης.Ακόμα πιο συγκεκριμένα βήματα για κάθε άσκηση ξεχωριστά, υπάρχει επεξήγηση της λειτουργίας των κυκλωμάτων στο κομμάτι των αποτελεσμάτων πριν από κάθε simulation.

4.Αποτελέσματα:

Παρακάτω φαίνονται τα αποτελέσματα των κυλωμάτων που σχεδιάσαμε: Να σημειωθεί ότι φαίνονται σε πλήρη μεγένθυση για να είναι ορατό το σύνολο της λειτουργικότητας, καθώς η εξέταση σε βάθος της λειτουργίας έγινε μέσω των εργαστηρίων.

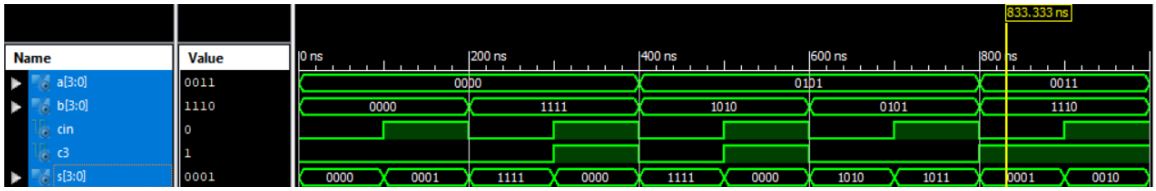
Lab1-Full adder



Εικόνα 4:simulation της λειτουργίας του full adder

Παρατηρούμε την λειτουργία του 2 bit full adder(στην εικόνα φαίνεται μόνο το topLevel testbench simulation των δύο συνδεδεμένων half-adder που τον υλοποιούν),όπυ a[1:0],b[1:0], οι δύο 2Bit τελεστέοι της εισόδου , το cin το κρατούμενοl εισόδου και το c1 το κρατούμενο εξόδου. Αξιοσημείωτο είναι να ελένξουμε την ορθή λειτουργία του αθροιστή μας στην περίπτωση την υπερχείλισης,δηλαδή όταν το κρατούμενο εξόδου είναι 1.

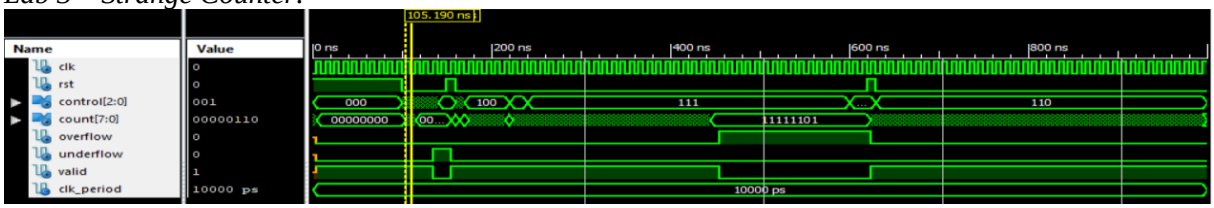
Lab 2 – Carry Look Ahead Adder:



Εικόνα 4.1: Simulation της λειτουργίας του CLA Adder.

Παρατηρούμε την λειτουργία του 4-bit CLA Adder(στην εικόνα φαίνεται το topLevel testbench simulation των τριών συνδεδεμένων μονάδων Propagate Generate unit,Carry Look Ahead unit και Sum unit) , όπου a[3:0], b[3:0] οι δύο 4bit τελεστέοι και το Cin το κρατούμενο εισόδου .Στην έξοδο παρουσιάζεται τα c3 το κρατούμενο εξόδου .Εδώ θα πρέπει να αναφερθεί πως αποτελεί και το critical path ,δηλαδή την χρονικά πιο αργή έξοδο του κυκλώματος.Τέλος ,το Sum που είναι το αποτέλεσμα του αθροιστή.Αξιοσημείωτο είναι να ελένξουμε την ορθή λειτουργία του αθροιστή μας στην περίπτωση την υπερχείλισης,δηλαδή όταν το κρατούμενο εξόδου είναι 1.

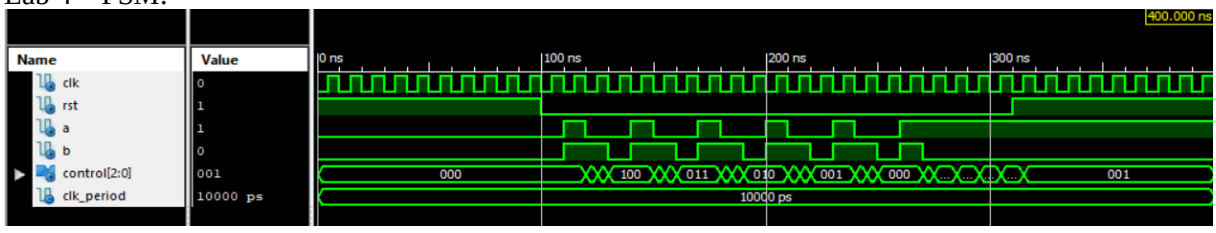
Lab 3 – Strange Counter:



Εικόνα 4.3: Simulation πλήρους λειτουργίας του Strange Counter.

Παρατηρούμε την λειτουργία του Strange counter που μετράει [0 - 255],όπου το control[3:0] είναι μια 3bit είσοδος που ανάλογα με την τιμή της αλλάζει το βήμα μέτρησης του μετρητή (βήμα μπορεί να αλλάζει σε κάθε κύκλο ρολογιού) και το cout[7:0] που είναι η 8Bit έξοδος του μετρητή.Οσόν αναφορά τις περιπτώσεις υπερχείλισης και υποχείλισης(στην περίπτωση που ξεπεραστεί το πάνω ή το κάτω άκρο αντίστοιχα) , τις διαχειριζόμαστε με τα δύο σήματα overflow ,underflow τα οποία όταν είναι ενεργά, παγώνουν το σύστημα στην τιμή που βρίσκεται και μπορούμε να το επαναφέρουμε στην αρχική μηδενική κατάσταση μόνο με την ενεργοποίηση του σήματος RST(το οποίο πρέπει να αναφερθεί πως δεν το χρησιμοποιούμε ποτέ σε συνδιασμό με κάποιο άλλο σήμα).Τέλος το σήμα valid , μας βοηθάει στην υλοποίηση της παρπάνω διαδικασίας , καθώς και είναι ένα σήμα που είναι ανενεργό μόνο όταν είναι ενεργό κάποιο από τα δύο σήματα overflow ή underflow.

Lab 4 – FSM:

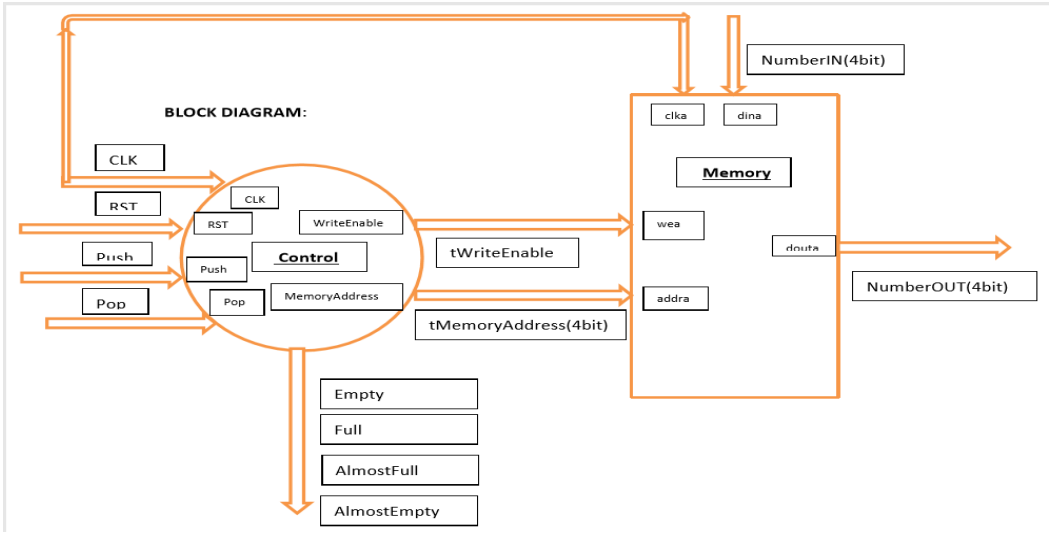


Εικόνα 4.4: Simulation πλήρους λειτουργίας της FSM

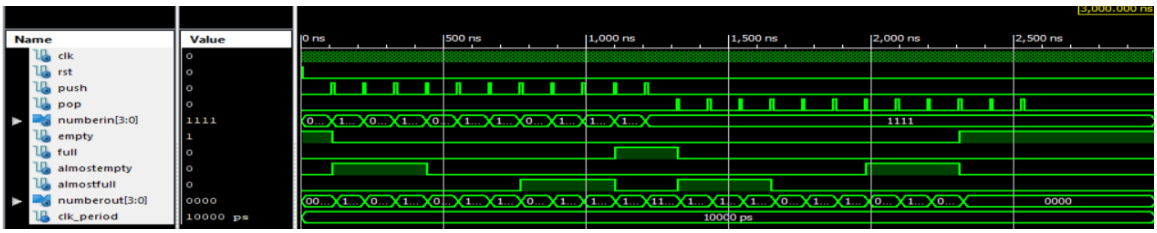
Παρατηρούμε την λειτουργία της mealy FSM(στην εικόνα φαίνονται στο testbench όλες

οι δυνατές περιπτώσεις μετάβασης από κάθε ακμή σε κάθε δυνατή κατάσταση),όπου τα a,b είναι οι είσοδοι από τις οποίες εξαρτώνται οι μεταβάσεις απο κατάσταση σε κατάσταση(είναι αναγκαίο να διευκρινίσουμε πως οι δυνατές περιπτώσεις των state είναι πέντε οπότε πρέπει να τις αναπαραστήσουμε με 3bit) , συγκερκιμένα με ab=10 , η μετάβαση γίνεται στην επόμενη κατάσταση,με ab=01 η μετάβαση γίνεται στην προιγούμενη κατάσταση, ενώ με ab=00,11 παραμένουμε στην ίδια κατάσταση.Το control[2:0] είναι η 3bit έξοδος του κυκλώματος η οποία όντας η FSM μας mealy, εξαρτάται όχι μόνο απο την τωρινή κατάσταση αλλά και από την είσοδο.Τέλος, το σήμα RST το οποίο όταν είναι ενεργό , μας επιστρέφει στην αρχική κατάσταση state 0.

Lab 5 - Stack:



4.5.1: Block Diagram Λειτουργίας του Stack.



4.5.2: Simulation Πλήρους λειτουργίας Stack

Παρατηρούμε την λειτουργία της lifo(last in first out) stack με αριθμό χωρητικότητας 10 στοιχείων και 4 bit το πλάτος του κάθε στοιχείου (στην εικόνα φαίνεται στο testbench διαδοχικά 12 push στην στοίβα και ύστερα 12 pop με 10 κύκλους ρολογιού αναμονή μετά από κάθε push και pop),όπου τα σήματα push και pop μας δείχνουν πότε εισάγουμε και πότε εξάγουμε στοιχείο από την στοίβα αντίστοιχα,το στοιχείο που εισάγουμε είναι το σήμα 4bit numberin[3:0] και το στοιχείο που εξάγουμε το numberout[3:0] 4bit.Τα σήματα almostempty και almostfull ,μας υποδεικνύουν πότε η στοίβα είναι σχεδόν άδεια(θέση 1-3) και σχεδόν γεμάτη(θέση 7-9) αντίστοιχα,ενώ τα σήματα empty και full μας υποδεικνύουν πότε η στοίβα είναι άδεια (θέση 0)άρα ακόμα και αν κάνουμε pop παραμένει στην ίδια κατάσταση και πότε είναι γεμάτη(θέση 10), άρα ακόμα και αν κάνουμε push παραμένει στην ίδια κατάσταση.Το οποίο το επαληθεύει το

tesbench ,εξού και ο λόγος που δώσαμε 12 διαδοχικά push και pop ενώ η στοίβα χωράει μόνο 10 στοιχεία.

5.Συμπεράσματα:

Από το σύνολο των εργαστηριακών ασκήσεων αποκομίσαμε ποικίλα συμπεράσματα. Αρχικά, λοιπόν, αν εξετάσουμε τα συμπεράσματα που εξήγαμε διακριτά από ασκήσεις, στο πρώτο και δεύτερο εργαστήριο συνειδητοποιήσαμε σε βάθος την έννοια της ιεραρχικής σχεδίασης, καθώς οργανώσαμε το τελικό μας κύκλωμα σε απλούστερα modules το οποία ώς σύνολο αργότερα έκαναν εφικτή την λειτουργία του κυκλώματος.Έπετια μία βασική παρατήρηση που πραγματοποιήσαμε από την εκπόνηση του 3^{ου} εργαστηρίου, είναι ότι είναι πολύ σημαντικό όταν ελέγχουμε συνθήκες στον κώδικα μας , να είμαστε ιδιαίτερα προσεκτικοί, οι if να είναι φωλιασμένες και όχι απλά να τοποθετούνται σειριακά, διότι όταν τοποθετούνται κατ' αυτόν τον τρόπο ελέγχονται όλες οι συνθήκες, οπότε μπορεί να έχουμε απροσδόκητα αποτελέματα. Προχωρόντας στο 4^{ο} εργαστήριο, συμπαιράναμε ότι σε κυκλώματα με ρολόι αναγκαίο είναι να εξασφαλίζουμε στον κώδκά, ότι θα φροντίζουμε οι μεταβολές να γίνονται στην θέση του ρολογιού που θα λειτουργόυσε και στην υλοποίηση του κυκλώματος(συνήθως θετικά ακμοπυροδότητο), καθώς δεν πρέπει να ξεχνάμε ότι η VHDL αποτελεί γλώσσα περιγραφής υλικού. Τέλος στο 5^{ο} εργαστήριο με την δημιουργία της του stack, συνειδητοποιήθηκε ότι είναι προτιμότερο να ακολουθουθείται pre-increment post-decrement σχεδίαση ,καθώς σε post-increment- pre decrement, θα χρειαζόταν η χρήση ενός επιπλέον pointer , κάνοντας περισσότερο σύνθετη, χωρίς λόγο, την υλοποίηση μας.

Από την άλλη συμπεράσμτα εξήγαμε και σε πιο γενικό βαθμό, από το σύνολο των εργαστηρίων .Συγκεριμένα κατανοήσαμε ότι πάντα όταν γράφουμε κώδικα είναι σημαντικό να τοποθετούμε σχόλια τα οποία βοηθούν τόσο εμάς στην αποσφαλμάτωση του κώδικα, όσο και τους υπολοιπους, για την κατανόησή του. Όσον αφορά τον τρόπο με τον οποία θα πρέπει να εργαζόμαστε έγινε κατανοητό, ότι είναι εξέχουσας σημασίας πρώτα να δημιουργόυμε το block diagram που περιγράφει την σχεδίαση που θέλουμε να πραγματοποιήσουμε και έπειτα να γράφουμε κώδικα για την υλοποίησή της, ώστε να μπορούμε να οργανώνουμε κατάλληλα την σκέψη μας.Διαφορετικά η επίλυση πιο σύνθετων σχεδιαστικών προβλημάτων θα ήταν αδύνατη. Συνοψίζοντας , αν κάτι έγινε προφανές με το πέρας των εργαστηριακών ασκήσεων, αυτό είναι ότι η VHDL και γενικότερα οι γλώσσες περιγραφής υλικού (Verilog κλπα.) αποτελούν εξαιρετικά εργαλεία τα οποία μπορούν να μας γλιτώσουν από πολύ κόπο, καθώς κάνουν πολύ πιο εύκολο το κομμάτι της σχεδίασης, και του ελέγχου της λειτουργίας του συστήματος που αναπτύσοουμε.