Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Πανεπιστήμιο Θεσσαλίας

ECE333 - Εργαστήριο Ψηφιακών Συστημάτων

Εργαστηριακή Εργασία 1η

Περιεχόμενα

1.Τίτλος

Όνομα: Τσελεπή Ελένη

ΑΕΜ:03272

Ημερομηνία:18/10-13/11

Τίτλος εργασίας: Εργαστηριακή Εργασία 1η, Οδηγός Ένδειξης 7-τμημάτων

2.Περίληψη

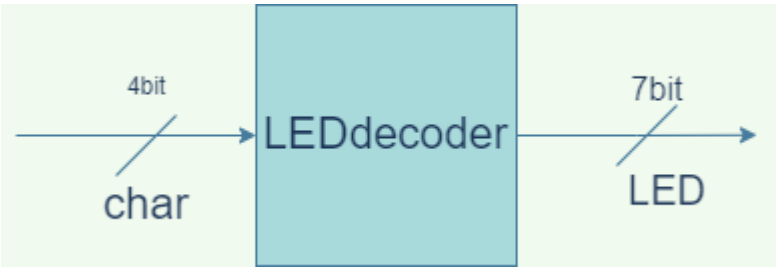
Το παρακάτω κείμενο αποτελεί την εργαστηριακή αναφορά για την 1η εργαστηριακή εργασία όπου στόχος της αναφοράς είναι η σαφής περιγραφή της διαδικασίας σχεδίασης, επαλήθευσης, δοκιμής και τελικής υλοποίησης του κυκλώματος. Παρακάτω αναφέρονται αναλυτικά όλα τα μέρη υλοποίησης της εργασίας καθώς και στιγμιότυπα από τις κυματομορφές στο vivado.

3.Εισαγωγή

Ο στόχος της 1ης εργαστηριακής εργασίας ήταν η υλοποίηση ενός οδηγού των τεσσάρων ενδείξεων 7-τμημάτων LED της πλακέτας Nexys A7-100T, για να επιτευχθεί η περιστροφική παρουσίαση του μηνύματος μεγέθους 16 χαρακτήρων *"0123456789abcdeF"* με το πάτημα ενός κουμπιού αλλά και μετά από ένα δεδομένο χρονικό διάστημα . Και οι δυο προαναφερθέντες στόχοι επιτευχθήκαν υλοποιώντας 2 διαφορετικά project στο vivado των οποίων τα βήματα που ακολουθήθηκαν θα αναλυθούν παρακάτω.

4. Μέρος Α - Υλοποίηση Αποκωδικοποιητή **7**-τμημάτων

Για την αποκωδικοποίηση των 7 τμημάτων υλοποίησα ένα συνδυαστικό αποκωδικοποιητή, ο οποίος έχει ως είσοδο το ψηφίο ή χαρακτήρα που θέλω να εμφανίσω, και παράγει ως έξοδο τις τιμές οδήγησης των 7 τμημάτων.

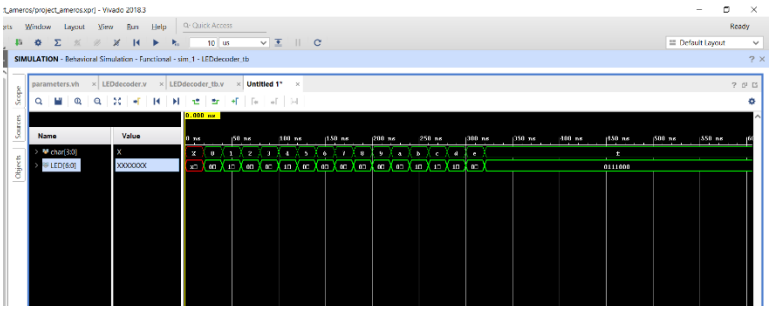


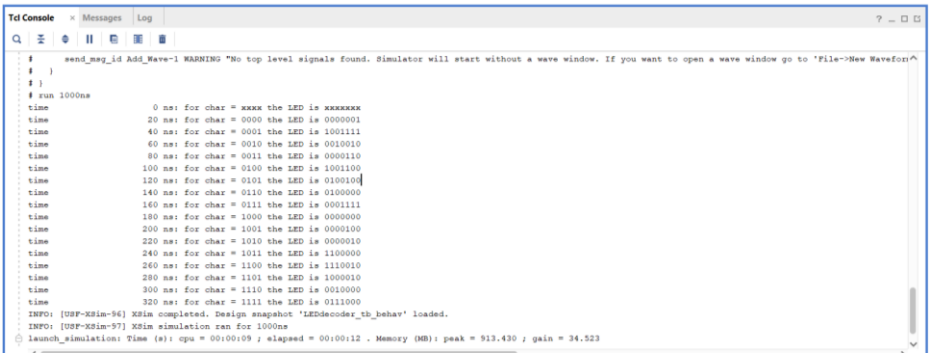
Διαβάζοντας το τεχνικό δελτίο της πλακέτας Nexys A7-100T όπου παρουσιάζεται αναλυτικά η συνδεσμολογία των τεσσάρων ενδείξεων 7-τμημάτων στην πλακέτα κατάλαβα πως για να ανάψει ένα συγκεκριμένο τμήμα LED ενός ψηφίου πρέπει το σχετικό σήμα καθόδου τμήματος, CA-G/DP, που επιλέγει το τμήμα, να οδηγείται στο μηδέν (0) και με βάση αυτό σχεδίασα την έξοδο του LEDdecoder. Πχ για να εμφανίσω στην πλακέτα τον αριθμό 1 χρειάζεται να φωτιστούν τα τμήματα b και c, έτσι η αποκωδικοποίηση είναι 7'b1001111 (Σελ. 23 τεχνικού δελτίου «*To illuminate a segment, the anode should be driven high while the cathode is driven low. However, since the Nexys A7 uses transistors to drive enough current into the common anode point, the anode enables are inverted. Therefore, both the AN0..7 and the CA..G/DP signals are driven low when active.»).*

Για την επαλήθευση της υλοποίησης στο testbench με σταθερή καθυστέρηση τα 20ns άλλαζα την είσοδο του LEDdecoder στις 16 πιθανές εισόδους και με την monitor έβλεπα την binary μορφή της εξόδου και το συνέκρινα με την υλοποίηση που έφτιαξα στο αρχείο LEDdecoder.

Το συγκεκριμένο μέρος δεν προγραμματίστηκε στην FPGA καθώς αποτελεί μια μονάδα της συνολικής εργασίας .

Τα αποτελέσματα από το testbench για το Μέρος Α .





5. Μέρος Β - Οδήγηση Τεσσάρων Ψηφίων

Για τη χρήση των τεσσάρων ψηφίων απαιτούνταν η εναλλάξ οδήγηση του κάθε ψηφίου με προτεινόμενη ελάχιστη καθυστέρηση φόρτισης τα 0.20μs και για να πετύχω την συγκεκριμένη καθυστέρηση χρησιμοποίησα μια δομική μονάδα MMCM (Mixed-Mode Clock Manager). Για τον προγραμματισμό της MMCM χρησιμοποίησα τους τύπους στη σελίδα 72 του τεχνικού δελτίου όπου

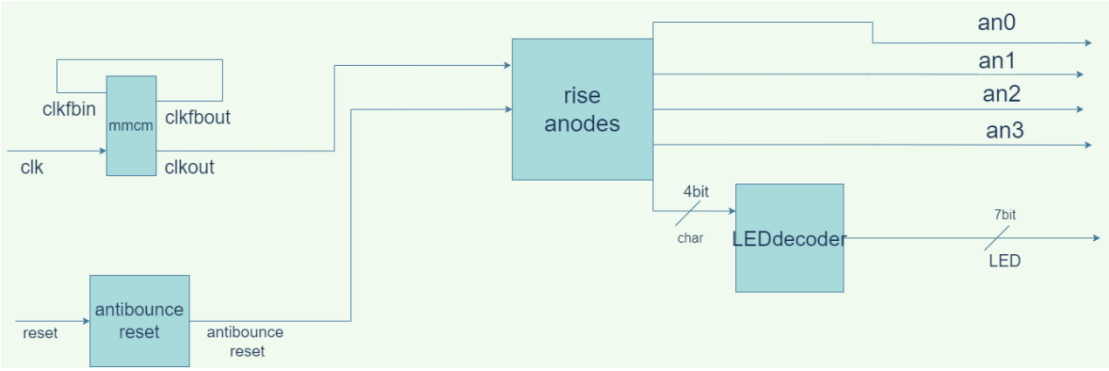
- FVCO=FCLKIN * M/D , 600MHZ <= FVC0 <= 1600MHZ , FCLKIN=100MHZ
- FOUT=FCLKIN * M/(D*O) ,FOUT=5HZ για περίοδο 200ns

Για την επιλογή των M,D,O υπήρχαν και περιορισμοί για τις τιμές που μπορούν να πάρουν συγκεκριμένα:

- M:CLKFBOUT_MULT_F -> 2-64
- D:DIVCLK_DIVIDE -> 1-106
- O: CLKOUT_DIVIDE -> 1-128

Με προτροπή των βοηθών του εργαστήριου να επιλέξουμε το M=6 και με βάση τους συγκεκριμένους τύπους κατέληξα πως D=1 και O=120 ώστε FVCO=600MHZ . Τις ίδιες τιμές για το FVCO=600MHZ θα έδιναν και M=12 , D=2 όμως με βάση το τεχνικό δελτίο *“The goal is to make D and M values as small as possible while keeping fvco as high as possible.”*

Επιπλέον για την σωστή λειτουργία του MMCM χρειαζόταν να συνδέσουμε το CLKFBIN στο CLKFBOUT , με βάση το τεχνικό δελτίο (*CLKFBIN must be connected either directly to the CLKFBOUT for internal feedback or IBUFG(through a clock-capable pin for external deskew).)*

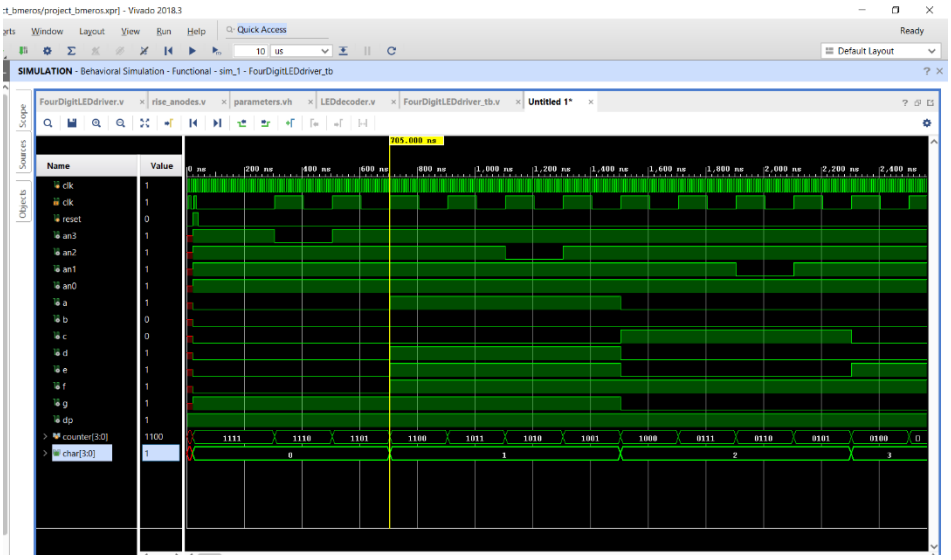


Εκτός από το ρολόι για το μέρος Β έφτιαξα 2 module το rise anodes και το antibounce reset . Το rise anodes οδηγεί τα σήματα των ανόδων τα οποία δε θα πρέπει να επικαλύπτονται όσο βρίσκονται στο μηδέν, αλλιώς θα αλλοιωθούν οι ενδείξεις. Έτσι, απαιτείται και ένα περιθώριο ασφαλείας από την επιστροφή ενός σήματος ανόδου στο λογικό ένα, μέχρι την πτώση του επόμενου. Γι αυτό τον λόγο οδηγώ τις ανόδους με την χρήση ενός counter . Επιπλέον για την οδήγηση των σημάτων των ενδείξεων χρησιμοποιούμε πάλι τιμές του μετρητή, έτσι ώστε να υπάρχει χρόνος προετοιμασίας (πρόθεσης - setup) μεταξύ των ανόδων και των δεδομένων των ενδείξεων. Συγκεκριμένα τα δεδομένα των ενδείξεων προετοιμάζονται 2 κύκλους ρολογιού πριν να ανάψει η άνοδος και παραμένουν και ένα κύκλο μετά την αλλαγή της ανόδου. Για το module antibounce θα αναφερθώ εκτενώς στο Μέρος Γ της εργασίας.

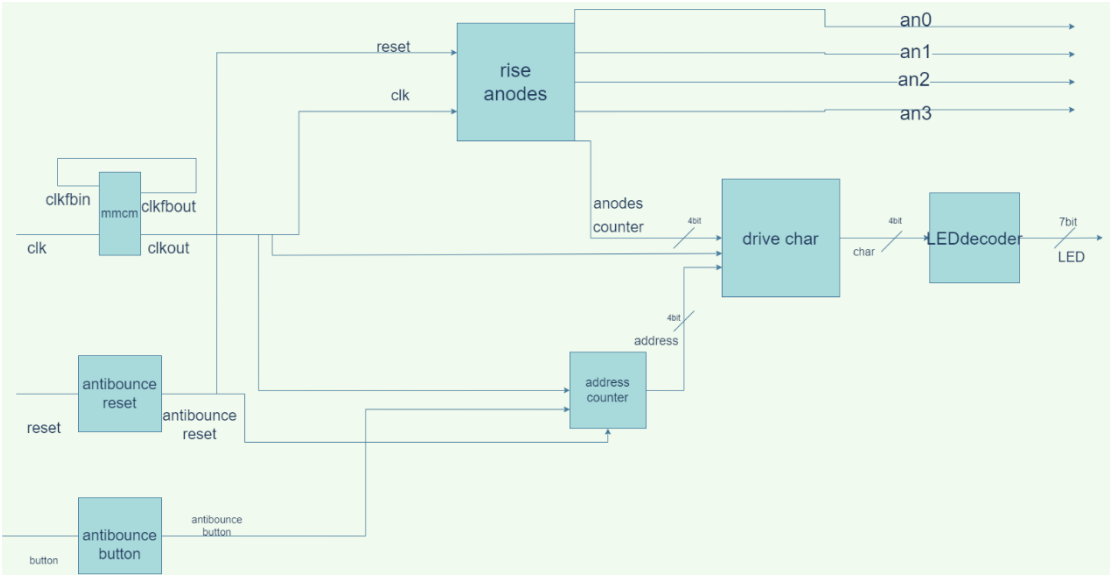
Για την επαλήθευση της υλοποίησης χρησιμοποίησα ένα αρχείο testbench στο οποίο φαίνεται πως όταν γίνεται reset ο counter αρχικοποιείται στην τιμή 1111 , στην τιμή 1110 η άνοδος 3 ενεργοποιείται και πέφτει στο 0 ενώ παρατηρούμε πως τα δεδομένα των ενδείξεων φορτώνονται 2 κύκλους ρολογιού πριν να ανάψει η άνοδος όπως συγκεκριμένα για την άνοδο 2 το char έχει πάρει τιμή από την τιμή 1100 του counter.

Όσον αφορά τον προγραμματισμό της πλακέτας χρειάστηκε μόνο ένας προγραμματισμός της πλακέτας καθώς με την πρώτη φορά τα ψηφιά εμφάνισαν το μήνυμα «0123» όπως το περίμενα με βάση τις τιμές που είχα δώσει για τα δεδομένα των ενδείξεων. Έτσι επιβεβαιώθηκε και πως το Α μέρος είχε γίνει σωστά .

<u>Στιγμιότυπο από το testbench για το μέρος Β</u>

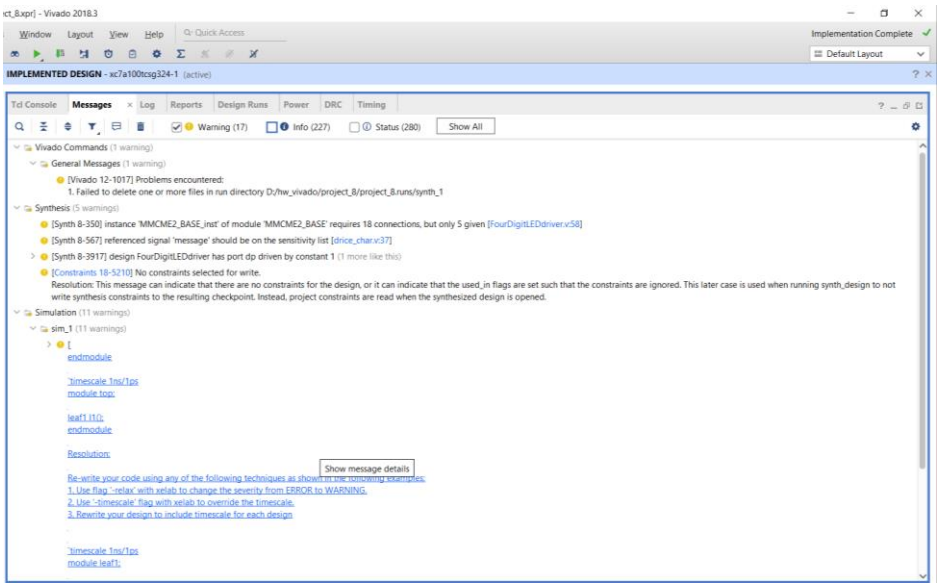


6.Μέρος Γ - Βηματική Περιστροφή του Μηνύματος με χρήση κουμπιού



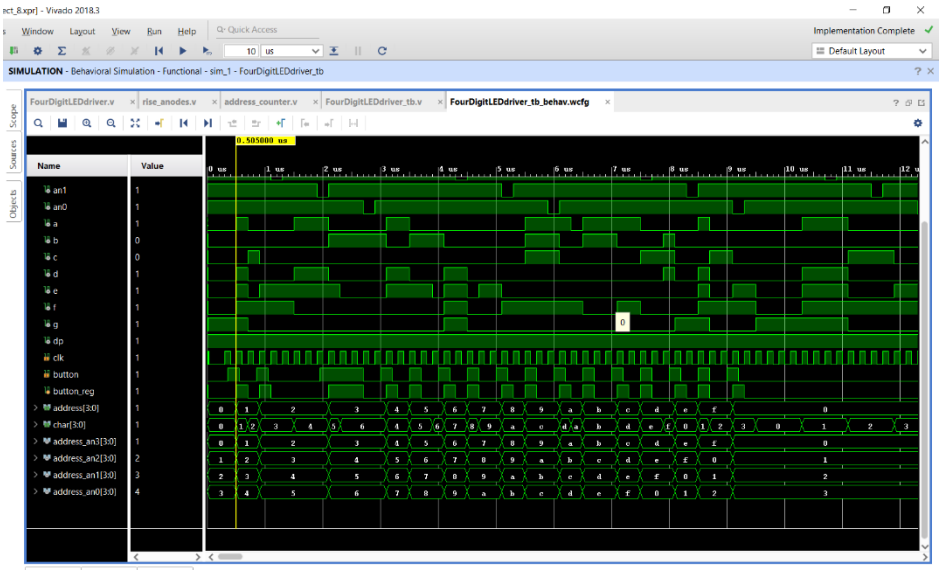
Για το Γ μέρος της εργασίας επέκτεινα την υλοποίηση που είχα ήδη πραγματοποιήσει στο Β μέρος . Συγκεκριμένα , έφτιαξα 2 καινούργια module ένα το address counter το οποίο έχει έναν counter ο οποίος δείχνει στην ενεργή περιοχή της μνήμης και το module drive char το οποίο περιέχει ότι έχει να κάνει με τον χαρακτήρα που θέλουμε

να οδηγήσουμε στα ψηφιά. Το module address counter έχει ως είσοδο το ρολόι , το reset μετά το antibounce και το σήμα από το κουμπί μετά το antibounce όπου συμφώνα με αυτό αυξάνεται ο counter κατά μια θέση. Το module drive char παίρνει ως είσοδο τους counter από το module rise anodes και address counter και το ρολόι για την αρχικοποίηση της μνήμης και έχει ως έξοδο τον χαρακτήρα τον οποίο θέλουμε να αποκωδικοποιήσουμε στο module LEDdecoder. Για την υλοποίηση του module address counter επειδή θέλω η διεύθυνση να αλλάζει με το πάτημα του κουμπιού το οποίο όμως πρέπει να συγχρονιστεί με το ρολόι πριν το always που αυξάνει τον counter της μνήμης έβαλα ένα Flip Flop για να κρατάω την προηγουμένη τιμή του σήματος του κουμπιού και να μπορώ να συγκρίνω αν έχει αλλάξει το σήμα και αν έχει γίνει ένα πρώτη φορά ώστε να αυξήσω κατά μια θέση τον μετρητή της μνήμης . Όσον αφορά το module drive char εκεί αρχικοποιώ την μνήμη και μετέπειτα με βάση τον counter του module drive anodes αλλάζω τα δεδομένα της εξόδου του module αναλόγως με το ποια άνοδος θα ανάψει. Επιπλέον στο συγκεκριμένο module με βάση τον μετρητή της μνήμης έχω 4 wires που κοιτάνε την μνήμη για κάθε άνοδο ξεχωριστά. Συγκεκριμένα αν η άνοδος 3 πρέπει να δείξει στην θέση 0 της μνήμης η άνοδος 2 πρέπει να δείξει στην θέση 0+1 , η άνοδος 1 στην θέση 0+2 και τέλος η άνοδος 0 στη θέση 0+3 . Για να αποφύγω να εξετάσω θέσεις που είναι εκτός της μνήμης αν το άθροισμα της μνήμης συν τις ανάλογες θέσεις που πρέπει να προστεθούν κάνει overflow τότε αφαιρώ τις 16 πιθανές καταστάσεις του μετρητή(4bit) αυτή την τιμή ώστε να γυρίσει πάλι από την αρχή της μνήμης .



Τα συγκεκριμένα warning δεν με προβλημάτισαν καθώς για το MMCM για να λειτουργήσει το ρολόι χρειαζόταν να συνδέσω μόνο τις πύλες που σύνδεσα , το dp το είχα στο 1 ώστε να είναι μονιμά σβηστό και επίσης το message επειδή είναι η μνήμη μου δεν γινόταν να μπει στο sensitivity list . Επιπλέον το timescale έβαλα μόνο στο αρχείο testbench γιατί δεν θεώρησα απαραίτητο να μπει και στα υπόλοιπα module.

Για την επαλήθευση της υλοποίησης χρησιμοποίησα ένα αρχείο testbench στο οποίο έδινα τιμές στο σήμα button για να μοντελοποιήσω το πάτημα του κουμπιού στην πλακέτα . Στόχος μου ήταν να εξετάσω όλες τις περιπτώσεις του κουμπιού δηλαδή είτε να το πατάω συνεχόμενα είτε να το πατήσω πολλές φορές ώστε να αλλάξει όλες τις τιμές ο μετρητής της μνήμης .



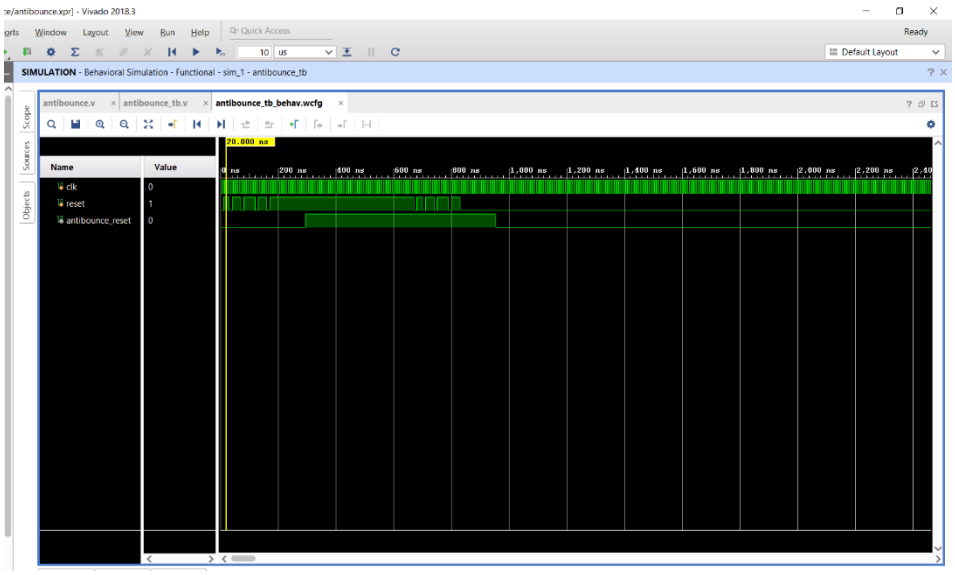
Στο συγκεκριμένο στιγμιότυπο φαίνεται πως αν άλλαζα πολλές τιμές στο button το address αλλάζει όλες τις θέσεις και επιστρέφει στο 0 αλλά και πως αν το button έμενε στο 1 για περισσοτέρους του ενός κύκλου ρολογιού τότε το address άλλαζε μόνο μια φορά .

Όσον αφορά τον προγραμματισμό της πλακέτας στην πρώτη προσπάθεια προγραμματισμού τα ψηφιά καθώς πατούσα το κουμπί δεν αλλάζαν μια θέση αλλά ήταν τυχαία η αλλαγή θέσεων . Αυτό συνέβαινε διότι δεν έλεγχα αν το σήμα button θα γίνει μια φόρα ένα αλλά κάθε φόρα που ήταν ένα άλλαζε μια θέση ο μετρητής της μνήμης με αποτέλεσμα να ήταν τυχαίο το πόσες θέσεις θα μετακινούνταν το μήνυμα ανάλογα με πόση ώρα είχα κρατημένο του κουμπί . Το πρόβλημα αυτό αντιμετωπίστηκε βάζοντας ένα Flip Flop το οποίο κρατάει την παλιά τιμή του κουμπιού και αν το σήμα αλλάξει και είναι στο ένα μόνο τότε αλλάζει μια θέση ο μετρητής της μνήμης . Μετά από μερικούς προγραμματισμούς το ζήτημα επιλύθηκε και λειτουργούσε κανονικά το μέρος Γ.

Module antibounce

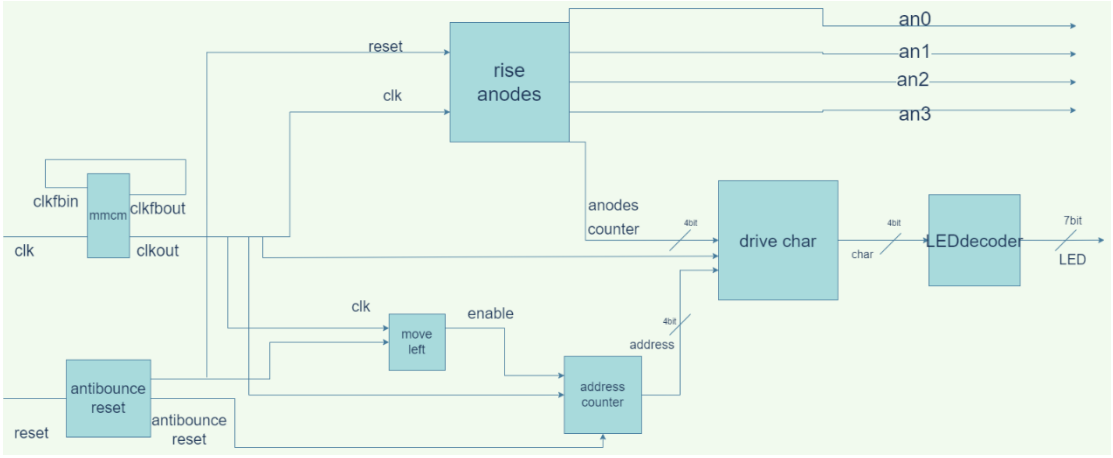
Το module antibounce χρησιμοποιήθηκε τόσο για το σήμα reset όσο και για το σήμα button που ερχόταν από την ακίδα M17. Το συγκεκριμένο module υπάρχει γιατί όταν πατάμε το κουμπί εκείνο πάλλεται για λίγο ανάμεσα στο 1 και στο 0 μέχρι να

σταθεροποιηθεί στο 1 κάτι το οποίο μας προκαλεί πρόβλημα στο κύκλωμα μας γι αυτό χρειάζεται να φιλτράρουμε αυτές τις ταλαντώσεις του σήματος . Για τον σκοπό αυτό χρησιμοποιούμε 2 Flip Flop (χρησιμοποιούμε 2 για να γλιτώσουμε το πρόβλημα της μεταστάθειας) και μετά χρησιμοποιούμε μια πύλη xnor για να συγκρίνουμε αν το αποτέλεσμα της εξόδου του 1^{ου} Flip Flop είναι ίδιο με την εξόδου του 2^{ου} Flip Flop. Αν το αποτέλεσμα είναι ίδιο τότε έχω έναν 22bit counter (2^22= 4.194.304 τιμές) και τον αυξάνω κατά μια τιμή. Όταν ο counter ξεπεράσει την τιμή που του έχουμε θέσει κάνει «reset» και επιστρέφει ξανά στο 0. Αν αυτός ο counter φτάσει σε μια συγκεκριμένη τιμή (2_500_000 κύκλους ρολογιού) τότε το σήμα βγαίνει στην έξοδο. Το 2_500_000 προήλθε καθώς για να θεωρήσουμε το σήμα σταθερό θα πρέπει να περάσει μισό δευτερόλεπτο= 500_000_000ns/200ns(η περίοδος του ρολογιού)=2_500_00 κύκλους ρολογιού. Το module antibounce το τέσταρα ξεχωριστά και δεν το συμπεριέλαβα στα testbench ούτε για το Μέρος Γ ούτε για το Μέρος Δ της εργασίας καθώς είχα ελέγξει ότι λειτουργούσε.



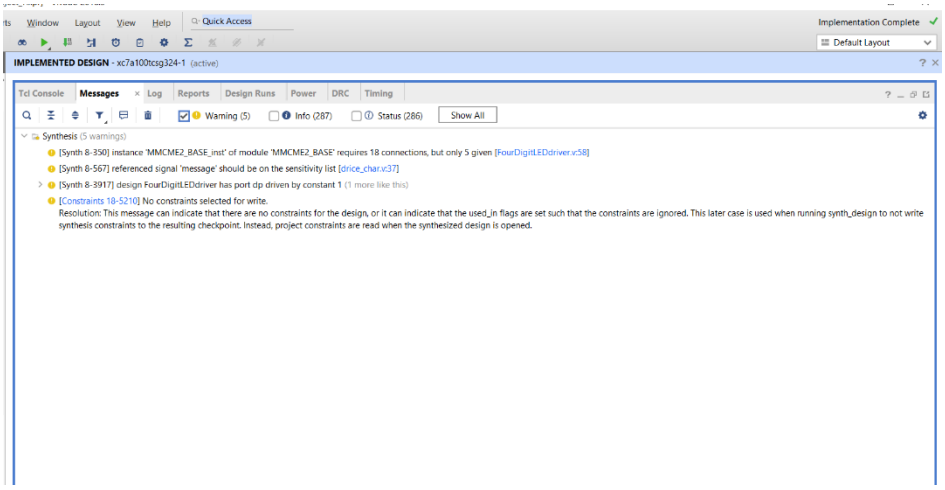
Το συγκεκριμένο στιγμιότυπο παρουσιάζει πως λειτουργεί το module antibounce. Όσο ταλαντώνεται το σήμα reset η έξοδος antibounce reset δεν γίνεται ένα . Όταν όμως περάσουν οι κύκλοι ρολογιού που έχουμε θέσει το σήμα γίνεται ένα και αλλάζει μέχρι να έρθει σταθερά σήμα 0 για να αποφύγουμε αν πατάμε συνεχόμενα το κουμπί το antibounce να το θεωρεί σαν πολλαπλά σήματα .

7.Μέρος Δ - Βηματική Περιστροφή του Μηνύματος με σταθερή καθυστέρηση



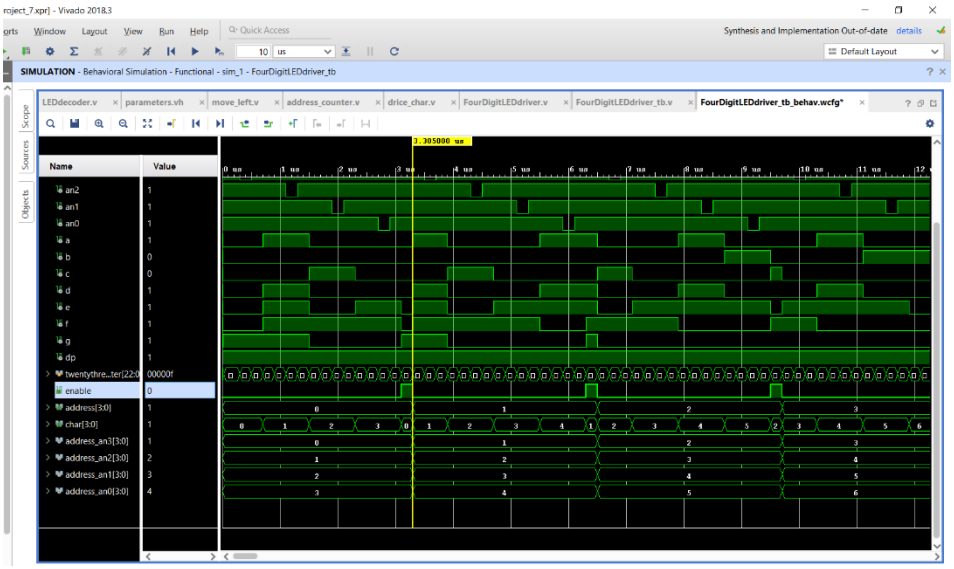
Το Δ μέρος της εργασίας μοιάζει πολύ με το Γ μέρος καθώς το μόνο που αλλάζει είναι το ότι το μήνυμα μας πλέον δεν αλλάζει με την χρήση κουμπιού αλλά περιστρέφεται προς τα αριστερά μετά από περίπου 1,67 δευτερόλεπτα. Για την υλοποίηση του πρόσθεσα μόνο το module move left το οποίο έχει έναν 23bit counter ο οποίος όταν μηδενίζεται σηκώνει ένα σήμα enable=1 το οποίο οδηγείται στο module address counter και μετακινεί την ενεργή θέση της μνήμης μια θέση παρακάτω.

Τα warning του implementation :



Τα συγκεκριμένα warning δεν με προβλημάτισαν καθώς για το MMCM για να λειτουργήσει το ρολόι χρειαζόταν να συνδέσω μόνο τις πύλες που σύνδεσα , το dp το είχα στο 1 ώστε να είναι μονιμά σβηστό και επίσης το message επειδή είναι η μνήμη μου δεν γινόταν να μπει στο sensitivity list .

Για την επαλήθευση της υλοποίησης χρησιμοποίησα ένα αρχείο testbench αλλά και άλλαξα τον counter του move left σε έναν 4bit counter ώστε να μπορώ να ελέγχω μέσω της προσομοίωσης αν αλλάζει το σήμα enable και μετέπειτα τις θέσεις στη μνήμη.



Όπως παρατηρούμε στο συγκεκριμένο στιγμιότυπο από το testbench την στιγμή που ο counter της καθυστέρησης πάρει την τιμή 0 το σήμα enable γίνεται 1 και στον επόμενο κύκλο ρολογιού (επειδή τα always blocks για τον counter της καθυστέρησης και για τον counter της μνήμης εκτελούνται ταυτόχρονα στον επόμενο κύκλο ρολογιού θα αυξηθεί ο counter της μνήμης address κατά μια θέση) αυξάνεται το address κατά μια θέση .

Όσον αφορά τον προγραμματισμό της πλακέτας στην πρώτη προσπάθεια προγραμματισμού το μήνυμα μου δεν μετακινούταν ανά μια θέση αριστερά αλλά ανά δύο δηλαδή μετά το 0123 εμφανιζόταν το μήνυμα 2345. Σε αυτό έφταιγε το γεγονός ότι δεν γινόταν κάλος συγχρονισμός της καθυστέρησης με την αλλαγή μηνύματος γιατί στο module address_counter στο sensitivity list του always block είχα γράψει always@(posedge enable) μετατρέποντας το δικό μου σήμα σε ρολόι κάτι το οποίο εισήγαγε καθυστερήσεις και δεν λειτουργούσε σωστά η περιστροφική κίνηση του μηνύματος. Το συγκεκριμένο λάθος λύθηκε βάζοντας στο sensitivity list το ρολόι και ελέγχοντας μέσα στο always την τιμής του σήματος enable. Το λάθος αυτό ήταν ένα λάθος που από το testbench αρχείο δεν μπορούσε να φανεί γιατί το σήμα enable δεν είχε καθυστερήσεις στο simulation που έκανα οπότε εγώ έβλεπα ότι όταν το enable ήταν ένα τότε το address άλλαζε μια θέση .

8. Συμπεράσματα

Η 1η εργαστηριακή εργασία πραγματοποιήθηκε με επιτυχία αφού όμως εξετάστηκαν σχολαστικά τα λάθη που προαναφέρθηκαν στο μέρος Γ και Δ . Χρειάστηκαν πολλές μέρες δουλείας τόσο στο επίπεδο της υλοποίησης αλλά και στο επίπεδο του προγραμματισμού της FPGA στις ώρες του εργαστήριου αλλά και στις εξτρά ώρες της Δευτέρας για να καταλάβω γιατί η πλακέτα δεν εμφάνιζε τα σωστά μηνύματα που έπρεπε να εμφανίζει.