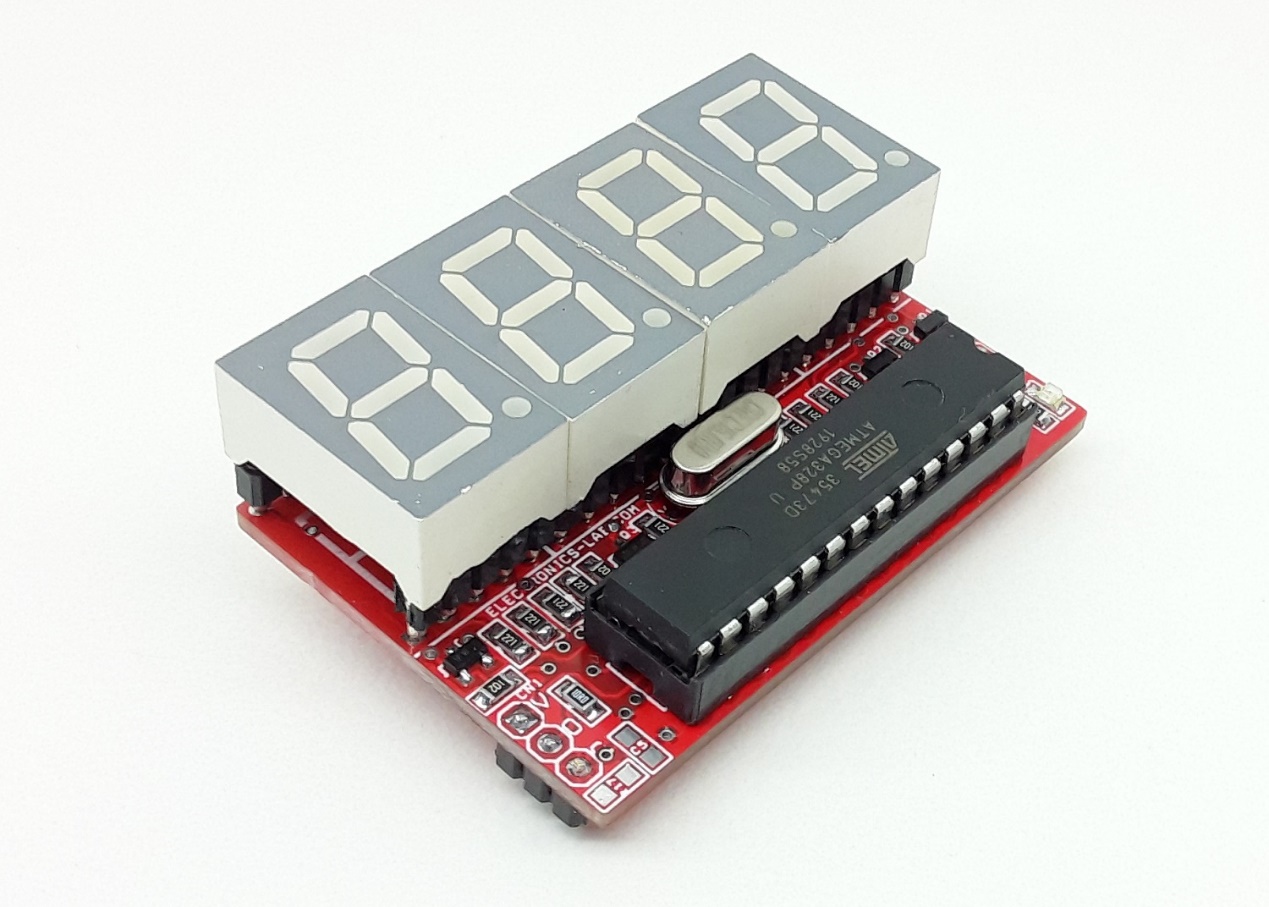
# 1η Εργαστηριακή Εργασία - Seven Segment Display



## Καπάκος Γεώργιος – 03165 19/11/2023

## Περίληψη

Με την χρήση Verilog και της πλακέτας Nexys A7-100T, προσομοιώνουμε έναν οδηγητή 4ων ψηφίων. Αρχικά, για το **Μέρος Α**, χρησιμοποιούμε έναν αποκωδικοποιητή για την αντιστοίχιση των τιμών των ψηφίων σε 4 bit αριθμούς. Ύστερα, για το **Μέρος Β** με την χρήση μιας MMCM μονάδας, μετατρέπουμε το γρήγορο ρολόι της εισόδου μας σε αργό. Μετά για την οδήγηση των ανόδων κάνουμε χρήση ενός περιστροφικού μετρητή, ο οποίος ενεργοποιεί σε one-cold κωδικοποίηση τις ανόδους. Αρχικοποιούμε το κύκλωμά μας με τη δημιουργία ενός anti-bounce, ώστε να αποφεύγονται οι αναπηδήσεις για το reset. Τοποθετούμε τους περιορισμούς, για την αντιστοίχιση των παλμών στις ακίδες της πλακέτας. Επιπρόσθετα, για το **Μέρος Γ** δημιουργούμε ένα κουμπί, με του οποίου το πάτημα, κάθε αριθμός στο seven segment display, θα παίρνει την επόμενη του τιμή. Χρησιμοποιώντας ένα κύκλωμα memory δίνουμε στον ανάλογο μετρητή την επόμενη τιμή που του αντιστοιχεί. Προσθέτουμε στους περιορισμούς του κυκλώματός μας, το κουμπί BTNR. Τέλος, για το **Μέρος** **Δ** φτιάχνουμε με έναν 23-bit counter, μία σταθερή καθυστέρηση 1,6777214 δευτερολέπτων, για την αλλαγή των ψηφίων μας στα επόμενά τους.

# Εισαγωγή

Ο στόχος της εργασίας είναι η υλοποίηση ενός οδηγού των 4ων ψηφίων 7-τμημάτων LED της πλακέτας Nexys A7-100T. Περιστρέφοντας ένα μήνυμα 16 χαρακτήρων. Το μήνυμα προς περιστροφή είναι είναι οι 16 χαρακτήρες των ψηφίων του δεκαεξαδικού συστήματος, δηλ. 0123456789AbCdEF. Το μήνυμα θα παρουσιαστεί μετατοπίζοντας τους χαρακτήρες, έναν προς έναν, κατάλληλα προς τα αριστερά, με:

• (1) με το πάτημα ενός κουμπιού(BTNR), από την πλακέτα.

• (2) μετά από ένα δεδομένο χρονικό διάστημα. Μετά τον τελευταίο χαρακτήρα του μηνύματος, θα ακολουθεί ο πρώτος, έτσι το μήνυμα ουσιαστικά θα περιστρέφεται διαρκώς. Εγκαθιστώντας μία σταθερή καθυστέρηση στο κύκλωμα για την αλλαγή των ψηφίων.

# Μέρος Α - Υλοποίηση Αποκωδικοποιητή 7-τμημάτων

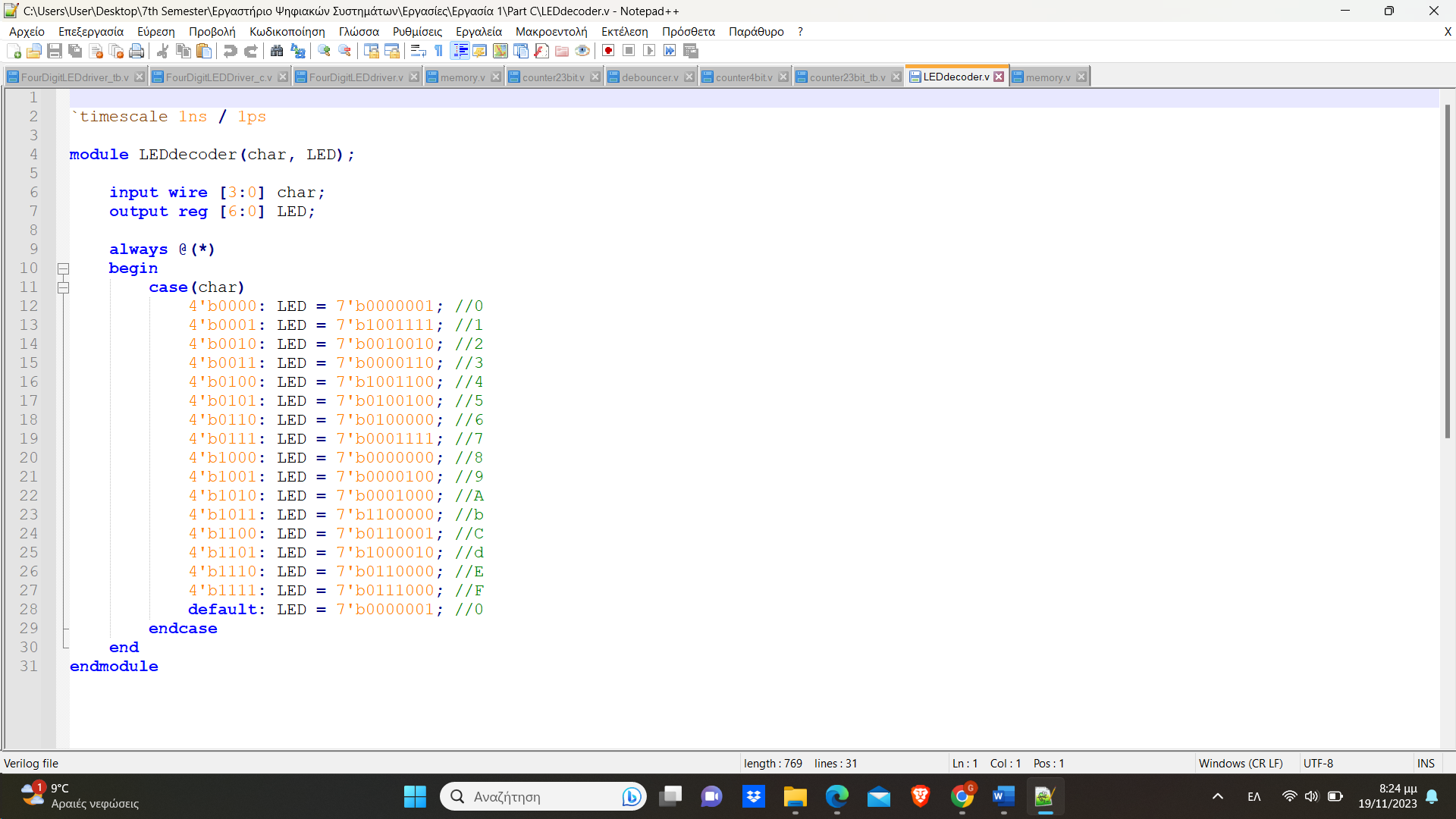
# Υλοποίηση:

Δημιουργήσαμε έναν αποκωδικοποιητή με 4 εισόδους και 7 εξόδους. Αντιστοιχίζει την κάθε είσοδο σε ένα ψηφίο, από τα: 0123456789abcdef.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, εικονίδιο υπολογιστή

Περιγραφή που δημιουργήθηκε αυτόματα

Το κύκλωμα μας αποτελείται από ένα always block, το οποίο με ένα case statement, ελέγχει την είσοδό μας και την αντιστοιχίζει σε μία συγκεκριμένη έξοδο.



# Επαλήθευση

Το πλαίσιο ελέγχου αποδείχτηκε απλό, δίνοντας μία πληθώρα τιμών στις εισόδους του αποκωδικοποιητή μας βλέπουμε πως έχουμε τα επιθυμητά αποτελέσματα. Χρησιμοποιήσαμε κάθε διαφορετική τιμή εισόδου για τον αποκωδικοποιητή και πήραμε τις ακόλουθες εξόδους, όπως περιγράφονται στην υλοποίηση.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, λογισμικό πολυμέσων

Περιγραφή που δημιουργήθηκε αυτόματα

# Πείραμα/Τελική Υλοποίηση:

Το μέρος Α δεν υλοποιήθηκε στην πλακέτα.

# Μέρος Β - Οδήγηση Τεσσάρων Ψηφίων

# Υλοποίηση:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, οθόνη, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματα

**MMCME module:**

Υλοποιήσαμε μία μονάδα **MMCME**, η οποία μετατρέπει το γρήγορο ρολόι σε αργό. Θέσαμε την τιμή **CLFBOUT\_MULT\_F**, σε 6, την τιμή **CLKIN1\_PERIOD** σε 10, την τιμή **CLKOUT1\_DIVIDE** σε 120 και την τιμή **DIVCLK\_DIVIDE** σε 1. Σύμφωνα με την σχέση που έχουμε στο manual μας, ισχύει ότι: Fout = Fclkin x M/(D x O), όπου Fout, η νέα συχνότητα με το αργό ρολόι, το οποίο έχει περίοδο 200ns, άρα η Fout = 5MHz. Fin, η συχνότητα με την οποία οδηγείται το γρήγορο ρολόι, το οποίο έχει περίοδο 10ns, άρα Fclkin = 100Mhz. Το Μ αντιστοιχίζεται στο CLFBOUT\_MULT\_F, το οποίο καθορίζει την ποσότητα με την οποία θα πολλαπλασιάσουμε όλα τα CLKOUT, δηλαδή τα output clocks, με τη συχνότητα που επιθυμούμε. Σε συνδυασμό με το DIVCLK\_DIVIDE, καθορίζει τη συχνότητα της εξόδου. Το D, αντιστοιχίζεται στο DIVCLK\_DIVIDE, το οποίο καθορίζει το μέγεθος για όλα τα ρολόγια των εξόδων, σε συνδυαστολή με το ρολόι της εισόδου, διαιρώντας αποτελεσματικά το CLKIN. Το Ο αντιστοιχίζεται στο CLKOUT\_DIVIDE, και στην δικιά μας περίπτωση στο CLKOUT1\_DIVIDE, το οποίο κάνει την ίδια δουλεία με το DIVCLK\_DIVIDE, αλλά μόνο για το συγκεκριμένο ρολόι. Το CLKIN1\_PERIOD, είναι η περίοδος του ρολογιού της εισόδου μας, το οποίο είναι 10ns.

Επίσης για την σύνδεση με την πλακέτα χρησιμοποιήσαμε ένα αρχείο με constraints, το οποίο συνδέει τις μεταβλητές μας στην βέριλογκ με τα pins της πλακέτας μας.

Εικόνα που περιέχει κείμενο, ηλεκτρονικές συσκευές, στιγμιότυπο οθόνης, οθόνη

Περιγραφή που δημιουργήθηκε αυτόματα

**MMCME instantiation:**

Το στιγμιότυπο του MMCME, στην έξοδο CLKOUT1, παίρνει την τιμή του νέου αργού ρολογιού του κυκλώματός μας. Βραχυκυκλώνουμε το CLKFBIN, με το CLKFBOUT, επειδή για την εξωτερική ευθυγράμμιση του ρολογιού, το feedback path clock buffer type, πρέπει να ταιριάζει με το forward clock buffer type. Παίρνει σαν είσοδο στο CLKIN1, το αρχικό γρήγορο clk και το reset στην αντίστοιχη είσοδο RESET.

**counter4bit module:**

Υλοποιήσαμε έναν 4 bit μετρητή, μετράει από το 0000->1111 και όταν φτάσει στην τιμή 0010, ενεργοποιεί την an0, όταν φτάσει στην τιμή 0110, ενεργοποιεί την an1, όταν φτάσει στην τιμή 1010, ενεργοποιεί την an2 και όταν φτάσει στην τιμή 1110, ενεργοποιεί την an3. Όπου an0, an1, an2, an3, είναι οι 4εις άνοδοι που οδηγούν τα ψηφία στο κύκλωμά μας.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, οθόνη, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματα

**debouncer module:**

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, διάγραμμα, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματα

Ο debouncer είναι ένα κύκλωμα, το οποίο αφού συγχρονίσει το σήμα μετράει μέχρι μία συγκεκριμένη περίοδο και τότε αφού θεωρήσει το σήμα σταθερό του δίνει την τιμή της εισόδου του. Ο debouncer αποτελείται από:

* 2 Flip Flops και μία πύλη xor, που χρησιμοποιούνται, για την σταθεροποίηση του σήματος.
* Έναν 8-bit μετρητή, που μετράει από ποιο σημείο και ύστερα θεωρούμε το σήμα μας σταθερό
* Ένα flip flop στην έξοδο, που μεταβιβάζει το σήμα μας στην έξοδο, μετά την περεταίρω σταθεροποίηση που υφίσταται, λόγω του flip flop D3.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, εικονίδιο υπολογιστή

Περιγραφή που δημιουργήθηκε αυτόματα

# Επαλήθευση

Χρησιμοποιούμε ένα απλό testbench, το οποίο ενεργοποιεί μία φορά το reset και ορίζει ένα ρολόι με περίοδο 10ns.

Παρατηρούμε πως το νέο αργό ρολόι έχει 20πλάσια περίοδο σε σχέση με το γρήγορο παλαιό ρολόι:

Εικόνα που περιέχει στιγμιότυπο οθόνης, κείμενο, οθόνη, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματα

Επίσης παρατηρούμε πως το an0, ενεργοποιείται έναν κύκλο μετά, αφού ο μετρητής πάρει την τιμή 0010, το an1, ενεργοποιείται έναν κύκλο μετά, αφού ο μετρητής πάρει την τιμή 0110, το an2, ενεργοποιείται έναν κύκλο μετά, αφού ο μετρητής πάρει την τιμή 1010 και το an3, ενεργοποιείται έναν κύκλο μετά, αφού ο μετρητής πάρει την τιμή 0111:

Εικόνα που περιέχει στιγμιότυπο οθόνης, κείμενο, λογισμικό, οθόνη

Περιγραφή που δημιουργήθηκε αυτόματα

Τέλος οι τιμές του FourDigitLEDdriver, του top level module μας, αφού δίνουμε στο char, σταθερή τιμή 1111, θα έχουν όλες την ίδια μορφή, για να αναπαραστήσουν το σύμβολο F, και στα τέσσερα ψηφία τα οποία οδηγούνται από τις ανόδους μας:

Εικόνα που περιέχει στιγμιότυπο οθόνης, κείμενο, λογισμικό, οθόνη

Περιγραφή που δημιουργήθηκε αυτόματα

# Πείραμα/Τελική Υλοποίηση:

Μετά την υλοποίηση του κυκλώματος στην πλακέτα παρατηρήσαμε ότι ανάβουν όλα τα ψηφία του seven segment display, με την ένδειξη F, όπως δηλαδή επιδεικνύεται και στο testbench, δεν αντιμετωπίσαμε κάποιο πρόβλημα στην υλοποίηση του κυκλώματος στην πλακέτα. Ένδειξη στην πλακέτα: FFFF

# Μέρος Γ - Βηματική Περιστροφή του Μηνύματος με χρήση Κουμπιού

# Υλοποίηση:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, οθόνη

Περιγραφή που δημιουργήθηκε αυτόματα

Σε αυτό το μέρος δημιουργήσαμε ένα κουμπί(button), το οποίο πατώντας το, τα ψηφία μας θα μεταβάλλονται στην επόμενη τιμή τους. π.χ από 0123 σε 1234. Στο κουμπί αυτό έχει προστεθεί ένα anti bounce module, για την αποφυγή των αναπηδήσεων, όταν το πατάμε για μικρό χρονικό διάστημα.

Επειδή βάζουμε στο κύκλωμα μας στο top level module μας το input button, χρειάζεται να το δηλώσουμε και ως constraint στο αρχείο με τα constraints:

Εικόνα που περιέχει κείμενο, ηλεκτρονικές συσκευές, στιγμιότυπο οθόνης, οθόνη

Περιγραφή που δημιουργήθηκε αυτόματα

**memory module:**

* Έχουμε ένα πίνακα από reg το message, το οποίο παίρνει όλες τις δυνατές τιμές από το 0000 στο 1111, και αρχικοποιούμε κάθε θέση του να έχει μία τιμή.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, οθόνη

Περιγραφή που δημιουργήθηκε αυτόματα

* Για να αποφύγουμε τις τιμές να αλλάζουν παραπάνω από μία φορές όσο έχουμε πατημένο το κουμπί, δημιουργούμε μία επιπλέον μεταβλητή που την ονομάζουμε button\_check, και ενεργοποιείται ένα κύκλο αφού έχουμε πατήσει το κουμπί μας. Τέλος εμείς δίνουμε στους μετρητές μας την επόμενη τιμή τους μόνο όταν είναι το button είναι 1 και το button\_check είναι 0, για να αλλάξουν οι τιμές μία φορά σε μία περίοδο ρολογιού.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, εικονίδιο υπολογιστή

Περιγραφή που δημιουργήθηκε αυτόματα

* Κάθε φορά που μία άνοδος είναι ανοιχτή θέλω λίγο νωρίτερα το char, που αποτελεί την έξοδο μας, δηλαδή το ψηφίο που αναπαρίσταται στο seven segment display, να λαμβάνει την τιμή του μηνύματος στην διεύθυνση της συγκεκριμένης ανόδου, όταν όμως καμία άνοδος δεν είναι ανοιχτή τότε, το char θα λάβει την τιμή F, η οποία είναι αδιάφορη επειδή ποτέ δεν θα αναπαρασταθεί στην πλακέτα μας. Αυτό, πραγματοποιείται δημιουργώντας μέσα ένα case statement, με case την τιμή ενός 4 bit counter. Ανάλογα με την τιμή που παίρνει ο counter, να αλλάζει και η τιμή του char. Από τις τιμές:
* counter: 0000->0011, το char, δέχεται τις τιμές με βάση την τιμή του count3
* counter: 0100->0111, το char, δέχεται τις τιμές με βάση την τιμή του count2
* counter: 1000->1011, το char, δέχεται τις τιμές με βάση την τιμή του count1
* counter: 1100->1111, το char, δέχεται τις τιμές με βάση την τιμή του count0

Όταν char =

* 1111->0010, τότε ενεργοποιείται η άνοδος 0
* 0011->0110, τότε ενεργοποιείται η άνοδος 1
* 0111->1010, τότε ενεργοποιείται η άνοδος 2
* 1011->1110, τότε ενεργοποιείται η άνοδος 3

Εικόνα που περιέχει κείμενο, ηλεκτρονικές συσκευές, στιγμιότυπο οθόνης, οθόνη

Περιγραφή που δημιουργήθηκε αυτόματα

# Επαλήθευση

Τα διανύσματα που χρησιμοποιήσαμε για την λειτουργία του testbench είναι το clk, το reset και το button, στα οποία αλλάζουμε τιμές αρκετές φορές, για να ελέγξουμε και την λειτουργία του debouncer, ανάλογα με την χρονική περίοδο που παραμένουν ανοιχτά. Επίσης καλύπτουμε την αλλαγή των τιμών του κυκλώματός μας, με το πάτημα του κουμπιού, άρα καλύπτεται, κατά 100%, η λειτουργία του κουμπιού.

Αρχικά η λειτουργία του **debouncer**, φαίνεται στο reset και στο button, από την παρακάτω κυματομορφή, όπου αφού ο counter φτάσει στο σημείο όπου ισούται με 255, δηλαδή 2^8-1, τότε ενεργοποιεί το σήμα του new\_reset, και του new\_button, μέχρι να πέσει το σήμα πάλι, καθώς παρατηρούμε πως εξαλείφονται και τα διάφορα bounces, που μπορεί να έχουν τα κουμπιά, αφού αποκόβονται εντελώς, από τις τιμές του new\_button και του new\_reset:

Εικόνα που περιέχει στιγμιότυπο οθόνης, κείμενο, λογισμικό, υπολογιστής

Περιγραφή που δημιουργήθηκε αυτόματα

Ύστερα, παρατηρούμε πως με το πάτημα του κουμπιού αλλάζουν οι τιμές που έχει το κύκλωμα, η an0, αρχίζει από την τιμή 3, η an1 από την τιμή 2, an2 από την τιμή 1 και η an3, από την τιμή 0 και με το πάτημα του κουμπιού, όλες αλλάζουν.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, οθόνη, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματα

Τα σχεδιαστικά προβήματα που αντιμετωπίσαμε στην υλοποίηση ήταν ότι αρχικά δημιουργήσαμε ένα κύκλωμα το οποίο, άλλαζε την τιμή του char, μετά την ενεργοποίηση της ανόδου, οπότε δεν άφηνε χρόνο στην άνοδο να αλλάξει τιμή έτσι ώστε να μπορέσει να προσαρμοστεί, κατά την λήψη μιας τιμής.

Εικόνα που περιέχει κείμενο, ηλεκτρονικές συσκευές, στιγμιότυπο οθόνης, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματα

Αυτό το πρόβλημα το επιλύσαμε, αλλάζοντας την δομή του προγράμματός μας, δίνοντας τιμές στις ανόδους νωρίτερα, για να έχουν χρόνο να φορτώσουν την τιμή στο ψηφίο και για να μπορούν με το πάτημα του κουμπιού να αλλάξουν εγκαίρως τιμή, όπως φαίνεται και πάνω στην περιγραφή του memory module.

# Πείραμα/Τελική Υλοποίηση:

Χρειάστηκε να δοκιμαστεί η λειτουργία του προγράμματος μερικές φορές μέχρι να κατανοήσουμε το λειτουργικό λάθος, από την λανθασμένη κατανόηση, της εκφώνησης του Γ Μέρους. Δεν αντιμετωπίσαμε τυχόν ηλεκτρικά προβλήματα, με την σύνδεση της πλακέτας, όλα κύλησαν ομαλά, καθώς και η μικρή περιπλοκότητα του κυκλώματος, δεν μας επέφερε δυσκολίες στην δοκιμή του στην πλακέτα. Εντοπίσαμε τα pins, όπου τοποθετήσαμε το κουμπί και το reset. Ο κώδικας, με τον προγραμματισμό της πλακέτας FPGA, με την άνω λάθος υλοποίηση της επαλήθευσης και η πλακέτα έφερε αποτελέσματα τα οποία ήταν ανεπιθύμητα. Αφού αλλάξαμε την λειτουργικότητα του κυκλώματος πάνω η πλακέτα άρχιζε να λειτουργεί κανονικά, εμφανίζοντας τα ψηφία σωστά και τα επαύξανε με το πάτημα του κουμπιού. Το reset λειτουργούσε με ακρίβεια αναιρώντας τα bits και επαναφέροντάς τα στην αρχική τους κατάσταση, χωρίς κάποιο πρόβλημα με τις αναπηδήσεις.

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

# Υλοποίηση:

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, λογισμικό, οθόνη

Περιγραφή που δημιουργήθηκε αυτόματα

Το κύκλωμα προς υλοποίηση ήταν ένα κύκλωμα στο οποίο τα ψηφία άλλαζαν με μία σταθερή βηματική καθυστέρηση 1.6777 sec, στο επόμενό τους ψηφίο. Δηλαδή από 0123->1234., κ.ο.κ.

**counter23bit module:**

Την καθυστέρηση την συγκεκριμένη την εγκαθιστήσαμε με ένα 23 bit counter, ο οποίος μετράει από το 0 έως το (2^23-1), όταν γίνουν όλα του τα bit 1, τότε εμείς τον μηδενίζουμε και δίνουμε το σήμα en(enable), το οποίο αλλάζει τα ψηφία στην επόμενή τους κατάσταση.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, οθόνη, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματα

**memory module:**

Έχει παρόμοια δομή με το μέρος Γ, εάν εξαιρέσουμε το γεγονός ότι για να δώσουμε την επόμενη τιμή στην έξοδό μας char, έτσι ώστε να αλλάξουν τα digits, απλά χρειαζόμαστε το σήμα en, από τον 23 bit counter module, που αναφέρουμε παραπάνω, το οποίο ενεργοποιείται για έναν κύκλο ρολογιού, όπου εκεί γίνονται οι αλλαγές.

Εικόνα που περιέχει κείμενο, στιγμιότυπο οθόνης, οθόνη, λογισμικό

Περιγραφή που δημιουργήθηκε αυτόματα

# Επαλήθευση:

Το πλαίσιο του testbench αποτελείται από αλλαγές στο reset, για τον περεταίρω έλεγχο του debouncer, που έχουμε βάλει στο reset και τον ορισμό του γρήγορου ρολογιού στα 10ns. Δεν απαιτείται η δήλωση επιπλέον μεταβλητών στο κύκλωμα καθώς αρκούν αυτά για να δούμε όλες τις μεταβολές που μπορούν να συμβούν.

Είναι αδύνατο να φανεί στην προσομοίωση η αλλαγή ανά 1.777sec, οπότε μειώνοντας αποκλειστικά για την προσομοίωση τον μετρητή μας σε 8 bit counter:

Εικόνα που περιέχει στιγμιότυπο οθόνης, λογισμικό, κείμενο, οθόνη

Περιγραφή που δημιουργήθηκε αυτόματα

Βλέπουμε ότι το σήμα en αλλάζει στον επόμενο κύκλο ρολογιού, από το τελευταίο στοιχείο του counter μας που εδώ τοπικά στην προσομοίωση είναι το 255(2^8-1). Μετά την ενεργοποίηση του en, παρατηρούμε πως και τα ψηφία στις ανόδους αλλάζουν τιμή, οπότε από το 0123, οδηγούμαστε στο 1234.

# Πείραμα/Τελική Υλοποίηση:

Δοκιμάσαμε να προγραμματίσουμε την FPGA, αρχικά έχοντας διαφορετικό κώδικα στον 23 bit counter, ο οποίος ενεργοποιούταν αν ο counter ήταν μεγαλύτερος από το (2^23-1)-16, το οποίο στην πλακέτα φαινόταν να μην άλλαζε ποτέ τιμή και παρέμενε σταθερό στην αρχική του κατάσταση, δηλαδή στο 0123. Μετά την αλλαγή του παρακάτω κώδικα με τον παραπάνω counter23bit το πρόγραμμα έδειχνε κανονικά την μεταβολή των ψηφίων, χωρίς κάποιο πρόβλημα.

Εικόνα που περιέχει κείμενο, ηλεκτρονικές συσκευές, στιγμιότυπο οθόνης, οθόνη

Περιγραφή που δημιουργήθηκε αυτόματα

# Συμπεράσματα

Σύνοψη της πορεία της εργασίας στο εργαστήριο στους τομείς σχεδίασης, επαλήθευσης, δοκιμής και τις συνολικές προκλήσεις και δυσκολίες που αντιμετωπίστηκαν και πως αυτές αντιμετωπίστηκαν.

Η εργαστηριακή πορεία μας άρχισε με την σχεδίαση και επαλήθευση του Μέρους Α και Μέρους Β, τα οποία όντας εύκολα ερωτήματα, χωρίς πολλές δυσκολίες αντιμετωπίστηκαν. Βέβαια το Μέρος Β, είχε την δυσκολία να ανατρέξεις στο manual, για να κατανοήσεις τις συνδέσεις του MMCME module και τις παραμέτρους που πρέπει να έχει, για να μπορέσεις να βρεις την κατάλληλη περίοδο για το νέο ρολόι και να δημιουργήσεις ένα κατάλληλο βραχυκύκλωμα με το ρολόι εισόδου του buffer και το ρολόι εξόδου του. Ύστερα πέρα από τις σχεδιαστικές δυσκολίες, η προσομοίωση του και η εφαρμογή του στην πλακέτα δεν αποτέλεσε πρόβλημα. Ύστερα το Μέρος Γ, παρουσίασε σχεδιαστικές προκλήσεις, τις οποίες καταλάβαμε πως υπήρχαν μετά την εφαρμογή του στην πλακέτα, οπότε και αναθεωρήσαμε τον κώδικα του module memory, για τη σωστή λειτουργία του κυκλώματος. Τέλος το Μέρος Δ, εμφάνισε δυσκολίες στον εντοπισμό του προβλήματος, για την αλλαγή των bits, τα οποία παρέμεναν σταθερά, χωρίς να μεταβάλλονται, το εντοπίσαμε και αυτό το πρόβλημα στην πλακέτα, όπου μετά από αλλαγές στον κώδικα λειτουργούσε κανονικά χωρίς προβλήματα.