ECE333 - Lab 1

Νίκας Ιωάννης Ιάσων

24/10/2024

Περιεχόμενα

1 Περίληψη **1**

2 Εισαγωγή **2**

3 Μέρος Α - Υλοποίηση αποκωδικοποιητή 7-τμημάτων **2**
3.1 Υλοποίηση . . . 2
3.2 Επαλύθευση . . . 2

4 Μέρος Β - Οδήγηση τεσσάρων ψηφίων **3**
4.1 Υλοποίηση . . . 3
4.1.1 ResetDebouncer . . . 4
4.1.2 Counter . . . 4
4.1.3 AnodeDecoder και CharacterDecoder . . . 5
4.1.4 Top level module . . . 5
4.2 Επαλήθευση . . . 6
4.3 Πείραμα . . . 7

5 Μέρος Γ - Βηματική περιστροφή του μηνύματος με χρήση κουμπιού **7**
5.1 Υλοποίηση . . . 7
5.1.1 ButtonCounter . . . 7
5.1.2 CharacterMemory . . . 8
5.1.3 InputDebouncer . . . 8
5.1.4 Top level module . . . 8
5.2 Επαλήθευση . . . 9
5.3 Πείραμα . . . 10

6 Μέρος Δ - Βηματική περιστροφή του μηνύματος με σταθερή καθυστέρηση **10**
6.1 Υλοποίηση . . . 10
6.2 Επαλήθευση . . . 11
6.3 Πείραμα . . . 12

1 Περίληψη

Στην αναφορά αυτή παρουσιάζονται ξεχωριστά το κάθε μέρος της 1ης εργασίας. Αναλύεται η λειτουργία του κάθε module ξεχωριστά και πως γίνεται η σύνδεση ανάμεσα τους. Επιπλέον εξηγήται ο τρόπος που έγιναν οι προσομοιώσεις και ο έλεγχος της ορθής λειτουργίας πριν την πραγματική δοκιμή τους στην πλακέτα. Αναλύονται προβλήματα που παρατηρήθηκαν τόσο στο επίπεδο προσομοίωσης όσο και στο πρακτικό επίπεδο δοκιμής.

2 Εισαγωγή

Ο στόχος της συγκεκριμένης εργασίας ήταν η σωστή οδήγηση των ανόδων και καθόδων των seven segment displays στην πλακέτα ώστε να μπορέσουμε να εμφανίσουμε ένα σταθερό μήνυμα και κινούμενο μήνυμα σε αυτήν. Αυτό πραγματοποιήθηκε με την χρήση ενός counter ο οποίος είχε συγχρονιστικό ρόλο ώστε να μπορούμε να συγχρονίσουμε την οδήγηση των καθόδων και τον ανόδων των displays ώστε να έχουμε το επιθυμητό αποτέλεσμα.

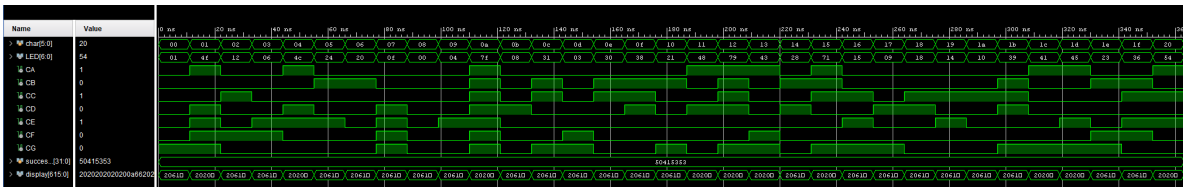
3 Μέρος Α - Υλοποίηση αποκωδικοποιητή 7-τμημάτων

3.1 Υλοποίηση

Η υλοποίηση στο συγκεκριμένο μέρος ήταν σχετικά απλή καθώς απαιτείται ένας απλός αποκωδικοποιήτης ο οποίος μετατρέπει έναν χαρακτήρα στα αντίστοιχα σήματα καθόδου `CA-G`. Δηλαδή χρειάζεται μόνο ένα module συνδυαστικής λογικής. Χρησιμοποιήθηκαν 6 bits για τον χαρακτήρα ώστε να μπορεί να αναπαρασταθεί όλο το αγγλικό αλφάβητο, τα ψηφία `0-9` και το κενό.

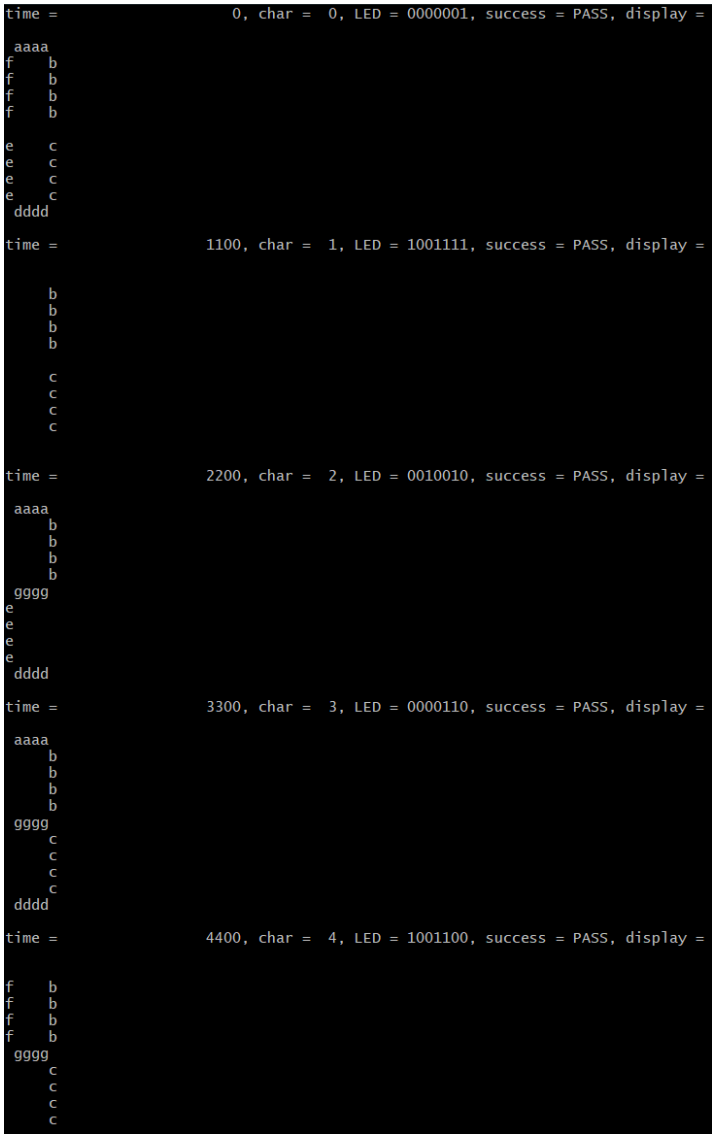
3.2 Επαλύθευση

Για την επαλύθευση του κυκλώματος χρησιμοποιήθηκε ένα testbench το οποίο ελέγχει τα σήματα εξόδου της μονάδας του αποκωδικοποιήτη για διαφορετικές εισόδους. Ελέγχει κάθε πιθανό χαρακτήρα που υποστηρίζεται από τον αποκωδικοποιήτη και τυπώνει ένα μήνυμα PASS ή FAIL ανάλογα με την έξοδο του αποκωδικοποιητή.



Σχήμα 1: LEDdecoder simulation

Επιπλέον για την ορθή επαλήθευση της αντιστοίχησης των χαρακτήρων χρησιμοποιήθηκε ακόμα ένα μήνυμα το οποίο απεικονίζει τον χαρακτήρα όπως θα φαινόταν πραγματικά σε ένα seven segment display. Έτσι μπορούμε πραγματικά να διαπιστώσουμε αν ο χαρακτήρας που δίνουμε σαν είσοδο στον αποκωδικοποιητή δίνει σωστή έξοδο στις καθόδους των LEDs. Με την χρήση αυτού του testbench διαπιστώθηκαν κάποιες λανθασμένες αντιστοιχίσεις (typos στα bits του αποκωδικοποιήτη) και διορθώθηκαν.



Σχήμα 2: Ενδεικτική έξοδος του testbench

4 Μέρος Β - Οδήγηση τεσσάρων ψηφίων

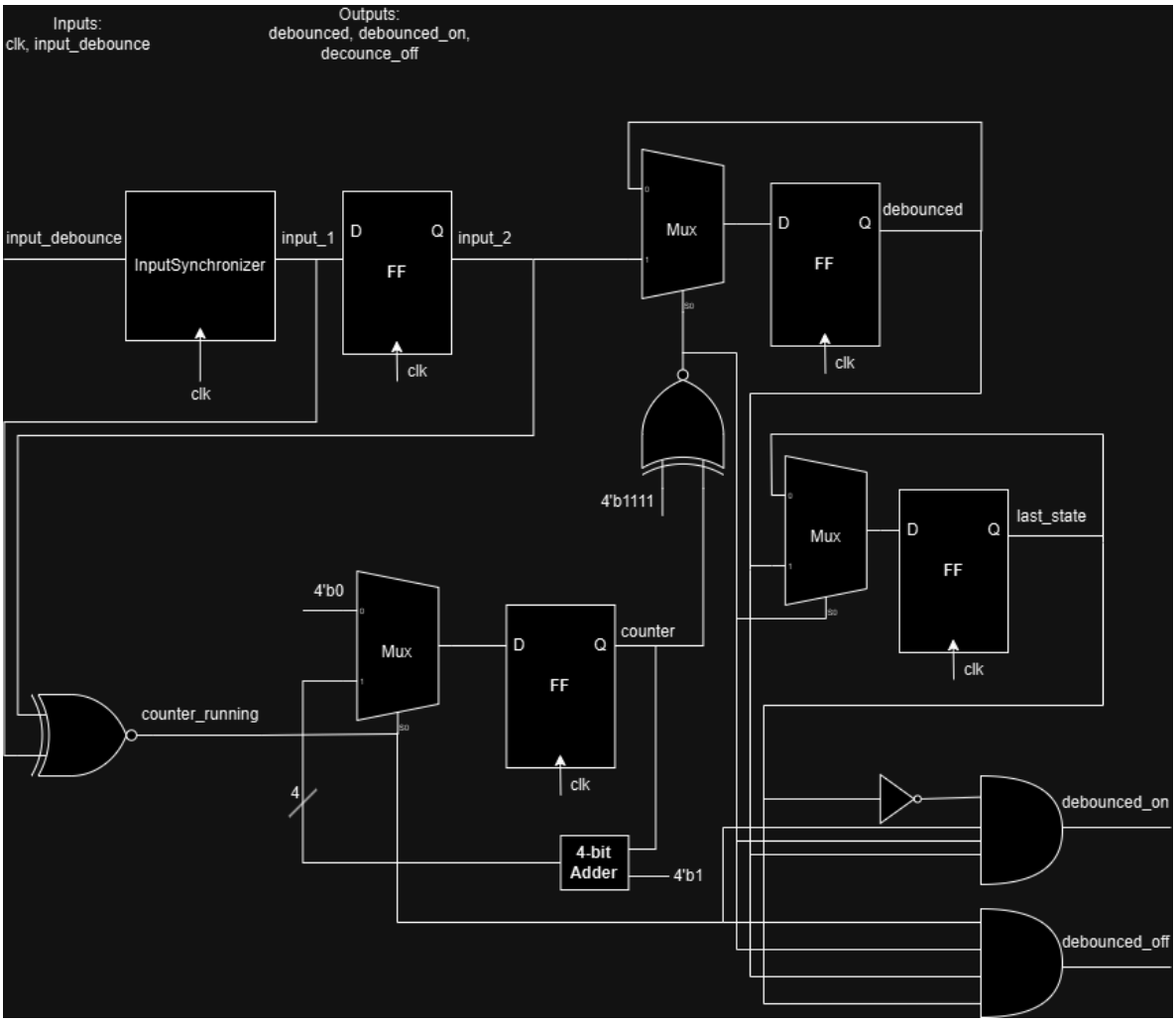
4.1 Υλοποίηση

Για την υλοποίηση του μέρους Β χρησιμοποιήθηκαν τα εξής modules:

- ResetDebouncer: Για το debouncing του κουμπιού του reset.
- Counter: Ένας counter για την ορθή οδήγηση των ανόδων και των καθόδων σύμφωνα με την εκφώνηση.
- AnodeDecoder: Ο decoder που μετατρέπει το σήμα του counter σε σήματα για τις ανόδους των displays.
- CharacterDecoder: Ο decoder που μετατρέπει το σήμα του counter στο σήμα για τον χαρακτήρα που είναι είσοδος στον LEDdecoder.
- LEDdecoder: Ο decoder που υλοποιήθηκε στο μέρος Α και μετατρέπει το σήμα του counter στα σήματα καθόδου των displays.

4.1.1 ResetDebouncer

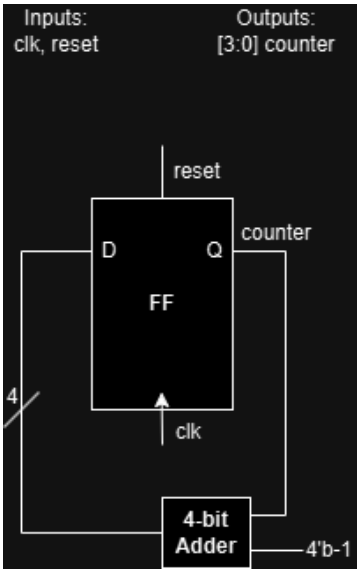
Ο reset debouncer έχει είσοδο το ρολόι και το σήμα reset από το κουμπί. Αρχικά χρησιμοποιεί το module InputSynchronizer το οποίο χρησιμοποιεί 2 flip-flops για να συγχρονίσει το σήμα που δέχεται στην είσοδο του. Χρησιμοποιεί έναν μετρητή 12 bits ώστε να μην επιτρέπει πολύ κοντινά σήματα από το κουμπί. Για την λειτουργία του χρησιμοποιεί ένα ακόμα flip-flop και συνεχώς συγκρίνει την έξοδό του με την έξοδο του synchronizer και μόλις ο μετρητής ολοκληρώσει έναν κύκλο βγάζει το σήμα στην έξοδο μαζί με ένα επιπλέον σήμα το οποίο έχει την ίδια τιμή με την έξοδο αλλά κρατάει μόνο έναν κύκλο.



Σχήμα 3: Dataflow του Reset debouncer

4.1.2 Counter

Ο counter είναι ένα απλό module ακολουθιακής λογικής το οποίο αποτελείται από ένα μόνο flip-flop το οποίο αρχικοποιείται στην τιμή `4'b1111` και έχει σαν είσοδο την έξοδό του από την οποία κάθε φορά αφαιρείται 1. Λόγω της λογικής της πρόσθεσης όταν ο counter φτάσει στο 0 ξεκινάει πάλι από την αρχική του τιμή.



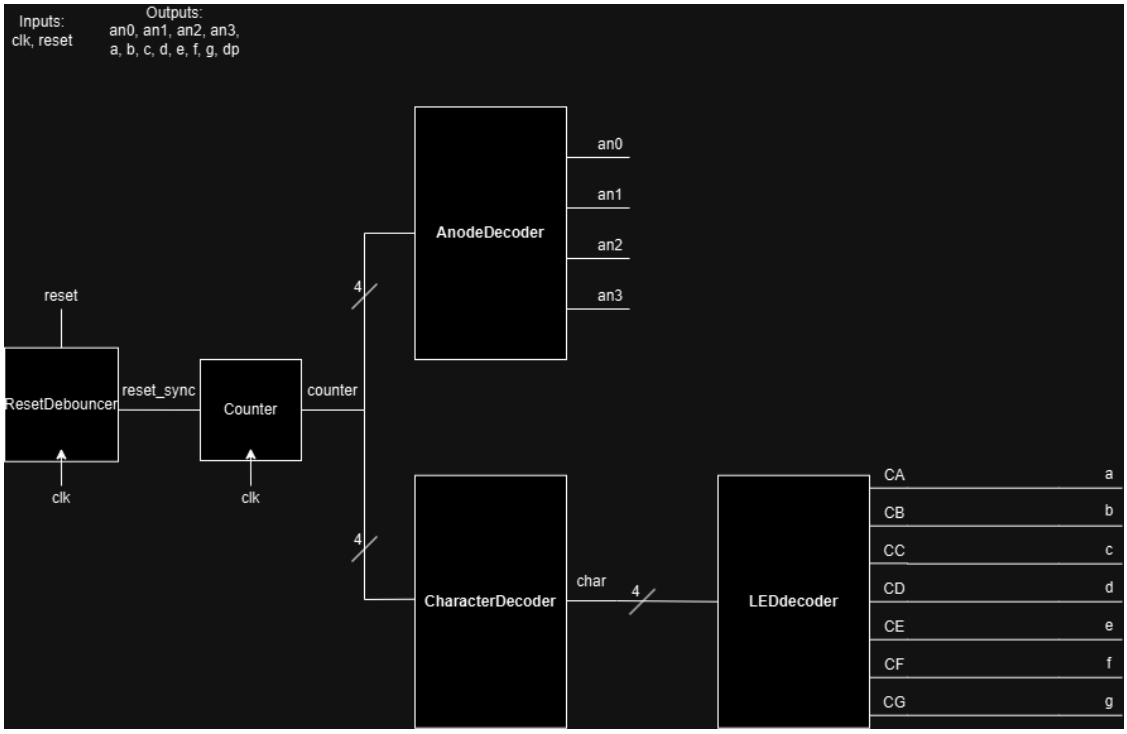
Σχήμα 4: Counter dataflow

4.1.3 AnodeDecoder και CharacterDecoder

Δυο modules συνδυαστικής λογικής τα οποία απλώς μετατρέπουν το σήμα του counter στα σήματα των ανόδων και το σήμα του χαρακτήρα αντίστοιχα σύμφωνα με τον πίνακα που δίνεται στην εκφώνηση. Είναι σημαντικό να προσέξουμε να καλύψουμε όλες τις περιπτώσεις του counter ώστε να μην δημιουργηθεί κάποιο στοιχείο μνήμης.

4.1.4 Top level module

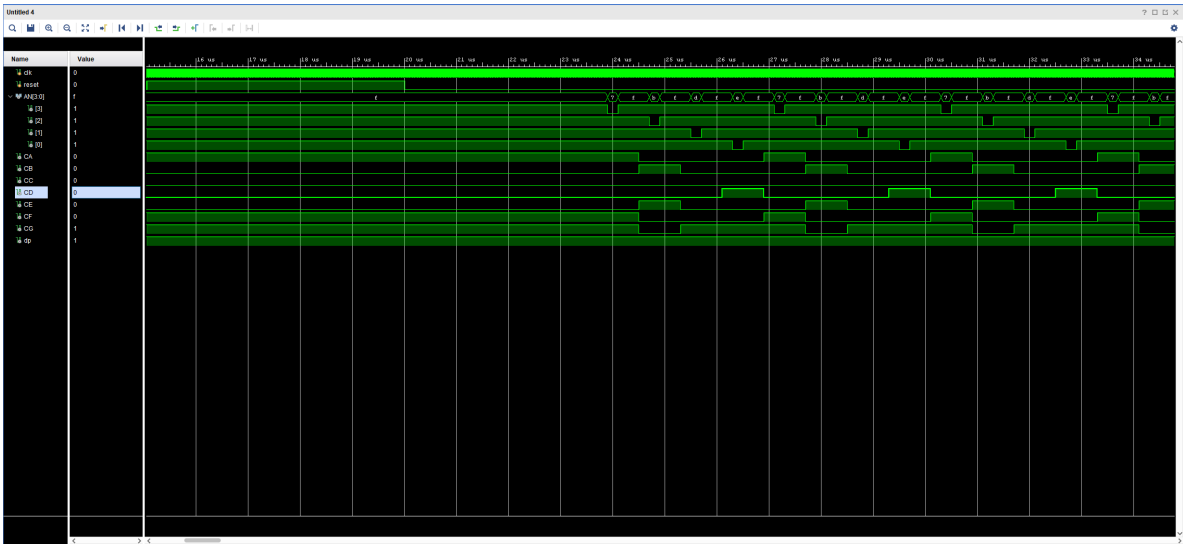
Τελικά συνδέουμε τα modules μεταξύ τους ώστε ο counter να δίνει την έξοδο του στους 2 decoders και ο CharacterDecoder να δίνει την έξοδο του στον LEDdecoder.



Σχήμα 5: Data flow του μέρους B

4.2 Επαλήθευση

Για την επαλύθευση του κυκλώματος χρησιμοποιείται ένα απλό testbench το οποίο οδήγει το ρολόι και το reset σήμα. Συγκεκριμένα το ρολόι έχει περίοδο `10ns` και το reset κρατείται στο 1 για 5ms ώστε να δοκιμαστεί σωστά και το ResetDebouncer module. Δεν είναι απαραίτητη η οδήγηση κάποιου άλλου σήματος. Μέσω των κυματομορφών μπορούμε να διαπιστώσουμε την λειτουργικότητα του κυκλώματος και διαπιστώνουμε πως η πολύπλεξη στον χρόνο πραγματοποίειται ορθά αφού τα σήματα στις ανόδους και καθόδους οδηγούνται τις κατάλληλες στιγμές ώστε να εμφανίζονται οι σωστοί χαρακτήρες στα displays.



Σχήμα 6: Στιγμιότυπο από το simulation του μέρος B

Εδώ μπορούμε να παρατηρήσουμε πως το κύκλωμα ξεκινάει κάποιους κύκλους αφού πέσει το σήμα reset (λόγω του debouncing) και έπειτα το βλέπουμε σε κανονική λειτουργία όπου οι άνοδοι οδηγούνται

στο 0 έναν κύκλο αφού έχει οδηγηθεί το σήμα του χαρακτήρα στον επιθυμητό και μετά παραμένει για έναν κύκλο αφού οδηγηθεί η άνοδος στον άσσο ώστε να μην προκληθεί ghosting. Το τεστβενση αυτό είναι ένας πρώτος έλεγχος για το κύκλωμα στο οποίο παρατηρήθηκαν αρκετά προβλήματα σχετικά με τον συγχρονισμό της οδήγησης των ανόδων και των καθόδων.

4.3 Πείραμα

Στην πραγματική δοκιμή το προγράμματος παρατηρήθηκε ένα πρόβλημα το οποίο οφειλόταν στην λανθασμένο προγραμματισμό των AnodeDecoder και CharacterDecoder οι οποίοι δεν οδηγούσαν στους κατάλληλους χρόνους τις εξόδους τους. Αυτό είχε ως αποτέλεσμα να υπάρχει φαινόμενο ghosting το οποίο ενώ δεν ήταν εμφανές στις προσομοιώσεις, ήταν ξεκάθαρο στην πλακέτα. Με την διόρθωση των modules το πρόβλημα διορθώθηκε.

5 Μέρος Γ - Βηματική περιστροφή του μηνύματος με χρήση κουμπιού

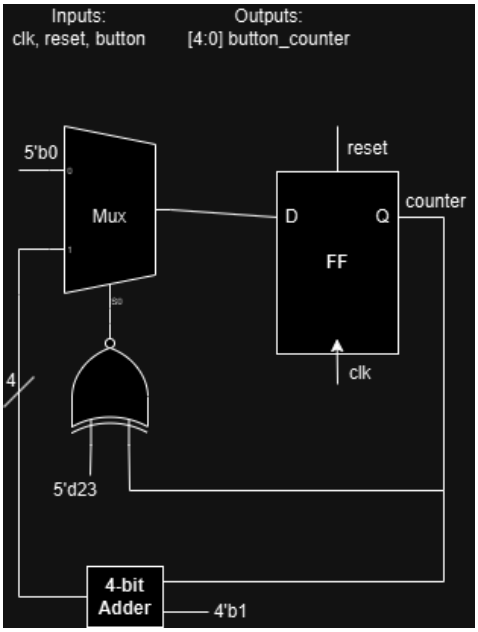
5.1 Υλοποίηση

Για την υλοποίηση του μέρους Γ χρησιμοποιήθηκαν τα ίδια μοδυλες με του μέρους Β και κάποια επιπλέον που χρησιμοποιούνται για την μνήμη του μηνύματος και το κουμπί που χρησιμοποιείται για την περιστροφή του μηνύματος. Τα επιπλέον modules είναι τα εξής:

- ButtonCounter: Για την περιστροφή του μηνύματος.
- CharacterDecoder: Παρόμοια υλοποίηση με του μέρος Β αλλά με μεταβλητούς χαρακτήρες για την έξοδο.
- CharacterMemory: Η μνήμη που κρατάει το μήνυμα που εμφανίζεται.
- InputDebouncer: Για το debouncing του κουμπιού που χρησιμποιείται για την περιστροφή.

5.1.1 ButtonCounter

Ο ButtonCounter χρησιμοποιείται για να μετράμε τις φορές που έχει πατηθεί το κουμπί για την περιστροφή. Όταν ο αριθμός αυτός ξεπεράσει τις νόμιμες θέσεις στην μνήμη των χαρακτήρων ο counter ξεκινάει απο την αρχή οπότε και το μήνυμα ξεκινάει από την αρχή.



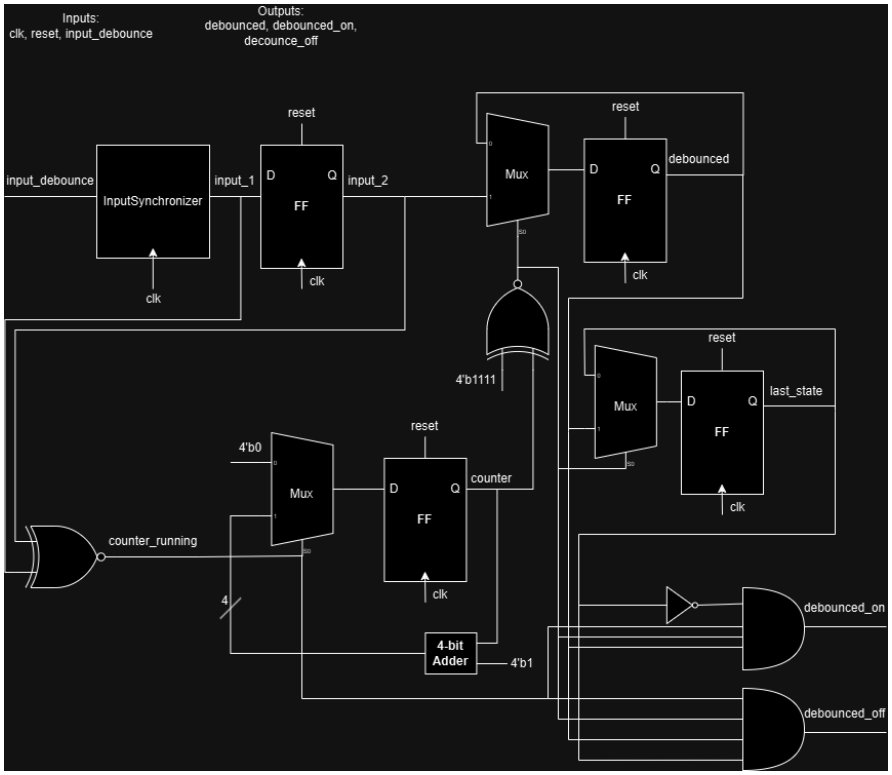
Σχήμα 7: ButtonCounter dataflow

5.1.2 CharacterMemory

Το CharacterMemory κρατάει το μήνυμα που θα εμφανίζεται στο display και έχει ως είσοδο μια διεύθυνση και βγάζει έξοδο τον χαρακτήρα σε εκείνη την διεύθυνση και τους 3 επόμενους χαρακτήρες. Η μνήμη αρχικοποιείται όταν το reset είναι 1 και δεν αλλάζει σε άλλη στιγμή.

5.1.3 InputDebouncer

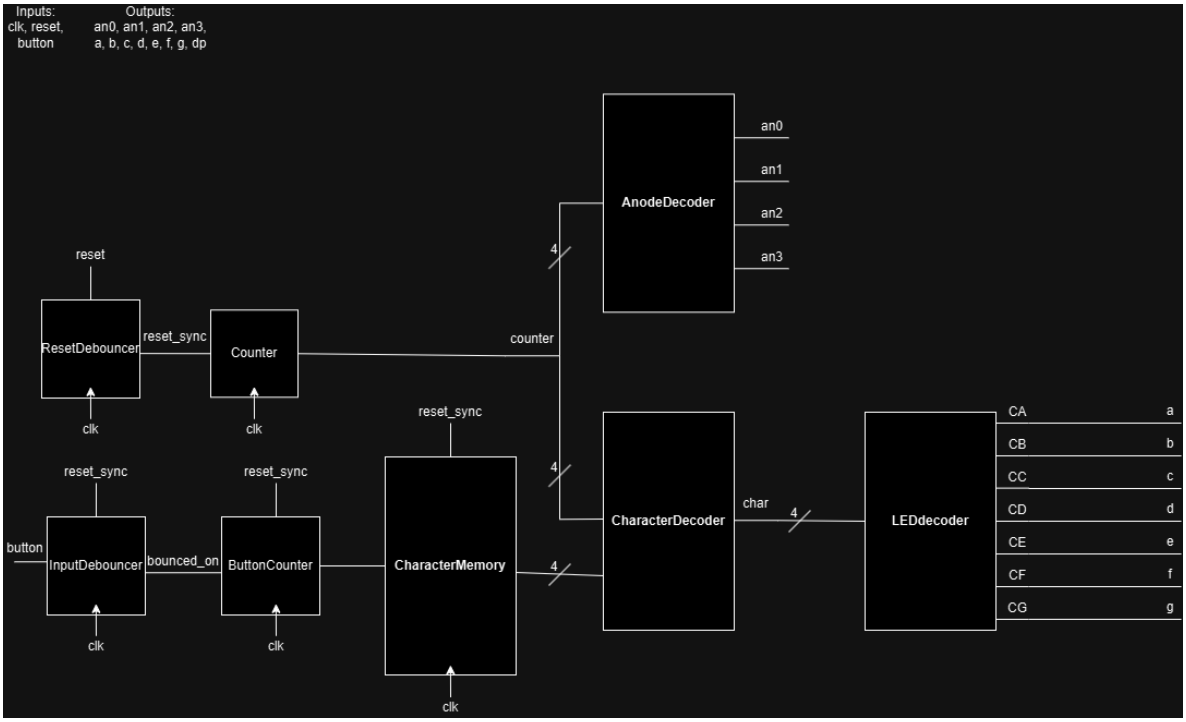
Το InputDebouncer είναι αρκετά παρόμοιο με το ResetDebouncer του μέρους Β απλώς εδώ αλλάζει ότι όλα τα flip-flops του έχουν σήμα reset.



Σχήμα 8: Input debouncer dataflow

5.1.4 Top level module

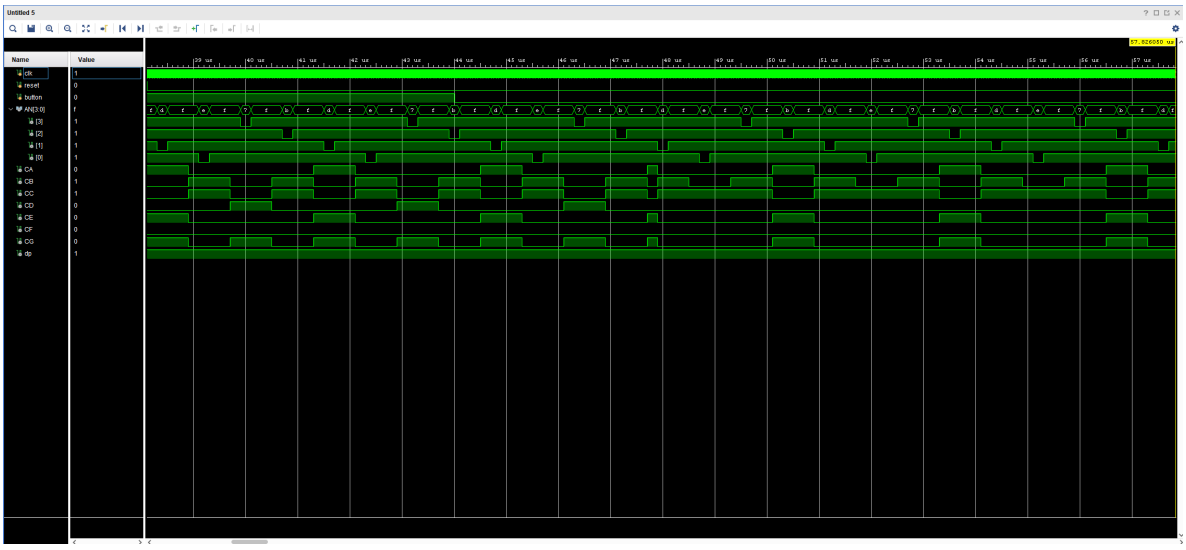
Το top level module έχει κάποιες επιπλέον συνδέσεις από αυτό του μέρους Β οι οποίες είναι το κουμπί με το debouncer και το memory. Επιπλέον το memory δίνει την έξοδο του στον CharacterDecoder.



Σχήμα 9: Dataflow του μέρους Γ

5.2 Επαλήθευση

Για την επαλύθευση χρησιμοποιήθηκε ένα testbench στο οποίο εκτός από το ρολόι και το σήμα του reset οδηγείται και ένα επιπλέον σήμα το οποίο παίζει το ρόλο του κουμπιού στην πλακέτα. Παρομοίως και αυτό κρατείται για αρκετή ώρα στον άσσο ώστε να προσομοιώνεται ένα πραγματικό πάτημα σε κουμπί και να ελέγχεται η λειτουργία του button debouncer.



Σχήμα 10: Στιγμιότυπο simulation μέρους Γ

Απο εδώ μπορούμε να παρατηρήσουμε ότι το κύκλωμα έχει την ίδια συμπεριφορά με το κύκλωμα στο μέρος Β αλλά εδώ έχουμε το σήμα button το οποίο όταν από 1 πάει σε 0 και περάσουν οι κύκλοι που απαιτεί το button debouncer οι χαρακτήρες που βγάζει το character decoder αλλάζουν και είναι οι επόμενοι που πρέπει να εμφανιστούν σύμφωνα με την μνήμη του μηνύματος. Εδώ είναι σημαντικό να προσέξουμε ότι

μπορεί για τον πρώτο κύκλο οι κάθοδοι να μην οδηγηθούν σε σωστούς χρόνους όμως αυτό θα συμβεί για τον πρώτο κύκλο και άρα δεν θα είναι ορατό πρακτικά.

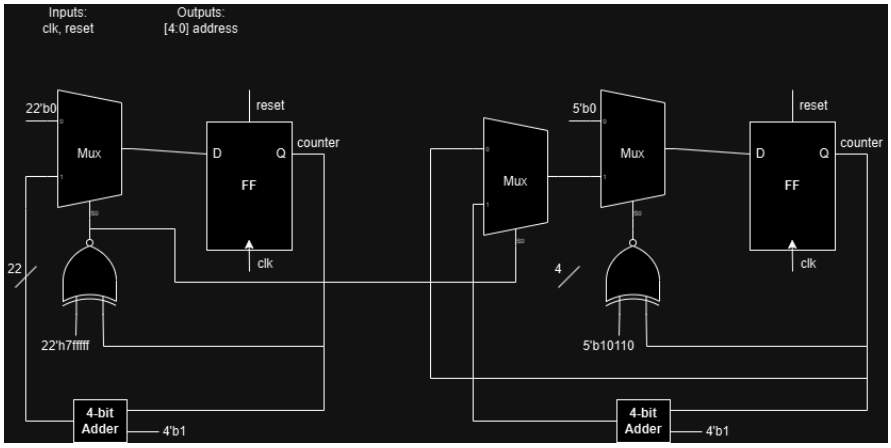
5.3 Πείραμα

Μέσω του πειράματος διαπιστώθηκε πως το κύκλωμα λειτουργούσε ορθά όμως ήταν προφανές ότι το debouncer δεν έιχε αρκετά μεγάλο counter ώστε να λειτουργεί σωστά. Με την αύξηση των bits στον counter το πρόβλημα διορθώθηκε.

6 Μέρος Δ - Βηματική περιστροφή του μηνύματος με σταθερή καθυστέρηση

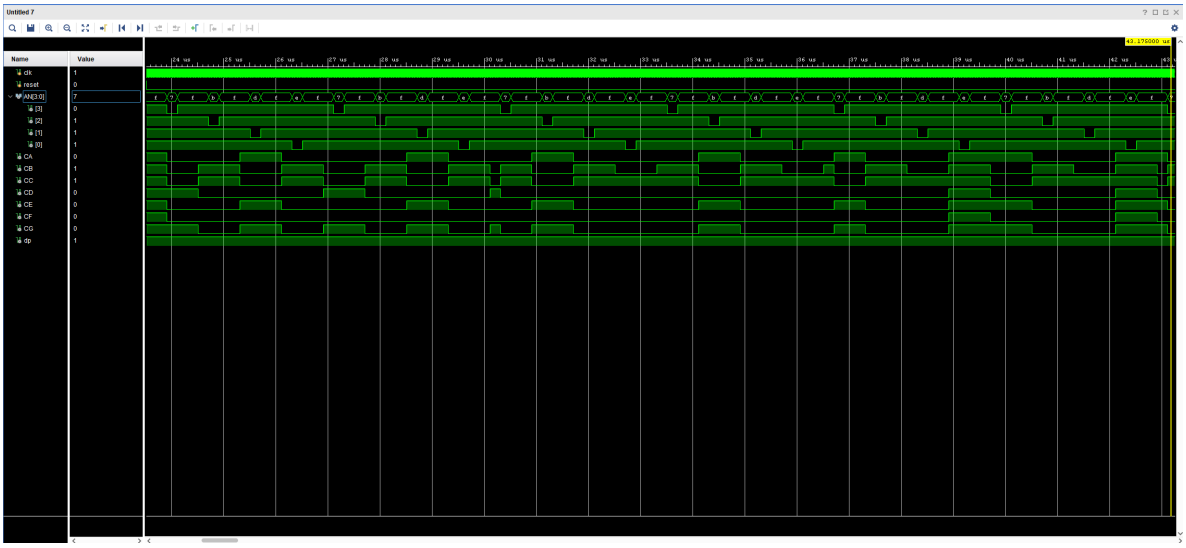
6.1 Υλοποίηση

Η μόνη διαφορά που υπάρχει στο μέρος Δ με το μέρος Γ είναι ότι στο μέρος Δ αντί να έχουμε ButtonCounter και InputDebouncer έχουμε έναν απλό counter των 23 bits ο οποίος κάθε φορά που κάνει wrap around προσθέτει 1 στην address που πάει στο CharacterMemory και όταν ξεπεράσει τις νόμιμες θέσεις της μνήμης ξεκινάει την διεύθυνση από το 0.

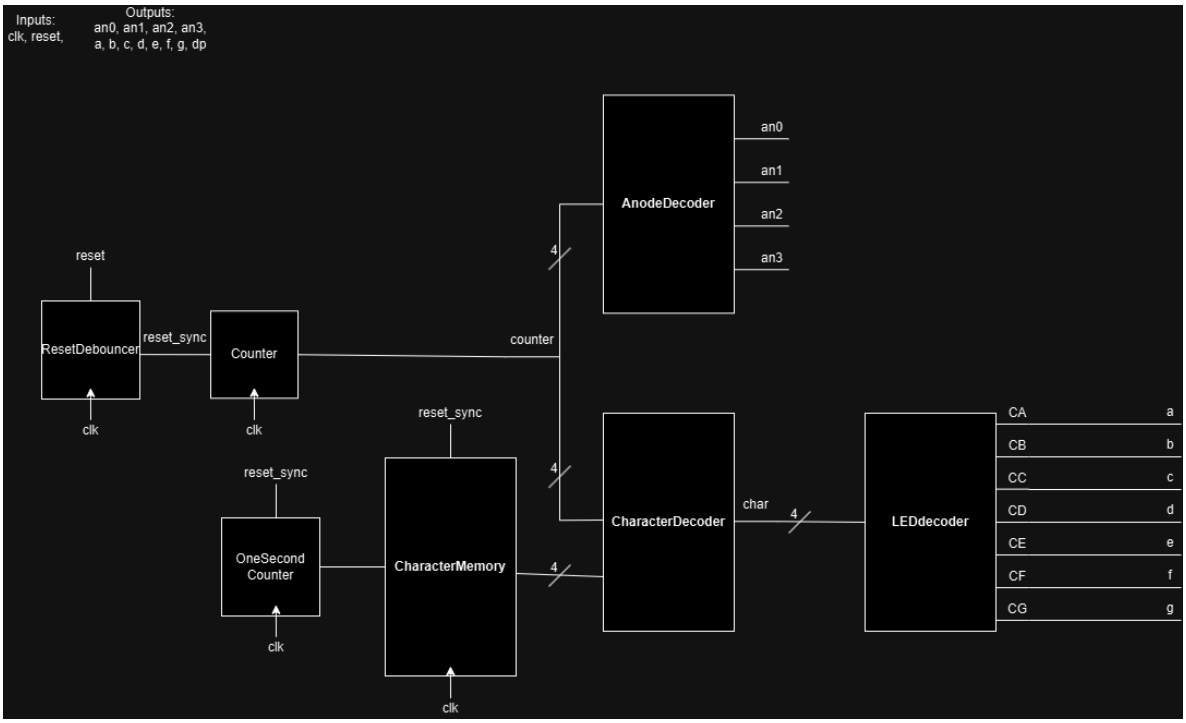


Σχήμα 11: OneSecondCounter dataflow

Και το τελικό top level module έχει το εξής dataflow:



Σχήμα 13: Στιγμιότυπο προσομοίωσης μέρους Δ



Σχήμα 12: Dataflow μέρους Δ

6.2 Επαλήθευση

Η επαλήθευση εδώ γίνεται με το ίδιο testbench του μέρους Β καθώς πάλι δεν υπάρχει ανάγκη για την οδήγηση του κουμπιού για την περιστροφή. Εδώ παρατηρούμε πως η έξοδος του CharacterDecoder αλλάζει μετά από κάποιους κύκλους και δείχνει τους επόμενους χαρακτήρες πάλι σύφμωνα με την μνήμη. (Για την προσομοίωση χρησιμοπήθηκε μικρότερος counter ώστε να μην χρειάζεται να τρέξει για πολυ χρόνο)

6.3 Πείραμα

Στην πραγματική δοκιμή στην πλακέτα σε αυτήν την περίπτωση δεν προέκυψαν προβλήματα και το κύκλωμα δούλεψε ορθά.