

# Report I

Τσόγκας Παναγιώτης Νικόλαος - 3672

23/10/2024

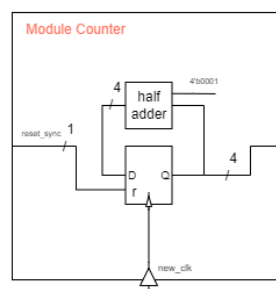
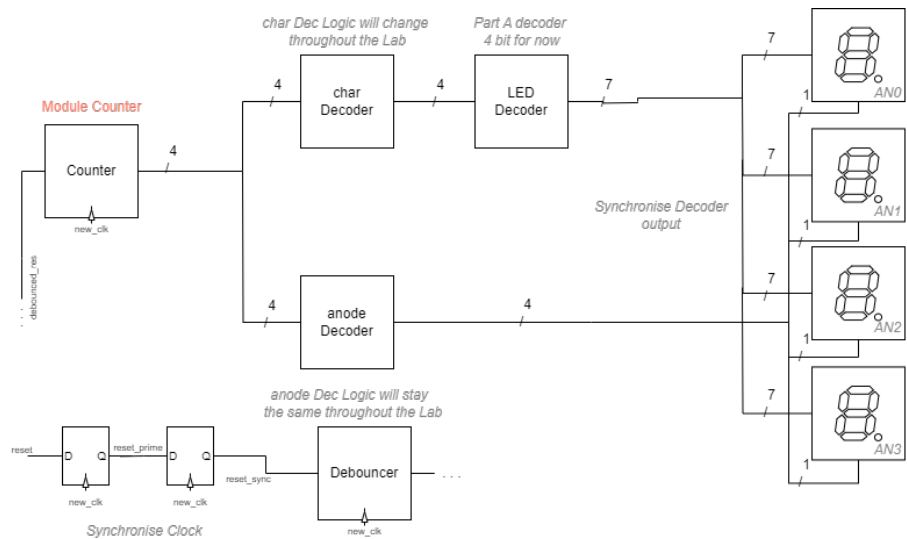
## Περίληψη

Η αναφορά αυτή αφορά την ανάπτυξη και υλοποίηση ενός συστήματος αποκωδικοποίησης και προβολής χαρακτήρων μέσω LED, βασισμένου σε μια σειρά από διακριτά modules, όπως το "LED Decoder", "Anode Decoder", και "Debouncer". Μέσα από διάφορα στάδια (Part A-D), εξετάζονται διαφορετικές προσεγγίσεις για την απεικόνιση σταθερών και δυναμικών χαρακτήρων σε Seven Segment Displays, με στόχο τη βελτίωση της σταθερότητας του σήματος και της λειτουργικότητας των κουμπιών.

## Εισαγωγή

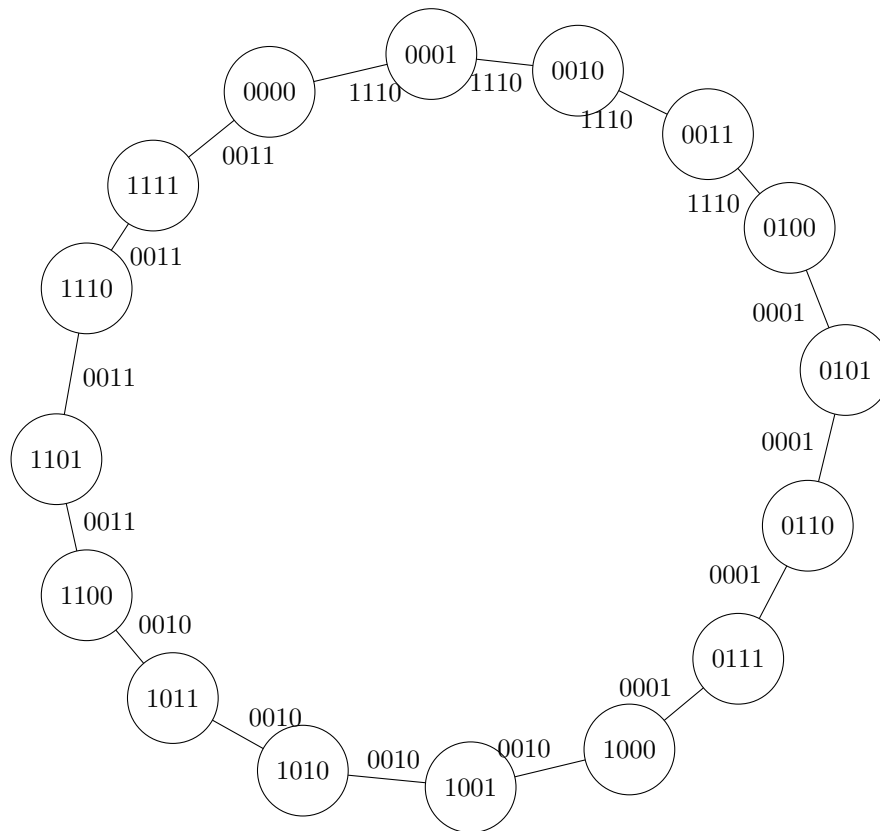
Ο στόχος της εργασίας είναι η σχεδίαση και υλοποίηση ενός αποκωδικοποιητή LED που μπορεί να μεταφράσει δεδομένα σε ορατούς χαρακτήρες, με την επέκταση της λειτουργίας του σε πιο σύνθετα συστήματα που περιλαμβάνουν δυναμική διαχείριση χαρακτήρων και χειρισμό κουμπιών. Οι στόχοι επιτεύχθηκαν μέσω της κατασκευής διαφορετικών modules που χρησιμοποιούν counters, αποκωδικοποιητές και μνήμη, ενώ επιλύθηκαν προβλήματα όπως το "ghosting" στην απεικόνιση και η σταθεροποίηση των εισόδων από κουμπιά με τη χρήση ενός "Debouncer".





Σχήμα 2: Dataflow

Μηχανή Πεπερασμένων Καταστάσεων :



Σχήμα 3: FSM for CharacterDecoder

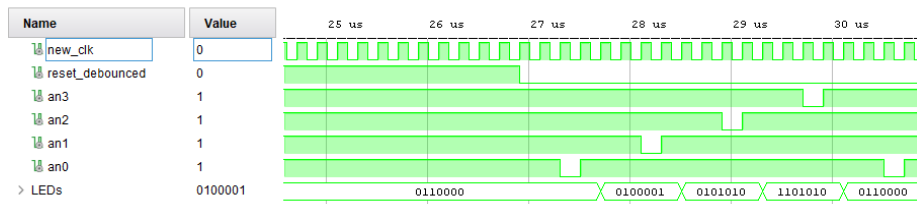
Τα modules παρουσιάζονται στα αντίστοιχα Part από ένα αρχείο module\_header.x.

## (2) Επαλήθευση:

Για το testbench του Part B απλά δοκιμάζουμε την λειτουργία του reset και την συμπεριφορά του σήματος debounced\_reset με την βοήθεια των κυματομορφών. Κάποια καλά πειράματα :

- Το κύκλωμα ξεκινάει με το reset πατημένο
- Το reset κρατιέται πατημένο για πολλούς κύκλους
- Το reset γίνεται ακριβώς στο posedge clk

Κυματομορφή σωστής υλοποίησης :



Σχήμα 3: VCD

### (3) Πείραμα/Τελική Υλοποίηση:

Με μία παλιά υλοποίηση στην οποία τα Anode άνοιγαν στον κύκλο ακριβώς πριν αλλάξει ο χαρακτήρας υπήρχε "ghosting". Για αυτό στην σωστή υλοποίηση ο char κρατάει την τιμή του για ένα κύκλο μετά το posedge του προηγούμενου Anode.

## Part C

### (1) Υλοποίηση:

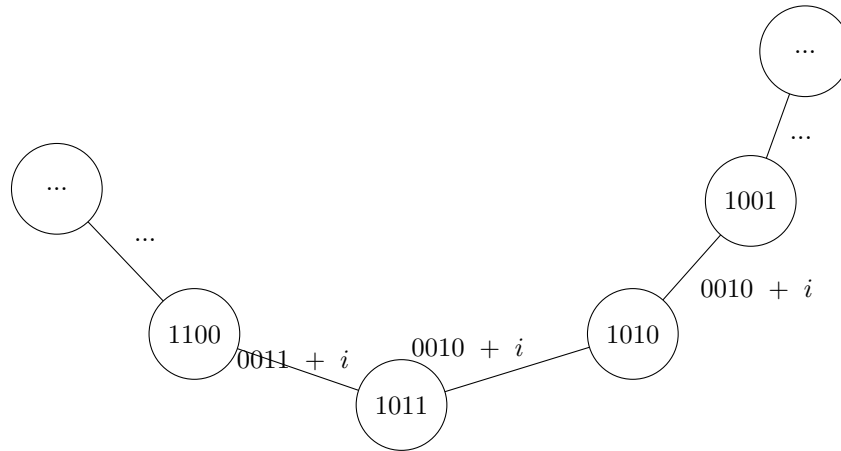
Για την σχεδίαση του μέρους Γ θα χρειαστεί να χρησιμοποιήσουμε ένα button για τον χειρισμό του οποίου θα μιλήσουμε παρακάτω. Παρατηρούμε ότι μπορούμε να χρησιμοποιήσουμε την παραπάνω υλοποίηση του Driver μας, αφού ουσιαστικά το κομμάτι αυτό της εργασίας είναι ίδιο με το προηγούμενο απλά θα χρειαστούμε 'δυναμικούς' char. Με άλλα λόγια χρειαζόμαστε μία δομή Memory, δηλαδή πολλά FF (Flip-Flop) τα οποία κρατάνε συνεχώς μνήμη και ανάλογα με τα πατήματα του button διαλέγουμε άλλους χαρακτήρες για εμφάνιση. Για τον χειρισμό του button θα χρειαστούμε ένα module "Debouncer" και κάποια λογική καταμέτρησης των πατημάτων. Για δική μας ευκολία θα το κάνουμε ένα καινούργιο module "Incrementer", έτσι ώστε να μην ξαναγράψουμε λογική στο Part D.

Τα module που χρησιμοποιήθηκαν.

- "AnodeDecoder.v"
- "CharacterDecoder.v"
- "ConstCounter.v"
- "Debouncer.v"
- "Incrementer.v"
- "LEDdecoder.v"



Η Μηχανή Πεπερασμένων Καταστάσεων είναι σχεδόν η ίδια με το κομμάτι B και είναι :



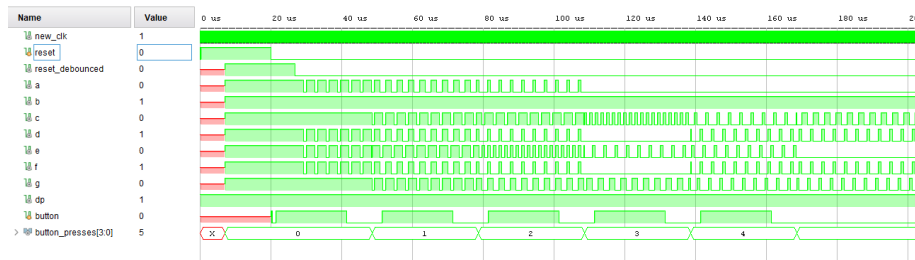
Τα modules παρουσιάζονται στα αντίστοιχα Part απο ένα αρχείο module\_header\_x.

## (2) Επαλήθευση:

Για το testbench του Part C απλά δοκιμάζουμε την λειτουργία του κυκλώματος με ένα button που έχει και noise και την συμπεριφορά του σήματος debounced\_button, button\_presses με την βοήθεια των κυματομορφών. Κάποια καλά πειράματα :

- Το κύκλωμα ξεκινάει με το button πατημένο πριν γίνει reset
- Το button πατιέται παραπάνω φορές απο ότι έχουμε memory
- Το button κρατιεται πατημένο (για να δούμε την λειτουργία του Incrementer).

Κυματομορφή σωστής υλοποίησης :



Σχήμα 5: VCD

### (3) Πείραμα/Τελική Υλοποίηση:

Με μία παλιά υλοποίηση στην οποία ο Incrementer λειτουργούσε στο "Posedge Button" δεν έκανε σωστά την περίπτωση 1.

## Part D

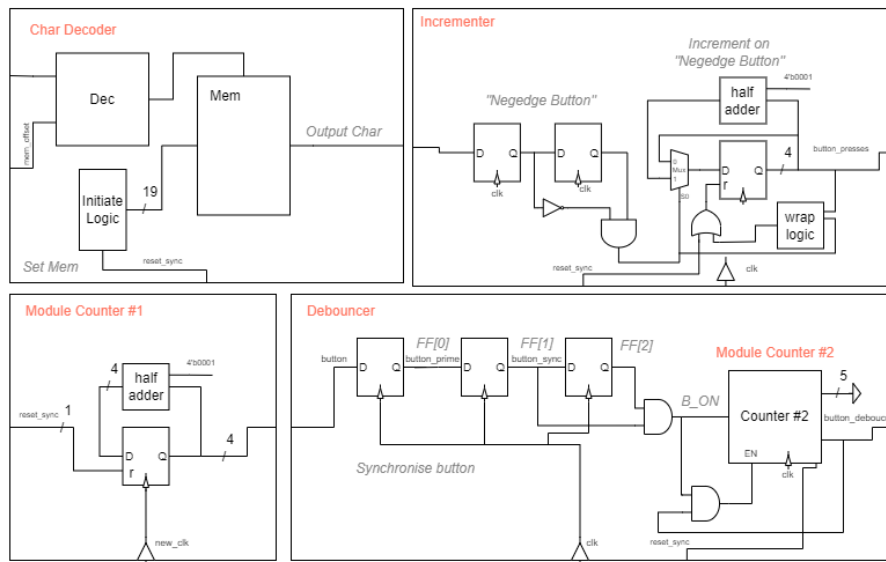
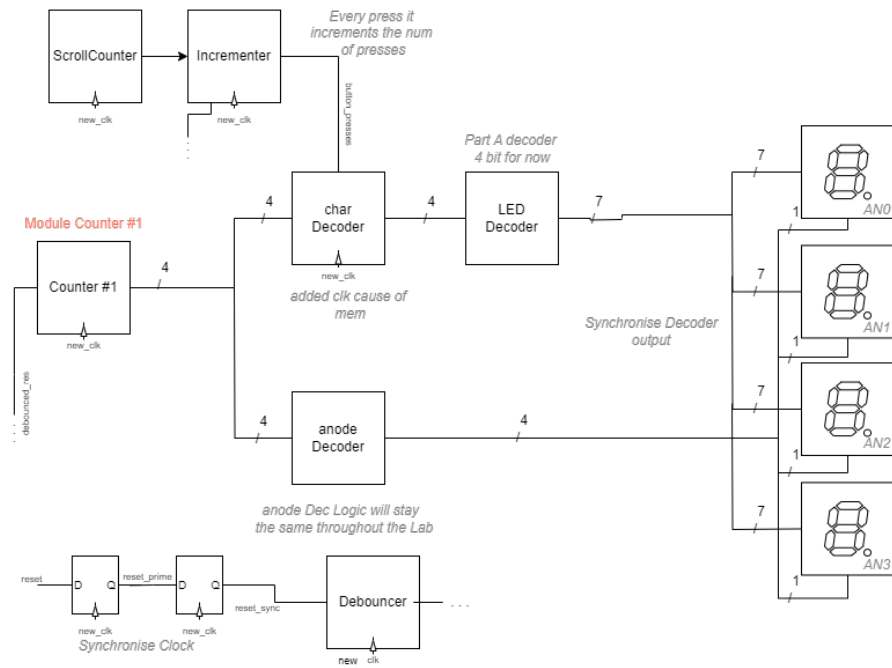
### (1) Υλοποίηση:

Για το Part D λόγω του ότι κάναμε το Incrementer ξεχωριστό module μπορούμε απλα να αλλάξουμε το button με έναν μεγάλο counter τον οποίο έστω οτι ονομάζουμε ScrollCounter το οποίο για το κύκλωμα θα μπορούσε να είναι και απλα ένα κουμπί, με άλλα λόγια δεν καταλαβαίνει την διαφορά. Προφανώς δεν χρειαζόμαστε πλέον τον έναν Debouncer.

Τα module που χρησιμοποιήθηκαν :

- "AnodeDecoder.v"
- "CharacterDecoder.v"
- "ConstCounter.v"
- "Debouncer.v"
- "Incrementer.v"
- "LEDdecoder.v"
- "ScrollCounter.v"





Σχήμα 6: Dataflow

## Modules - Part by part Analysis

Σημείωση: Κάθε κομμάτι της εργασίας έχει δικό του testbench. Επιπλέον υπάρχει testbench για τον Debouncer. Για μία καλύτερη εικόνα πως λειτουργούν κάποια απο τα πιο σύνθετα modules, έχουν δοθεί παραπάνω στα Dataflow κάποιες πιθανές υλοποιήσεις που περιγράφουν το behaviour καλά και την λογική πίσω απο αυτά τα module.

### 0.1 A

```
Add LEDdecoder, Add constraints
[+] module LEDdecoder
[+] constraints.xdc
```

### 0.2 B

```
Add AnodeDecoder, Add CharacterDecoder, Add ConstCounter
Add Debouncer, Add FourDigitLEDdriver
[+] module AnodeDecoder
[+] module CharacterDecoder
[+] module ConstCounter
[+] module Debouncer
[+] module FourDigitLEDdriver
```

### 0.3 C

```
Add Memory (change CharacterDecoder), Add Incrementer module,
Add button to constraints
[!] module CharacterDecoder
[+] module Incrementer
[!] constraints.xdc
```

### 0.4 D

```
Add ScrollCounter module, Remove button from constraints
[+] module ScrollCounter
[!] constraints.xdc
```