Report I

Τσόγκας Παναγιώτης Νικόλαος - 3672

23/10/2024

Περίληψη

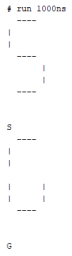
Η αναφορά αυτή αφορά την ανάπτυξη και υλοποίηση ενός συστήματος αποκωδικοποίησης και προβολής χαρακτήρων μέσω LED, βασισμένου σε μια σειρά από διακριτά modules, όπως το "LED Decoder", "Anode Decoder", και "Debouncer". Μέσα από διάφορα στάδια (Part A-D), εξετάζονται διαφορετικές προσεγγίσεις για την απεικόνιση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

Εισαγωγή

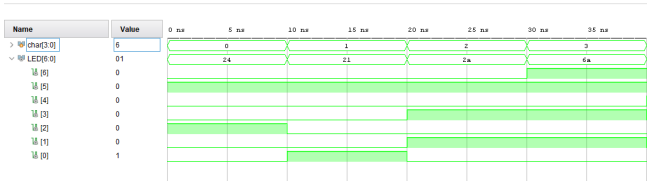
Ο στόχος της εργασίας είναι η σχεδίαση και υλοποίηση ενός αποκωδικοποιητή LED που μπορεί να μεταφράσει δεδομένα σε ορατούς χαρακτήρες, με την επέκταση της λειτουργίας του σε πιο σύνθετα συστήματα που περιλαμβάνουν δυναμική διαχείριση χαρακτήρων και χειρισμό κουμπιών. Οι στόχοι επιτεύχθηκαν μέσω της κατασκευής 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".

Part A

Η υλοποιήση και η σκεπτική του LED Decoder είναι αρκετά απλή. Ουσιαστικά θέλουμε να μετατρέψουμε ένα σήμα σε ένα άλλο, πράγμα που γίνεται με μια συνάρτηση module (Black Box) που την λέμε LedDecoder. Τρέχοντας το testbench.v βλέπουμε μία αναπαράσταση των LED στο τερματικό και τον πραγματικό χαρακτήρα ακριβώς απο κάτω, μέσω του task: "display_segment".
Παράδειγμα εξόδου testbench.v :



(α′) Αρχή εξόσου



(β′) VCD

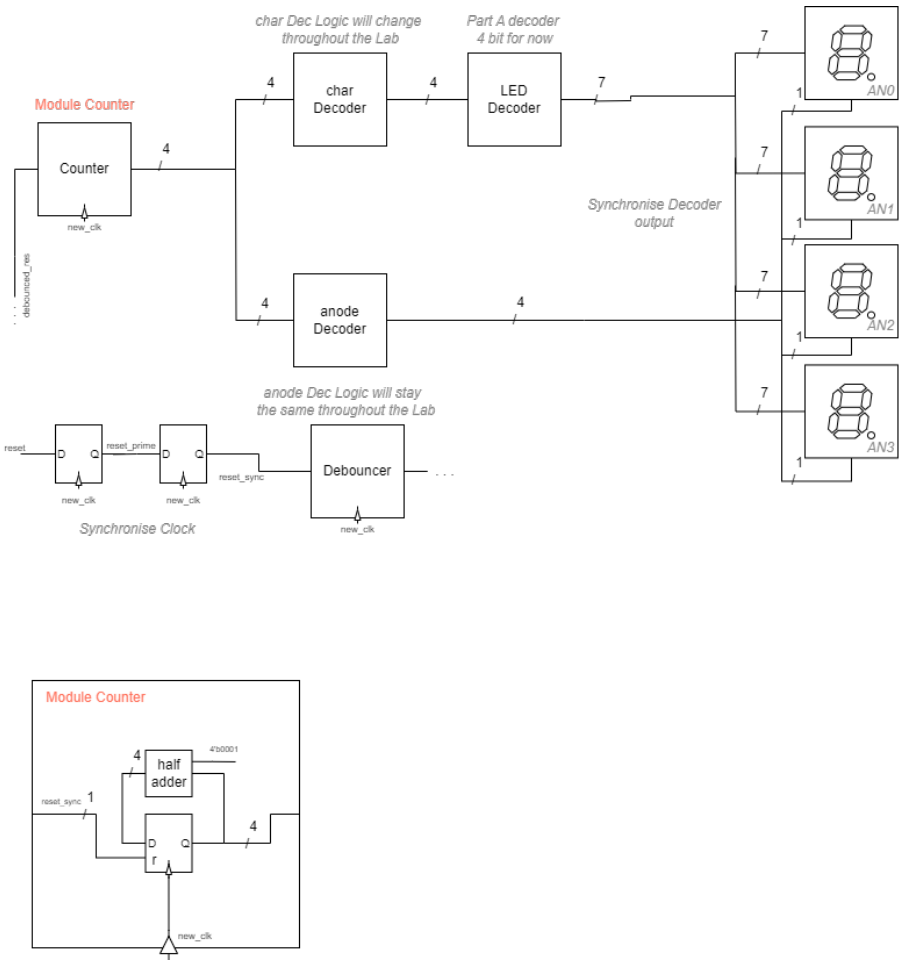
Part B

(1) Υλοποίηση:

Για την σχεδίαση του μέρους B αρχικά παρατηρούμε οτι χρειαζόμαστε έναν γρήγορο και συνεχές counter για να ανανεώνουνμε το μήνυμα που θέλουμε να δείξουμε. Χρησιμοποιώντας τον counter σύμφωνα με την λειτουργία του SSD (Seven Segment Display) της πλακέτας μεταφράζουμε τον counter σε ένα bus το οποίο διαλέγουμε που εμφανίζεται με την βοήθεια των Anode. Συνεπώς θα χρειαστούμε τουλάχιστον δύο Decoders. Αξιοποιώντας τον LED Decoder απο το Part A έχουμε τρείς Decoder όπως φαίνετε στο αντοίστοιχο αρχείο PART_B.drawio .

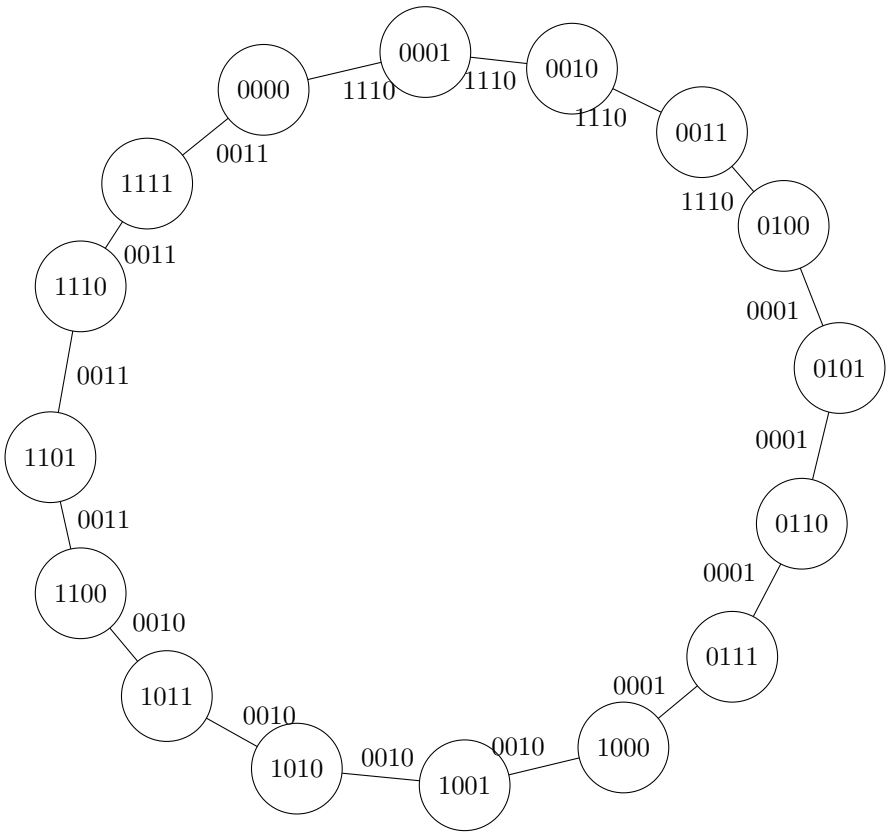
Τα module που χρησιμοποιήθηκαν.

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



Σχήμα 2: Dataflow

Μηχανή Πεπερασμένων Καταστάσεων :



Σχήμα 3: FSM for CharacterDecoder

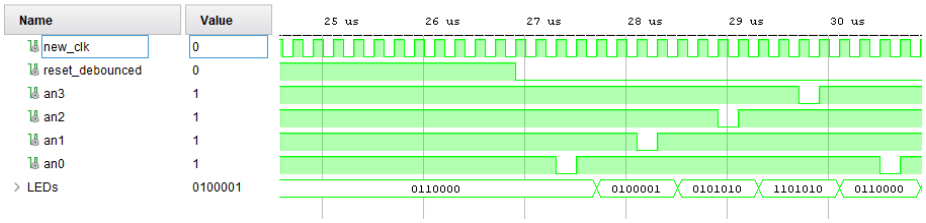
Τα modules παρουσιάζονται και εξηγούνται στα αντοίστοιχα Part απο ένα αρχείο module_header_x.

(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 πολούς 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.

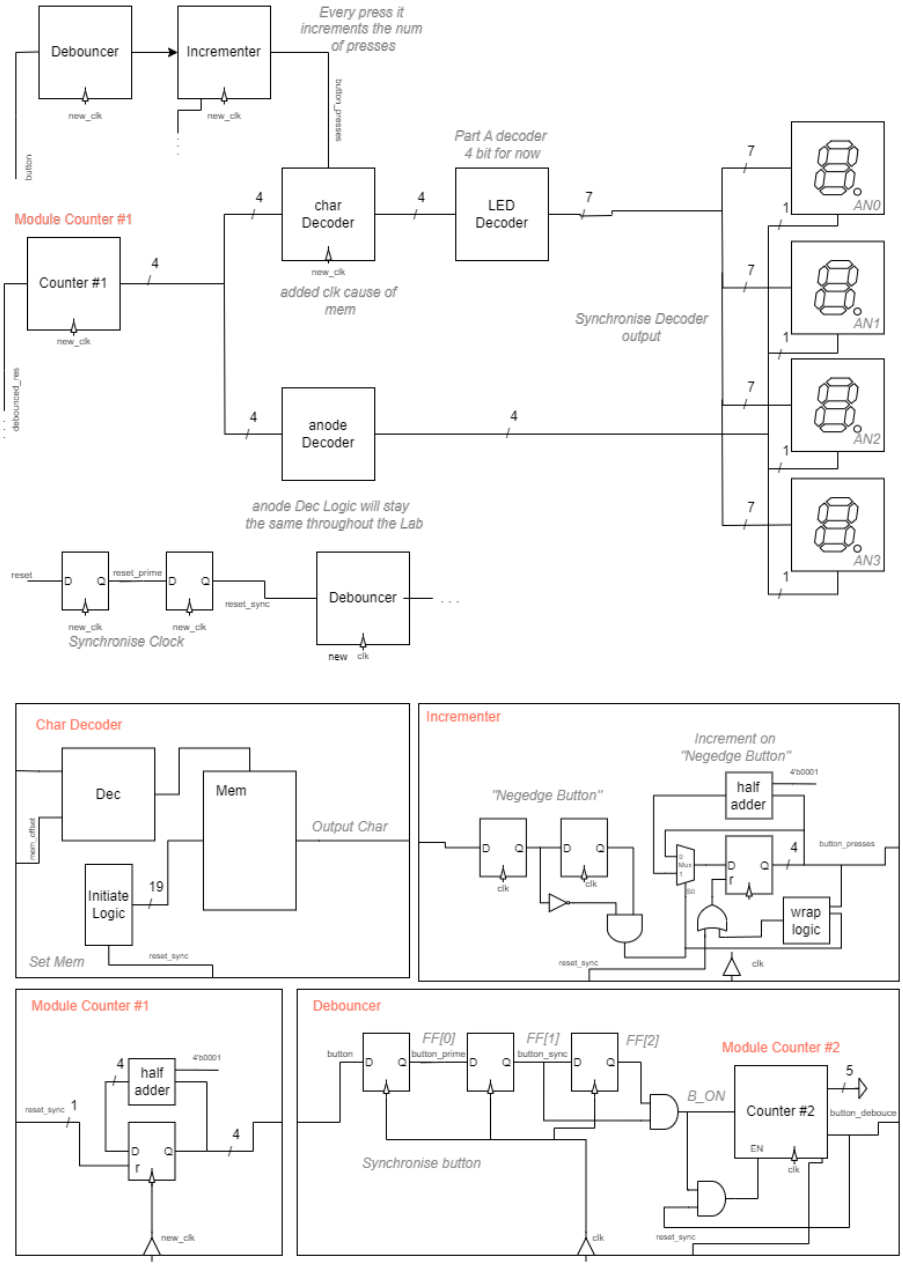
Part C

(1) Υλοποίηση:

Για την σχεδίαση του μέρους Γ θα χρειαστεί να χρησιμοποιήσουμε ένα button για του οποίου τον χειρισμό θα μιλήσουμε παρακάτω. Παρατηρούμε ότι δεν χρειάζεται να αλλάζουμε την προηγούνη υλοποίηση αφού ουσιαστικά το κομμάτι της εργασίας είναι ίδιο με το προηγούμενο απλά χρειαζόμαστε '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 δικιά μας ευκολία θα το ορίσουμε ως ένα καινούργιο module "Incrementer", ετσι ώστε να μην ξαναγράφουμε λογική στο Part D.

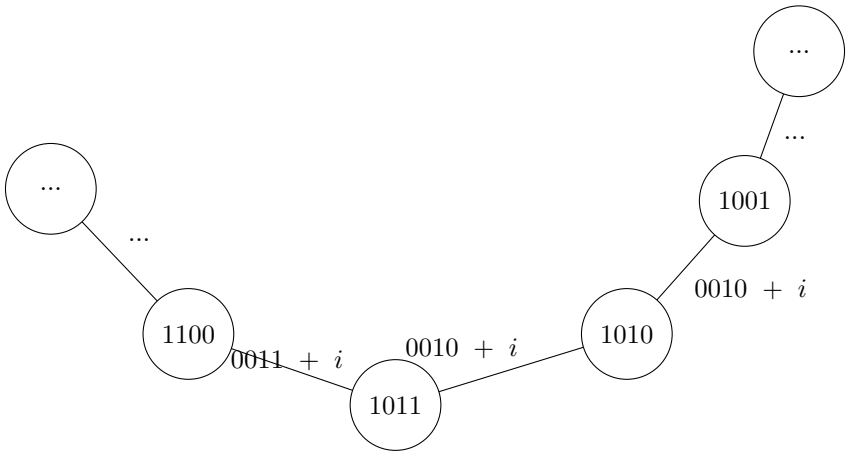
Τα module που χρησιμοποιήθηκαν.

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



Σχήμα 4: Dataflow

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



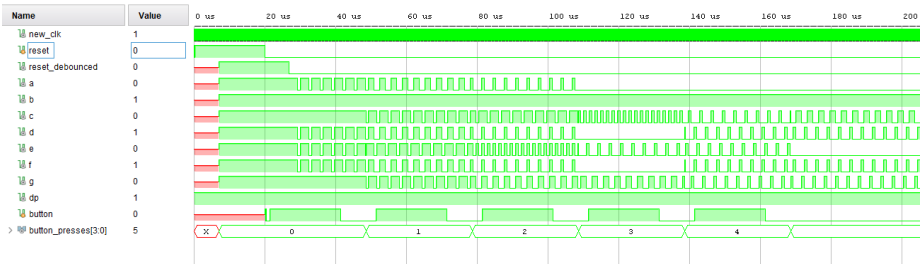
Τα modules παρουσιάζονται και εξηγούνται στα αντοίστοιχα Part απο ένα αρχείο module_header_x.

(2) Επαλήθευση:

Για το testbench του Part C απλά δοκιμάζουμε την λειτουργία του κυκλώματος με ενα 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 γινει reset
- Το button πατιέται παραπάνω φορες απο ότι έχουμε memory
- Το button κρατιεται πατημένο (για να δούμε την λειτουργεία του Incrementer.

Κυματομορφή σωστής υλοποίησης :



Σχήμα 5: VCD

(3) Πείραμα/Τελική Υλοποίηση:

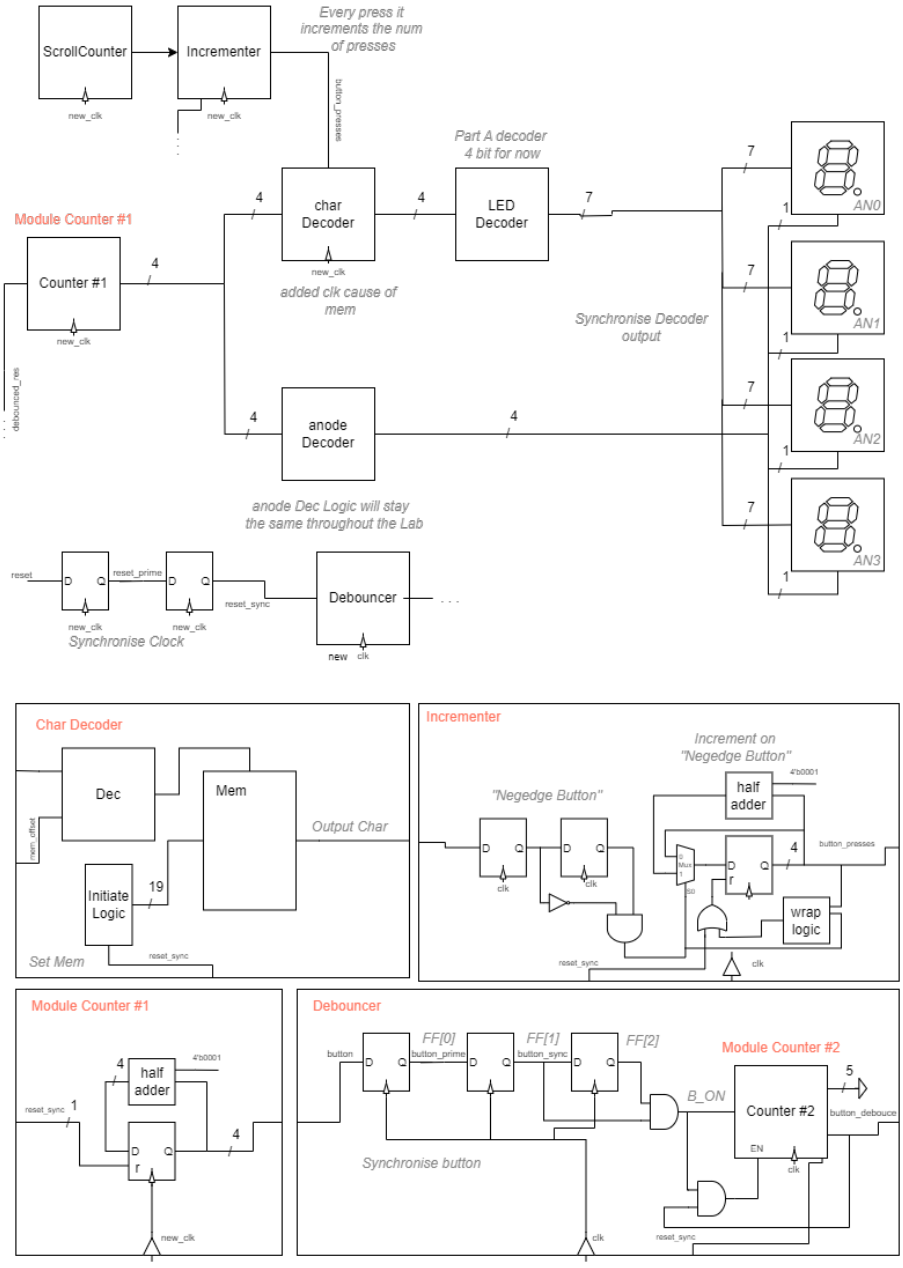
Με μία παλιά υλοποίηση στην οποία ο Incrementer λειτουργούσε στο "Posedge Button" δεν έκανε σωστά την περίπτωση 1.

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 πλεον τον έναν Debouncer.
Τα module που χρησιμοποιήθηκαν :

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



Σχήμα 6: Dataflow