LAB1 REPORT

1) Τίτλος: 7-Segment Display Driver
ΧΡΙΣΤΟΔΟΥΛΟΣ ΖΕΡΔΑΛΗΣ, ΑΕΜ: 03531, 28/10/2024

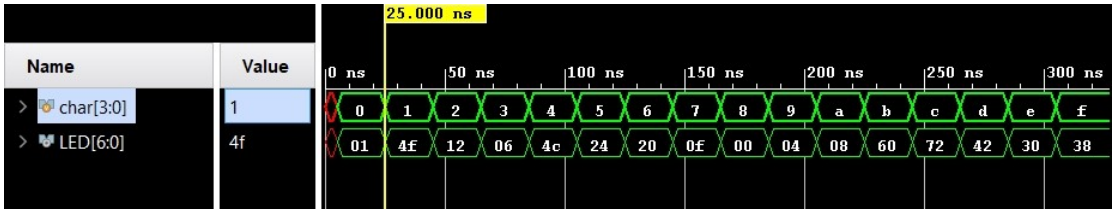
2) ΠΕΡΙΛΗΨΗ: Σε αυτή την εργασία υλοποιήθηκε ένας οδηγός για τις τέσσερις ενδείξεις 7-τμημάτων LED. Στο report θα αναλυθεί ο τρόπος σκέψης, υλοποίησης αλλά και δοκιμής που χρειάστηκε για να ολοκληρωθούν τα τέσσερα μέρη της εργασίας.

3) ΕΙΣΑΓΩΓΗ: Πρώτος στόχος της εργασίας ήταν η δημιουργία της μονάδας LED decoder που ανάλογα με τον χαρακτήρα που το δίνεται ενεργοποιεί τα καταλληλά σήματα στην οθόνη της πλακέτας. Έπειτα ο δεύτερος στόχος είναι η υλοποίηση της μονάδας που ελέγχει την συμπεριφορά τον ανόδων της πλακέτας και είναι ουσιαστικά υπεύθυνη για την προβολή των χαρακτήρων στην οθόνη (σε αυτό το στάδιο προβάλετε ένα μικρό μήνυμα τεσσάρων χαρακτήρων). Ο τρίτος στόχος είναι η προβολή ενός μεγαλύτερου μηνύματος περιστρέφοντάς το μέσο ενός κουμπιού, και τέταρτο η αυτόματη περιστροφή του κάθε 1,67 δευτερόλεπτα.

4) Μέρος Α - Υλοποίηση Αποκωδικοποιητή 7-τμημάτων:

Υλοποίηση*:* Σε αυτό μέρος κατασκευάστηκε η μονάδα LED decoder όπου έχει ως είσοδο ένα χαρακτήρα τεσσάρων bit και έξοδο ένα σήμα LED με εφτά bits. Ο λόγος που οι χαρακτήρες αναπαρίστανται από τέσσερα bits είναι επειδή θα υπάρχουν συνολικά 16 επιλογές. Αυτό το κύκλωμα είναι συνδυαστικό.
Μονάδες: LED decoder

Επαλήθευση*:* Η επαλήθευση έγινε με ένα απλό testbench οπού ένα for loop άλλαζε κάθε 20ns έναν 4-bit counter προσομοιώνοντας όλους τους πιθανούς χαρακτήρες



Πείραμα: Δεν ήταν δυνατό κάποιο πείραμα στην πλακέτα.

Μέρος Β - Οδήγηση Τεσσάρων Ψηφίων:

Υλοποίηση*:* Αρχικά κατασκευάστηκε η μονάδα clock div που πολλαπλασιάζει την περίοδο του ρολογιού της πλακέτας κατά 20 φορές. Παρόλο που για έναν άνθρωπο οι χαρακτήρες στην οθόνη φαίνονται στατικοί στην πραγματικότητα αναβοσβήνουν ο ένας μετά τον άλλον αρκετά γρηγορά ώστε το ανθρώπινο μάτι να μην το αντιλαμβάνεται. Αυτή η διαδικασία ελέγχεται από το ρολόι της πλακέτας όμως το ρολόι αυτό είναι πολύ γρήγορο και το κύκλωμα υπεύθυνο για την ενδείξεις της οθόνης δεν θα προλαβαίνει να φορτίσει και να αποφορτίσει τους πυκνωτές του αναμεσά από το ανάμα και το σβήσιμο των λυχνιών. Για αυτόν τον λόγο δημιουργήθηκε αυτή η μονάδα που διαιρεί τα 100MHz(10ns) της πλακέτας σε 5MHz(0.2μs). Έπειτα κατασκευάστηκε ένας 4 bit counter όπου αλλάζει με κάθε κύκλο του αργού ρολογιού. Αυτός ελέγχει την επόμενη μονάδα Anode driver όπου είναι υπεύθυνη για την ενεργοποίηση των ανόδων-ψηφίων στην οθόνη. Είναι ουσιαστικά ένας πολυπλέκτης που κάθε τέσσερις εναλλαγές του μετρητή ανάβει και διαφορετική άνοδο. Αυτή η μέθοδος επιλέχτηκε ώστε να μην γίνεται ghosting κάθε φορά που αλλάζει η άνοδος και να προλαβαίνει το σήμα του χαρακτήρα να αλλάζει από τον παλιό στον καινούργιο. Σε αυτό το σημείο της εργασίας προβάλλεται το μήνυμα 0123 στην πλακέτα και αποθηκεύεται στην μονάδα Anode driver σε επόμενα μέρη θα υπάρχει ειδική μονάδα αποθήκευσης μεγαλύτερου μνήματος.

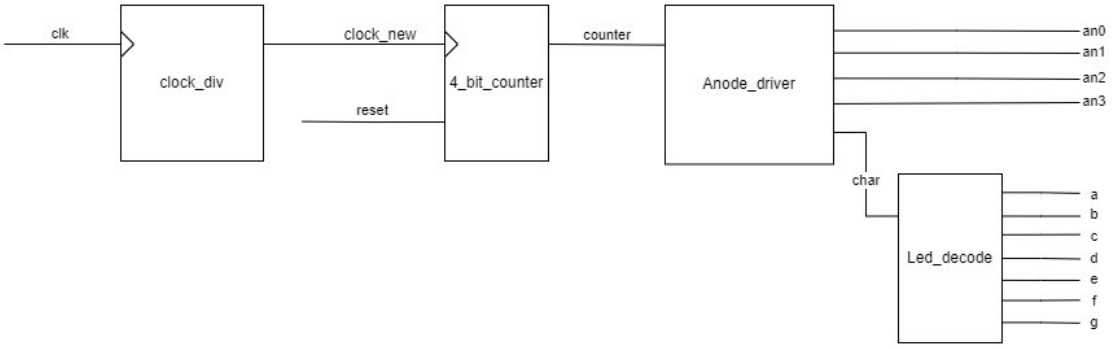
Μονάδες:

Clock_div: Διαιρεί το ρολόι 20 φορές.

Anode_driver: Ακολουθιακό κύκλωμα που έχει ως είσοδο έναν μετρητή και ως έξοδο τα σήματα τον ανόδων και τον χαρακτήρα που πρέπει να αποκωδικοποιηθεί.

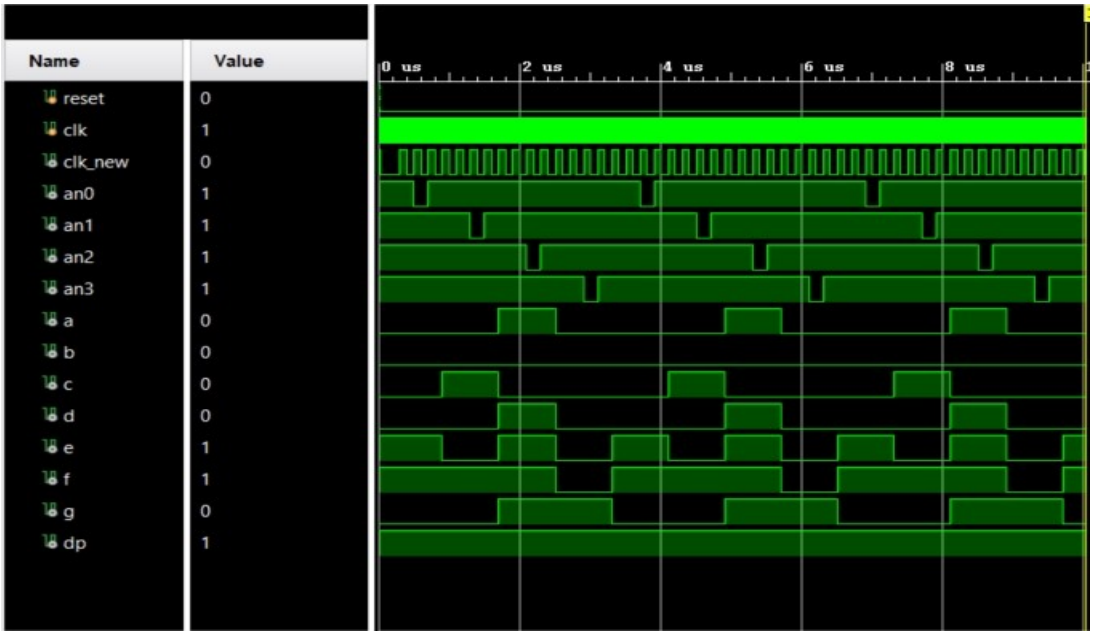
LED decoder: Η μονάδα που αναφέρθηκε στο προηγούμενο μέρος και τώρα έχει είσοδο την έξοδο του Anode driver.

Four_bit_counter: Ένας απλός μετρητής (ακολουθιακό) τεσσάρων bit που ελέγχει την Anode driver.

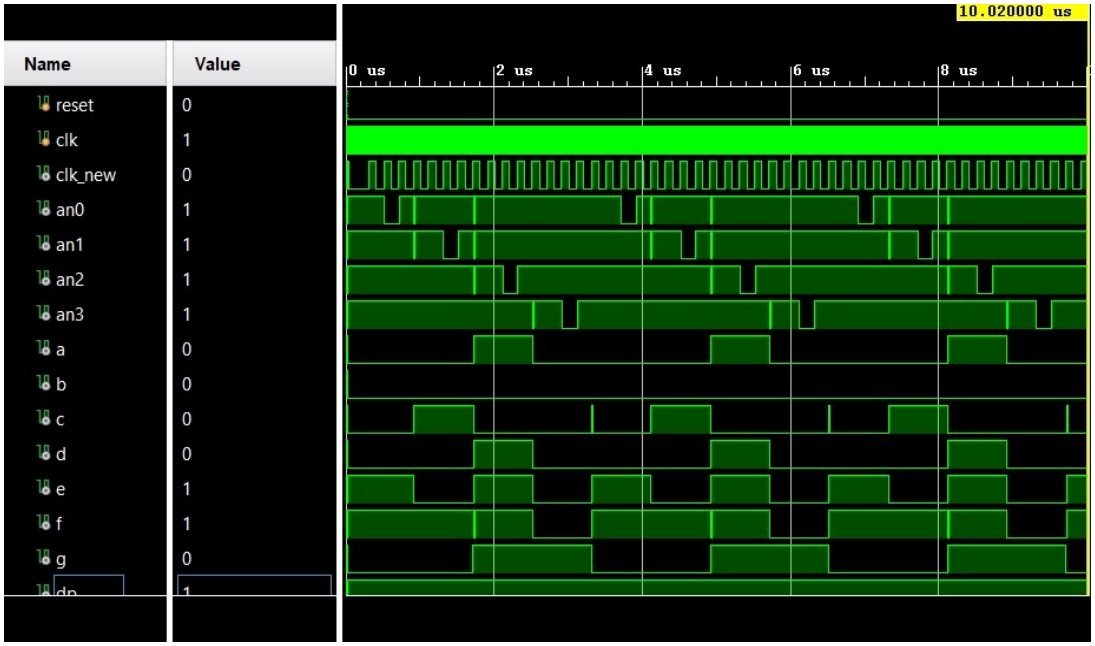


Επαλήθευση: Για την επαλήθευση χρησιμοποιήθηκε ένα testbench όπου αρχικοποιεί το clock και το reset και έπειτα εναλλάσσει το ρολόι μέχρι το κύκλωμα να έχει περάσει από όλες τις πιθανές να καταστάσεις του δηλαδή να έχουν ανάψει και να έχουν δείξει τον αριθμό τους όλες οι άνοδοι από μία φορά.

Behavioral



Post implementation timing sim



Εδώ φαίνεται ότι έχει περάσει από όλες τις φάσεις τέσσερίς φορές και μέσο τον σημάτων a->g μπορούμε να επαληθεύσουμε ότι όντως οι άνοδοι προβάλουν τα ψηφία 0123. Επίσης στο Behavioral και Post implementation το simulation έχει ιδιά αποτελέσματα εκτός από κάποια spikes που μάλλον πύλες που δεν δέχονται όλες τις εισόδους τους ταυτόχρονα, παρόλα αυτά τα spikes κρατάνε μόνο λίγα ps και δεν επηρεάζουν το αποτέλεσμα στην πλακέτα.

Πείραμα: Για το πείραμα στην πλακέτα χρειάστηκε να δοθούν οι περιορισμοί στο περιβάλλον του vivado και δεν υπήρχε κάποια δυσκολία διότι το πρόγραμμα ήταν απλό και το test κάλυπτε όλες τις περιπτώσεις.

Μέρος Γ - Βηματική Περιστροφή του Μηνύματος με χρήση Κουμπιού:

Υλοποίηση*:* Αρχικά αφού το μήνυμα σε αυτό το μέρος της εργασίας είναι μεγαλύτερο έπρεπε να υλοποιηθεί μια μονάδα message init που αρχικοποιεί την μεταβλητή reg στην οποία αποθηκεύεται αυτό. Έπειτα κατασκευάστηκε η μονάδα text scroll όπου η οποία είναι ένας πολυπλέκτης ελεγχόμενος από έναν μετρητή* και με κάθε αλλαγή αυτού μετακινεί ανά μία διεύθυνση τους δείκτες που δείχνουν στους τέσσερεις χαρακτήρες. Επίσης είναι απαραίτητο να γίνουν οι μονάδες που συγχρονίζουν το σήμα του κουμπιού με το ρολόι και που το κάνουν debounce, sync button και debounce αντίστοιχα. Η sync button ουσιαστικά αποτελείται από δυο flip-flops που συγχρονίζουν το καινούργιο σήμα με το ρολόι και επίσης δείχνει με μία έξοδο αυτή την

αλλαγή για έναν κύκλο. Για το debounce θα υπάρξει πιο αναλυτική εξήγηση αργότερα. Τέλος υλοποιήθηκε ένας ακόμη μετρητής* όπου μετράει κάθε φόρα που η μονάδα debounce του δίνει σήμα ότι το κουμπί πατήθηκε και χρησιμοποιείται στο text scroll όπως αναφέρθηκε. Οι μονάδες από το δεύτερο μέρος είναι ίδιες και χρησιμοποιούνται και σε αυτό.

Νέες μονάδες:

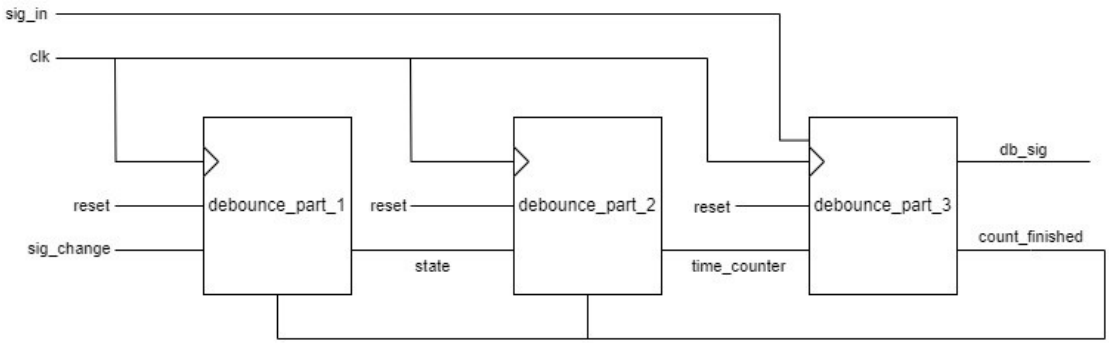
Message init: Ακολουθιακό κύκλωμα που αρχικοποιεί την μνήμη του μηνύματος και έχει ως έξοδο το μήνυμα αυτό.

Text scroll: Συνδυαστικό κύκλωμα που με την αλλαγή του μετρητή μετακινεί τους δείκτες των χαρακτήρων που πρέπει να φανούν στην οθόνη και τους περνάει σαν έξοδο.

Sync button: Ακολουθιακό κύκλωμα που συγχρονίζει το σήμα του κουμπιού με το ρολόι έχει ως είσοδο αυτό και έξοδο το συγχρονισμένο σήμα και ένα που δείχνει ότι πατήθηκε το κουμπί.

4_bit_counter_button: Μετρητής που μετράει κάθε φορά που το σήμα του κουμπιού ενεργοποιείται.

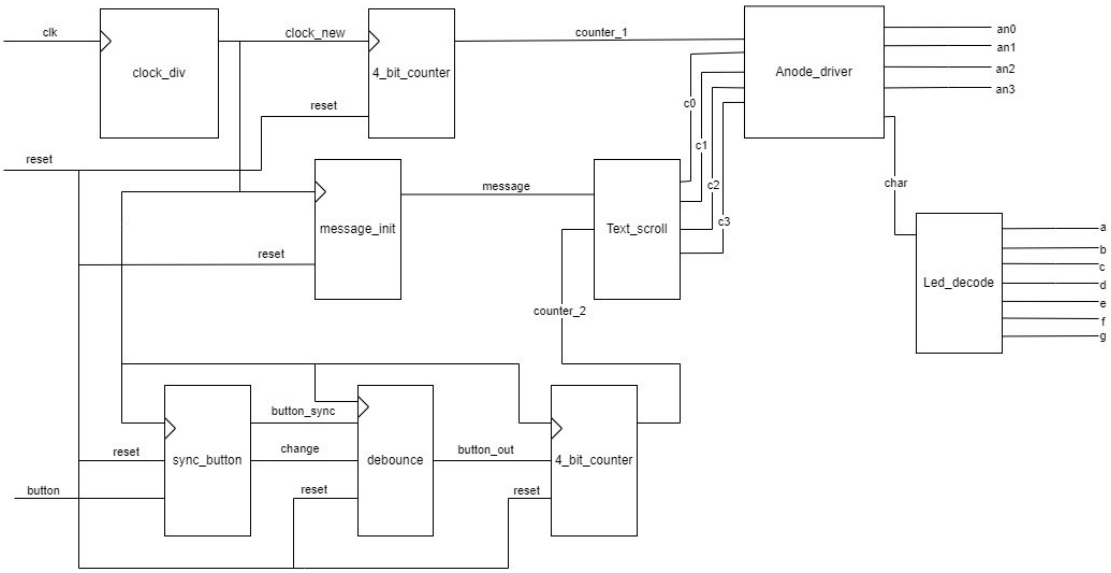
Debounce:



Ουσιαστικά αυτή η μονάδα όταν ενεργοποιείται το σήμα αλλαγής από το sync button (sig_change) αλλάζει την μεταβλητή state σε 1 και όσο αυτή είναι 1 ξεκινάει ένας μετρητής που δημιουργεί μία καθυστέρηση μέχρι το σήμα να σταματήσει να αναπηδάει και έπειτα στο part 3 παίρνει το σήμα sig_in από το sync button, το περνάει στην έξοδο και με το σήμα count_finished κάνει reset τον μετρητή και το state. Και τα τρία μέρη είναι ακολουθιακά.

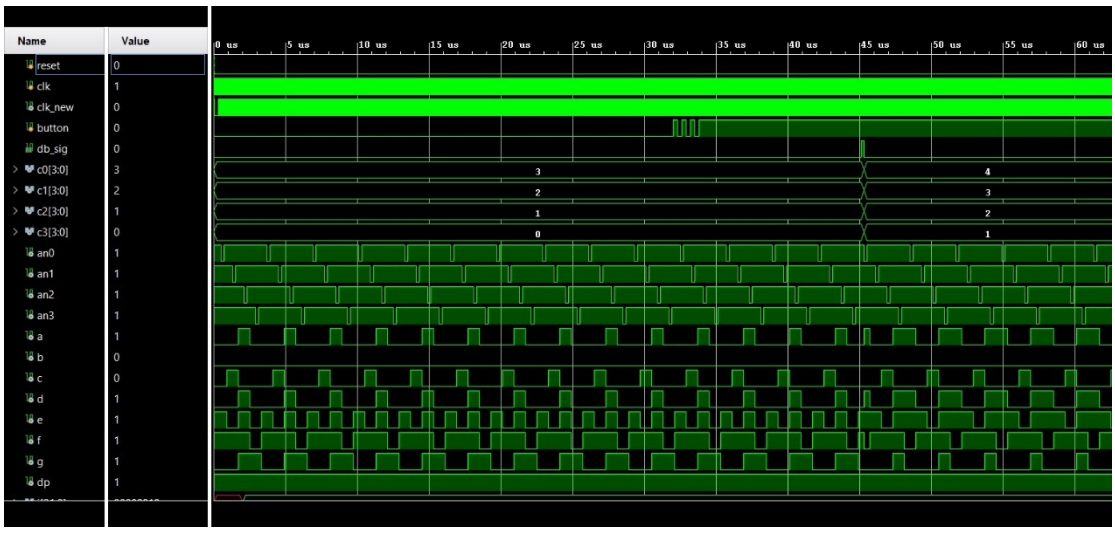
Anode driver: Αυτή η μονάδα παραμένει ίδια απλά τώρα αντί να αποθηκεύει τους χαρακτήρες, τους δέχεται σαν είσοδο από το text_scroll.

Συνολικό κύκλωμα:

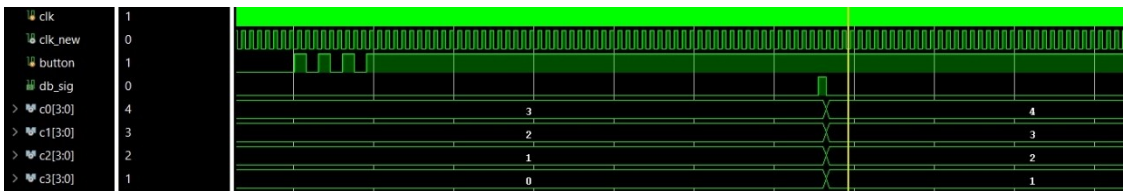


Επαλήθευση: Για την επαλήθευση χρησιμοποιήθηκε ένα testbench όπου αρχικοποιεί το clock και το reset και έπειτα εναλλάσσει το ρολόι. Επίσης προσομοιώνει το κουμπί αλλά και την αναπήδηση αυτού. Σε αυτό το τεστ θα φανεί η σωστή λειτουργία του debounce και η σωστή περιστροφή των χαρακτήρων του μηνύματος.

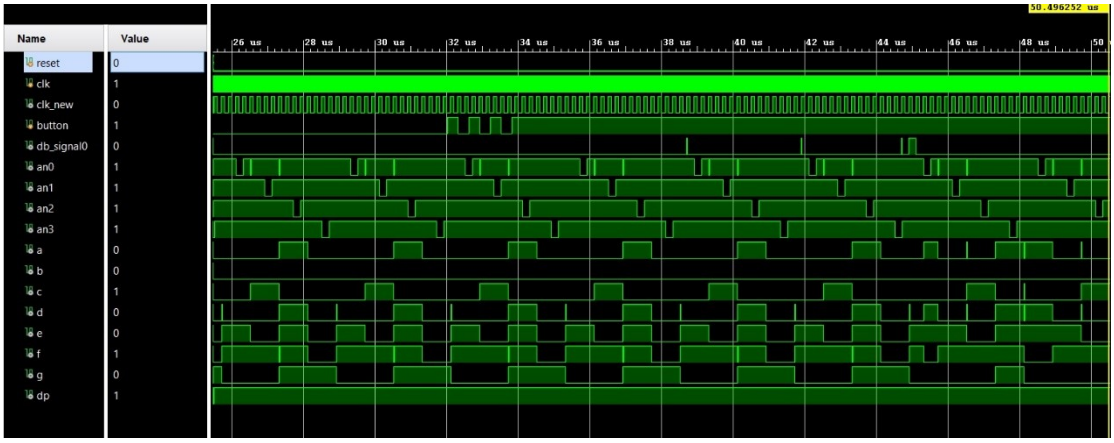
Behavioral



Debounce



Post implementation timing sim



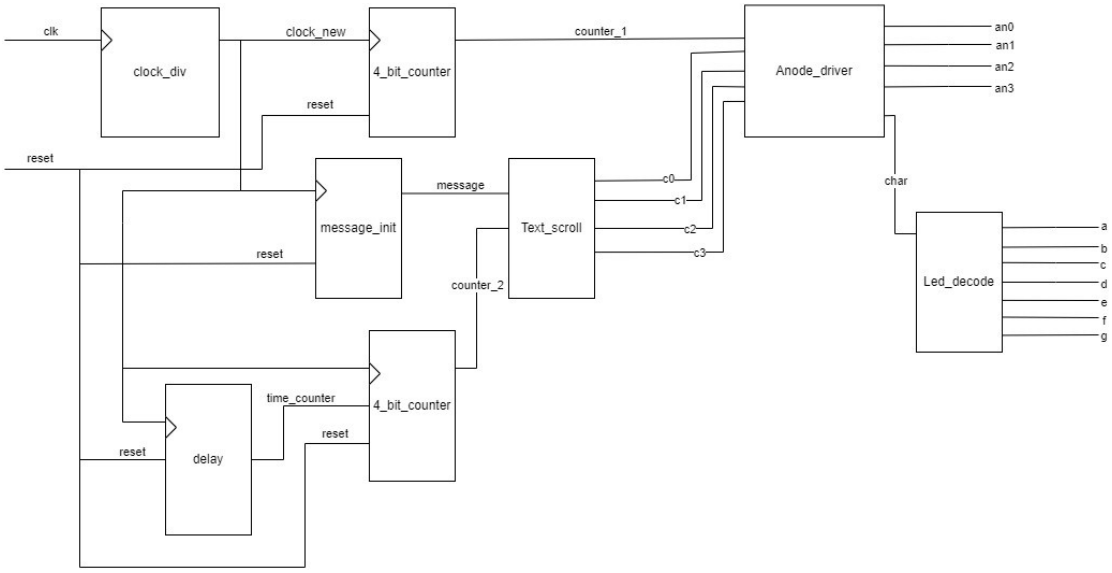
Σε αυτά τα τεστ φαίνεται την σωστή λειτουργία του debounce και η σωστή αλλαγή των χαρακτήρων προβολής στην πλακέτα κάθε φορά που το debounced σήμα ενεργοποιείται. Στο behavioral και post implementation βλέπουμε ακριβώς τα ίδια αποτελέσματα εκτός από κάποια spikes διάρκειας ελάχιστον ps και για κάποιο λόγο οι χαρακτήρες c0->c3 δεν είναι διαθέσιμοι για προβολή στις κυματομορφές στο post implementation. Παρόλα αυτά μπορούμε να δούμε τις κυματομορφές την σημάτων a->g που είναι ίδιες στο behavioral και post implementation.

Πείραμα: Στο πείραμα τα κύρια σημεία που έπρεπε να υπάρξει προσοχή ήταν, όταν το κουμπί για την αλλαγή χαρακτήρων πατιόταν γρήγορα δεν θα έπρεπε να υπάρχει skipping χαρακτήρων και επίσης όταν κρατιόταν πατημένο για πολύ ώρα τα μην άλλαζαν οι χαρακτήρες πάρα μόνο στην αρχή του πατήματος. Αφού αυτά δουλεύαν σημαίνει ότι η μονάδα debounce δουλευθεί σωστά.

Μέρος Δ - Βηματική Περιστροφή του Μηνύματος με σταθερή Καθυστέρηση:

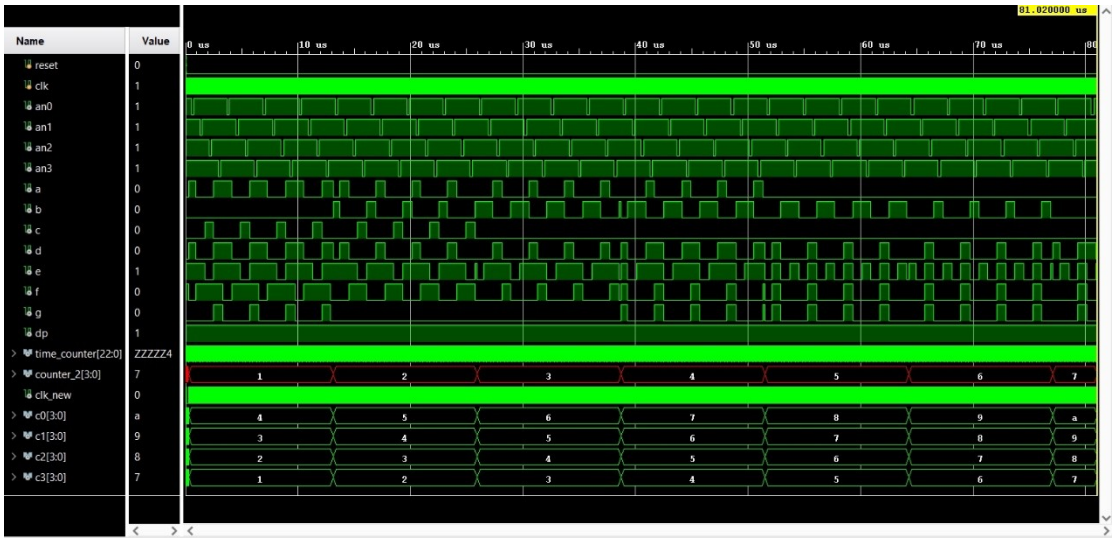
Υλοποίηση: Σε αυτό το μέρος η μόνη αλλαγή που έγινε ήταν η αντικατάσταση της εισόδου button, της μονάδας sync_button και του debounce με έναν μετρητή μεγέθους 23 bit που δημιουργεί καθυστέρηση 1,67 δευτερολέπτων με έπειτα ενεργοποιεί το scrolling των χαρακτήρων.
Νέες μονάδες:
Delay: Ακολουθιακό κύκλωμα με εισόδους το clk και reset οπού έχει ως έξοδο το time_counter το οποίο γίνεται μηδέν κάθε 1.67 δευτερόλεπτα.
Four_bit_time_counter: Μετρητής που αυξάνεται κάθε φορά που το time_count γίνεται μηδέν.

Συνολικό κύκλωμα:

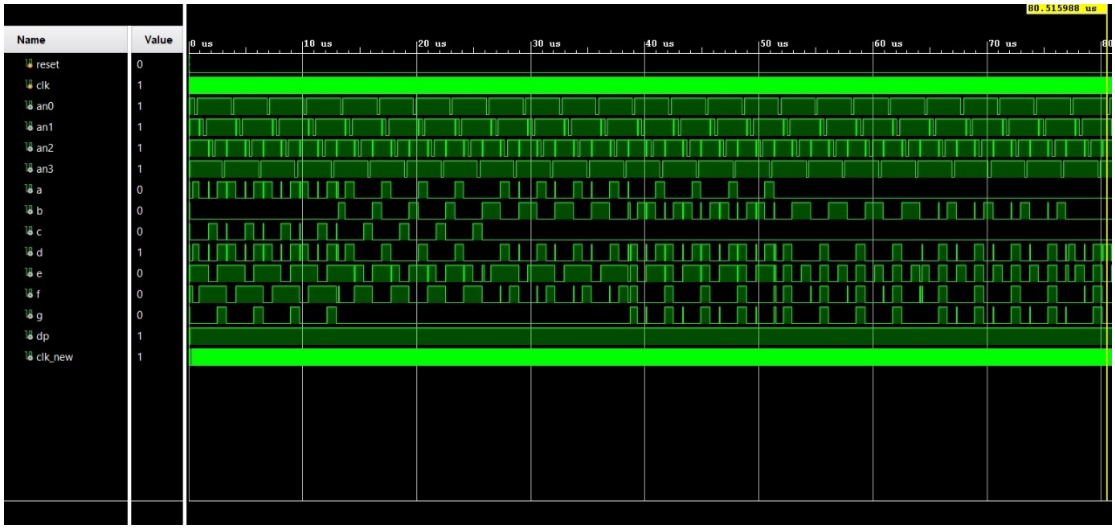


Επαλήθευση: Η επαλήθευση γίνεται με ένα testbench όπου απλά αλλάζει το clk μέχρι η μονάδα να έχει περάσει από όλες τις καταστάσεις της. Για το τεστ ο χρόνος αλλαγής 1.67 δευτερολέπτων γίνεται πολύ μικρότερος ώστε να τρέχει προσομοίωση σε λογικούς χρόνους.

Behavioral



Implementation



Πείραμα: Αυτό το πείραμα δεν χρειάστηκε ιδιαίτερη προσοχή εφόσον το προηγούμενο στο τρίτο μέρος δούλεψε κανονικά, το μόνο που χρειάστηκε να ελεγχθεί ήταν ο χρόνος αλλαγής των χαρακτήρων.

Συμπεράσματα: Αυτή η εργασία ήταν μία καλή εισαγωγή στην Verilog και στο Vivado, δεν ήταν πολύ απαιτητική όμως χρειάστηκε αρκετή προσοχή στα latch και στην μονάδα debounce. Τέλος το γεγονός ότι μπορούσαμε να τρέξουμε το τεστ μας στην πλακέτα βοήθησε πολύ στην κατανόηση όλων όσων υλοποιήθηκαν σε αυτή την εργασία.