

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

TMMY

7-Segment Display

LAB1

Μαρέλας Γιώργος

A.E.M 378

Πρόλογος

Αντικείμενο της εργαστηριακής εργασίας είναι η υλοποίηση ενός οδηγού των τεσσάρων ενδείξεων 7-τμημάτων LED στην FPGA πλακέτα Spartan 3. Η υλοποίηση είναι χωρισμένοι σε 4.

Στο πρώτο δημιουργήσαμε έναν αποκωδικοποιητή ώστε να οδηγούμε τα 7-τμήματα σύμφωνα με τον χαρακτήρα που θέλουμε να εμφανίσουμε. Στο δεύτερο μέρος οδηγούμε 4 ανόδους στην οθόνη 4 τμημάτων της Spartan 3 το οποίο επιτυγχάνεται με την γρήγορη εναλλαγή τους. Στο τρίτο μέρος με το πάτημα ενός κουμπιού περιστρέφουμε ένα μήνυμα 16 χαρακτήρων και στο τέταρτο και τελευταίο μέρος περιστρέφουμε το ίδιο μήνυμα με σταθερό ρυθμό 1,3421 δευτερόλεπτου.

Οι παραπάνω ενέργειες έχουν υλοποιηθεί με γλώσσα περιγραφής υλικού Verilog.

Εισαγωγή

Η εργασία έχει ως στόχο να δείξουμε πώς να χρησιμοποιούμε την οθόνη της πλακέτας με 3 διαφορετικούς τρόπους. Ο πρώτος τρόπος είναι εμφανίζοντας ένα σταθερό μήνυμα. Επειτα εναλλάσσοντας το μήνυμα με τη χρήση ενός κουμπιού. Η εργασία ολοκληρώνεται με την χρήση σταθερού ρυθμού για την εναλλαγή του μηνύματος της οθόνης.

Για την υλοποίηση της εργασίας χρειάστηκε να δημιουργήσουμε 5 module. Το αρχικό κύκλωμα που δημιουργήσαμε και αφορά το πρώτο μέρος της εργασίας είναι ο Decoder_7_Segment . Δέχεται ως είσοδο ένα σήμα 4-bits και το αποκωδικοποιεί σε ένα σήμα 7-bits. Στο δεύτερο μέρος υλοποιήσαμε το FOURDIGITDRIVER. Το συγκεκριμένο κύκλωμα στέλνει σε συγκεκριμένα χρονικά διαστήματα σταθερές τιμές στην 4 τμημάτων οθόνη έτσι ώστε το μήνυμα να φαίνεται σταθερό στο ανθρώπινο μάτι. Στο τρίτο μέρος ξανά χρησιμοποιήσαμε το FOURDIGITDRIVER αλλά προσθέσαμε την χρήση ενός κουμπιού και ενός μηνύματος το οποίο αλλάζει κάθε φορά που θα πιέζετε το κουμπί. Το FOURDIGITDRIVER το χρησιμοποιήσαμε και στο τελευταίο μέρος της εργασίας. Οι αλλαγές που έγιναν είναι η αφαίρεση του κουμπιού και το μήνυμα αυτή τη φορά αλλάζει με την βοήθεια ενός μετρητή 22-bits.

Επιπλέον έχει υλοποιηθεί το Debounce module το οποίο εμφανίζεται στα 3 παραπάνω τμήματα. Το Debounce είναι υπεύθυνο ώστε να μην υπάρχουν τυχών αστάθειες στα σήματα που ενεργοποιούνται με την χρήση κουμπιών. Το PulseButton module δημιουργήθηκε για τις ανάγκες του Γ μέρους του project και είναι υπεύθυνο για την εναλλαγή των χαρακτήρων στην οθόνη κατά μία φορά ακόμα και όταν το κουμπί παραμένει πατημένο παρατεταμένα.

Το DelayRotate module εμφανίζεται μόνο στο τελευταίο μέρος. Περιέχει τον μετρητή των 22bits με τον οποίο δημιουργούμε έναν τετραγωνικό παλμό και χρησιμοποιείτε στο τέταρτο μέρος για την περιστροφή του μηνύματος.

Τα modules που έχουν υλοποιηθεί είναι τα εξής:

-Για το πρώτο μέρος:

Decoder_7_Display με το Decoder_7_Segment_tb

-Για το δεύτερο μέρος (με σειρά ιεραρχίας):

i)FOURDIGITDRIVER ii)Decoder_7_Segment iii) Debounce

καθώς και τα αντίστοιχα test_bench για την επαλήθευση της ορθής λειτουργίας του κυκλώματος:

i)FOURDIGITDRIVER_tb ii)Decoder_7_Segment_tb

-Για το τρίτο μέρος :

i)FOURDIGITDRIVER ii)Decoder_7_Segment

iii)Debounce iv)ButtonPulse

με τα αντίστοιχα test-bench αρχεία

i)FOURDIGITDRIVER_tb ii)Decoder_7_Segment_tb

-Για το τέταρτο μέρος:

i)FOURDIGITDRIVER ii)Decoder_7_Segment

iii)Debounce iv)DelayRotate

με τα αντίστοιχα test-bench αρχεία

i)FOURDIGITDRIVER_tb ii)Decoder_7_Segment_tb

Το ucf αρχείο παραμένει ίδιο για το δεύτερο και τέταρτο μέρος ενώ στο τρίτο προσθέτουμε το κουμπί στο L13 pin.

Κατάλογος σημάτων

<u>Όνομασία</u>	<u>Πηγή</u>	<u>Περιγραφή</u>
Clk(input)	Πηγή ρολογιού	Χρονίζει όλες τις μονάδες του κυκλώματος
Rst(input)	Ελεγκτής Επανεκκίσης	Επανεκκινεί το κύκλωμα με τιμή 1
Button1(input)	Ελεγκτής αύξησης ψηφίου	Αυξάνει το ψηφίο της ένδειξης με την τιμή 1
An0,an1,an2,an3(output)	FOURDIGITDRIVER	Οδηγεί την κατάλληλη άνοδο με την τιμή 0
a,b,c,d,e,f,g,dp(output)	FOURDIGITDRIVER	Οδηγεί τις ενδείξεις των 7 τμημάτων
char(input)	Decoder_7_Segment	Είσοδος για την αποκωδικοποιήση τοθ σήματος
LED[6:0](output)	Decoder_7_Segment	Στέλνει την αποκωδικοποιημένη τιμή σύμφωνα με την τιμή του char
button(input)	Debounce	Είσοδος στο κύκλωμα Debounce
Debounce(output)	Debounce	Φιλτραρισμένη έξοδος

		του σήματος button
pulse(output)	ButtonPulse	Έξοδος σωστού παλμού για εναλλαγή χαρακτήρα
delayclk(output)	DelayRotate	Έξοδος του ρολογιού με καθυστέρηση

Μέρος Α

Υλοποίηση Αποκωδικοποιητή 7 τμημάτων

Υλοποίηση: Σε αυτό το μέρος υλοποιήθηκε το Decoder_7_Segment module. Το κύκλωμα παίρνει ως είσοδο ένα 4-bits ψηφίο ή χαρακτήρα και το αποκωδικοποιεί με την χρήση ενός 4μux7 σε 7-bits. Στην ουσία ο 4-bits αριθμός είναι ο χαρακτήρα που θέλουμε να εμφανίσουμε στην οθόνη και μετατρέπετε στο 7-bits σήμα το οποίο συσχετίζεται με 7 τμήματα οδήγησης κάθε οθόνης. Τα τμήματα της οθόνης ενεργοποιούνται όταν η τιμή του ανάλογου σήματος είναι 0 και ενεργοποιείται στην τιμή 1.

Για να εμφανίσουμε π.χ. τον χαρακτήρα “a” στην οθόνη πρέπει τα 4-bits εισόδου στο αποκωδικοποιητή να είναι $char[3:0] = 4'b1010$ το οποίο αποκωδικοποιείτε σε $LED[6:0]=7'b0001000$. Τα 7 bits αφορούν τα σήματα a-g . (Τα αποτελέσματα της κωδικοποίησης υλοποιήθηκαν σύμφωνα με το εγχειρίδιο της Spartan 3.)

Επαλήθευση: Για την επαλήθευση δημιουργήθηκε ένα test_bench στο οποίο δώσαμε όλες τις πιθανές τιμές σύμφωνα με τον πίνακα αλήθειας από το εγχειρίδιο της Spartan3. Παρακάτω φαίνεται και wave form της προσομοίωσης.

/Τελική υλοποίηση: Δεν δοκιμάστηκε στην πλακέτα FPGA.

Μέρος Β : Οδήγηση Τεσσάρων Ψηφίων

Υλοποίηση:Σε αυτό το μέρος ξεκινάει η υλοποίηση και η εφαρμογή του κυκλώματος στην πλακέτα.Στόχος είναι να εμφανίζεται ένα σταθερό μήνυμα στην οθόνη της πλακέτας. Η επιτύχia του βασίζεται στην σωστή καθυστέρηση εναλλαγής των σημάτων ανόδου.Για να πραγματοποιηθεί αυτό πολλαπλασιάσαμε την περιόδο του ρολογιού της πλακέτας κατά 16 φορές,ώστε οι εναλλαγές στην οθόνη να μην γίνονται αντιληπτές από το ανθρώπινο μάτι. Η οδήγηση των σημάτων είναι ορισμένη ανά κύκλο,βασισμένη στο διαιρεμένο ρολόϊ. Επομένως το FOURDIGITDRIVE περιλαμβάνει το τμήμα του κώδικα που διαιρεί το ρολόϊ, παίρνει ως είσοδο 2 σήματα, το clk και το rst και δίνει ως έξοδο τα σήματα an0,an1,an2,an3 και a, b, c, d, e, f, g, dp που σχετίζονται με τις ανόδους και τις ενδείξεις αντίστοιχα. Επίσης περιλαμβάνει 2 always block .

Το πρώτο always δημιουργεί έναν αντίστροφο μετρητή 4-bits με αρχική τιμή 4'b1111 στον οποίο αφαιρείτε 1 σε κάθε κύκλο ρολογιού.Αποφεύγοντας έτσι το να επικαλύπτονται οι άνοδοι και να αλλοιώνετε το αποτέλεσμα της ένδειξης. Το δεύτερο always block ενεργοποιείται καθώς αλλάζει τιμή του αντίστροφου μετρητή.Αποφασίζεται ποιά άνοδος θα είναι ενεργοποιημένη καθώς και ποιά θα είναι η ένδειξη της οθόνης. Σημαντικό είναι να σημειωθεί ότι το περιεχόμενο της ένδειξης της οθόνης αλλάζει δύο βήματα πριν την ενεργοποίηση της. Για παράδειγμα, το δεύτερο τμήμα της οθόνης ενεργοποιείται (an2=0) όταν η τιμή του μετρητή είναι 4'b1010, ενώ τα αποτελέσματα των ενδείξεων για το συγκεκριμένη άνοδο θα ενεργοποιηθούν όταν ο μετρητής θα έχει την τιμή 4'b1100 . Έτσι επιτυγχάνονται καθαρές ενδείξεις στην οθόνη.

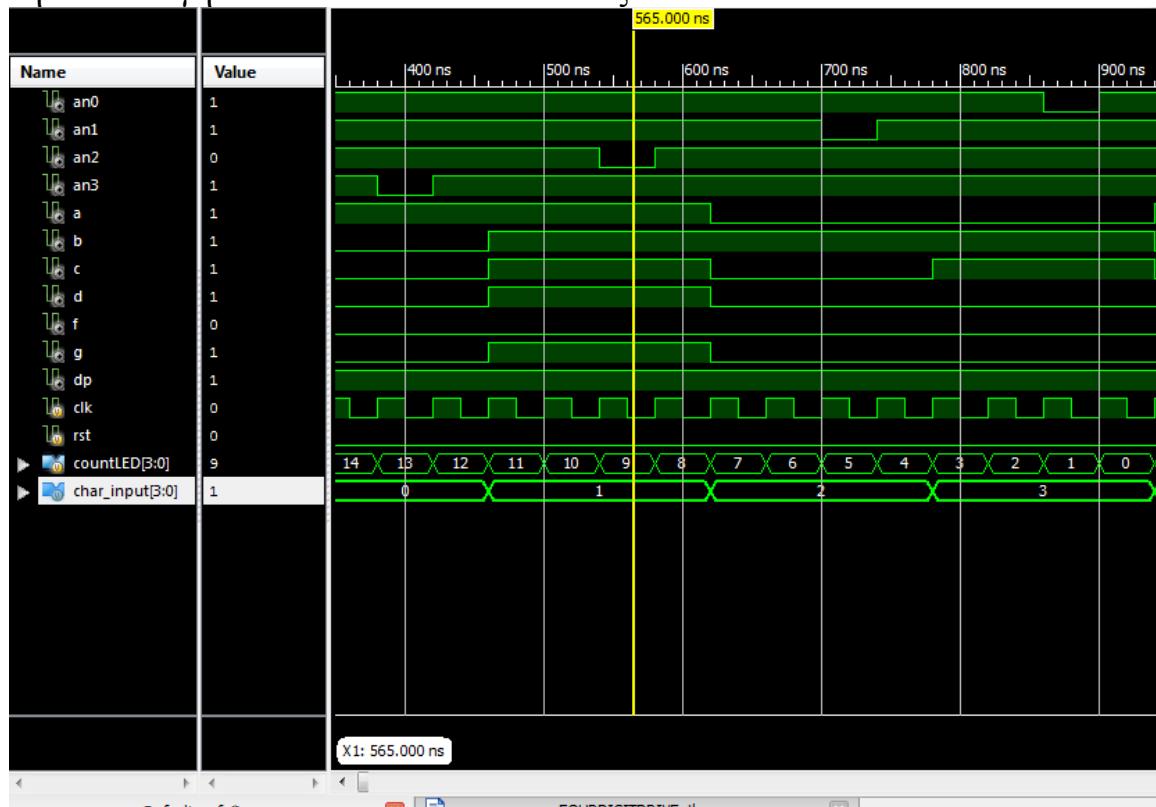
Στον κώδικα του FOURDIGITDRIVER έχουμε και τις εμφανίσεις του Decoder_7_Segment που υλοποιήσαμε στο πρώτο μέρος και το Debounce στο οποίο οδηγούμε το reset.

Στο Decoder_7_Segment οδηγείτε το σήμα που περιέχει τον χαρακτήρα που θέλουμε να εμφανίσουμε και περνούμε τα σήματα που θα οδηγηθούν στα σήματα a-g.

Το Debounce έχει υλοποιηθεί για να προλαμβάνει πιθανές αστάθειες και αναπηδήσεις σημάτων που προκύπτουν από την χρήση του κουμπιού της πλακέτας. Οι είσοδοι του είναι το clk, το σήμα που θέλουμε να διορθώσουμε(button) και η έξοδος του module είναι το σωστό

σήμα(debounce). Αυτό επιτυγχαίνεται με τη χρήση 2 always block. Σο πρώτο φλιτράρουμε το σήμα που πήραμε ως είσοδο. Γίνεται με την χρήση 2 flip-flop για τυχών αστάθειες και στο δεύτερο δειγματοληπτούμε το φιλτραρισμένο σήμα για να επιστρέψουμε πίσω στο FOURDIGITDRIVER το σωστό σήμα χωρίς θορύβους, πράγμα που θα επηρέαζε τη σωστή λειτουργία του κυκλώματος. Η δειγματοληψία γίνεται με την χρήση ενός καταχωρητή των 20-bits. Κάθε φορά που το ρολόι έρχεται στην θετική του ακμή, το σήμα ολισθαίνει κατά μία θέση στον 20-bits καταχωρητή. Όταν ο καταχωρητής γεμίσει από 20 άσους τότε το αποτέλεσμα της εξόδου θα είναι 1 και 0 όταν γεμίσει με μηδενικά, σε διαφορετική περίπτωση θα κρατάει την προηγούμενη τιμή του.

Επαλήθευση: Για την επαλήθευση της σωστής λειτουργίας του κυκλώματος υλοποιήθηκε test_bench FOURDIGITDRIVE_tb. Το test_bench θέτει τη χρονική στιγμή 0 τη τιμή του ρολογιού στο 0 και του rst με 1. Μετά από 200ns καθυστέρηση αλλάζουμε την τιμή του rst σε 0 και το ρολόι το εναλλάσσουμε με 20ns περίοδο. Τα αποτελέσματα της προσομοίωσης φαίνονται στην παρακάτω εικόνα όπου μπορούμε να δολθεμε την εναλλαγή των ανόδων και των ενδείξεων.



Πειράματα/Τελική υλοποίηση:

Προγραμματίζοντας την πλακέτα με το κατάλληλο bit file το πρόβλημα που παρατίθησα ήταν ότι δε μου εμφάνιζε σωστές τιμές στην οθόνη. Το πρόβλημα οφειλόταν στο γεγονός ότι δεν είχα δώσει σταθερές τιμές (αλλά μεταβλητές σύμφωνα με το ρολόι) κατά την άνοδο των ενδείξεων των 7 τυμημάτων. Με ανάθεση σταθερών τιμών το πρόβλημα διορθώθηκε.

Μέρος Γ: Βηματική Περιστροφή του Μηνύματος με χρήση Κουμπιού

Υλοποίηση:

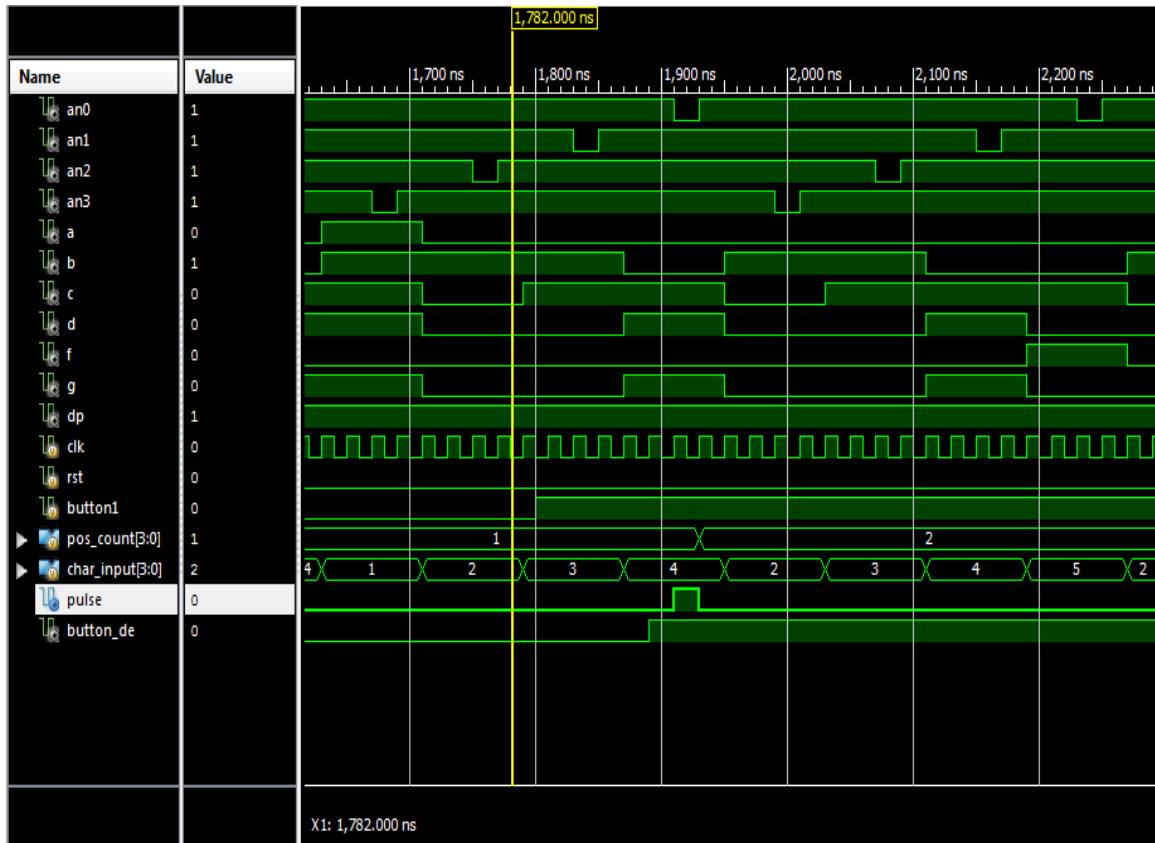
_Για την υλοποίηση του 3ου μέρους διατηρήσαμε την δομή του FOURDIFITDRIVER αλλά σε αυτή την περίπτωση προσθέσαμε μία μνήμη 16 θέσεων των 4-bits η καθεμία.Η αρχικοποίηση της μνήμης έγινε σε ένα always block με το επιθυμητό μήνυμα. Επίσης προσθέσαμε ως είσοδο ένα σήμα το οποίο συνδέετε με το κουμπί ώστε με κάθε πάτημα του να μετακινείτε το μήνυμα κατά 1 θέση. Για την περιστροφή του μηνύματος με το πάτημα του κουμπιού, δημιουργήσαμε ένα ακόμα always block, στο οποίο κάθε φορά που το κουμπί πατιέται, η τιμή ενός μετρητή αυξάνεται κατά 1. Ο μετρητής λειτουργεί ως δείκτης στη μνήμη. Το περιεχόμενο της μνήμης στην θέση που δείχνει ο μετρητής οδηγείτε στο σήμα εισόδου του Decoder_7_Segment.

Κρίθηκε απαραίτητη η δημιουργία ενός module επιπλέον,του ButtonPulse στο οποίο υλοποιούμε μια FSM. Παίρνοντας ως εισόδους το clk,rst,button και δίνοντας ως έξοδο το pulse,δημιουργεί αμοιβαίο αποκλεισμό μεταξύ κουμπιού και μετρητή ώστε να μην αυξάνεται ο μετρητής όσο το κουμπί είναι πατημένο και εξασφαλίζουμε ότι η εναλλαγή του μηνύματος στην οθόνη θα γίνει στην άνοδο του σήματος του κουμπιού και όχι στην κάθοδο.

Επαλήθευση

_Για την επαλήθευση της σωστής λειτουργίας του κυκλώματος χρησιμοποιήσαμε το ίδιο test_bench με το δεύτερο μέρος με μερικές μικρό αλλαγές. Η αλλαγή που έγινε είναι ότι προσθέσαμε τη χρονική στιγμή 0 το

`button1 = 0` και μετά από 500ns την θέτουμε στο 1 και την κρατάμε για άλλα 500ns. Την παραπάνω εναλλαγή την κάναμε ακόμη μια φορά μετά από 500ns. Η λειτουργία του κυκλώματος επαληθεύεται στο παρακάτω wave form της προσομοίωσης. Παρατηρούμε τη στιγμή που αφήσουμε το κουμπί η τιμή του σήματος `char_input`(σήμα που οδηγείτε στο Decoder_7_Segment και έχει την τιμή του χαρακτήρα) αυξάνετε κατά 1.



Πειράματα/Τελική υλοποίηση

Κατά την τελική δοκιμή στην πλακέτα παρατηρήθηκαν 2 προβλήματα:

Το πρώτο ήταν η αλλαγή των χαρακτήρων της οθόνης LED στην κάθοδο του σήματος του κουμπιού και η συνεχόμενη εναλλαγή τους όταν το κουμπί παρέμενε πατημένο. Διορθώθηκε με την χρήση του επιπλέον module ButtonPulse όπως ανέφερα πιο πάνω.

Το δεύτερο είναι πως οι χαρακτήρες που εμφανίζονται στην οθόνη της πλακέτας δεν ειναι οι επιθυμητοί και οι περισσότεροι δεν περιλβάνονται στην αρχικοποίηση που δόθηκε στην μνήμη. Το πρόβλημα μέχρι στιγμής δεν

έχει επιλυθεί καθώς μετά τις 30/10 δεν το δοκίμασα ξανά στην πλακέτα.
Παρόλα αυτά στο simulation όπως φαίνεται όλα λειτουργούν σωστά.

Μέρος Δ: Βηματική Περιστροφή του Μηνύματος με σταθερή Καθυστέρηση

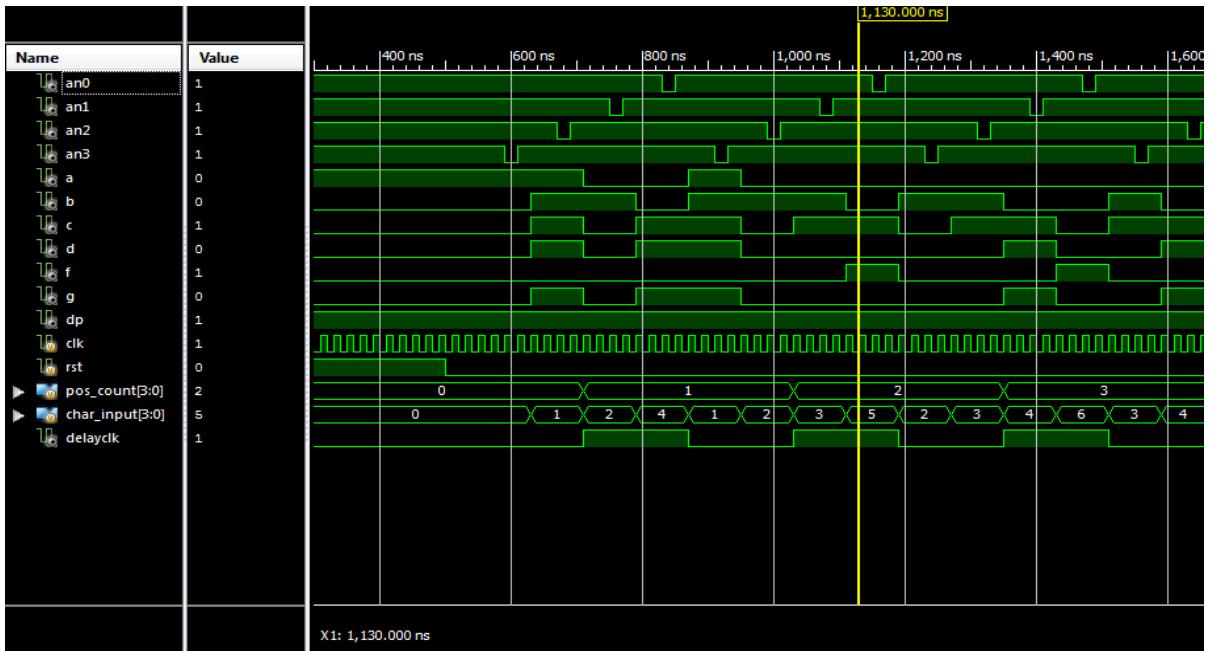
Υλοποίηση:

Στο τέταρτο και τελευταίο μέρος χρησιμοποιήσαμε την υλοποίηση του τρίτου μέρους(εκτός του επιπλέον module ButtonPulse). Άλλαγές έγιναν στο always block που περιελάμβανε την αύξηση του μετρητή με το πάτημα του κουμπιού. Το κουμπί αφαιρέθηκε ο μετρητής δείκτης της μνήμης παρέμεινε και το port map του always block άλλαξε και αντικαταστήσαμε το clk με ένα μειωμένο ρολόι, περιόδου 1,3421 δευτερολέπτων. Έτσι κάθε φορά που το μειωμένο ρολόι βρίσκεται στην θετική του ακμή ο μετρητής αυξάνεται κατά 1.

Για να μειώσουμε το ρολόι δημιουργήσαμε το DelayRotate. Το module παίρνει ως είσοδο το κανονικό μας ρολόι και επιστρέφει πίσω έναν σήμα με τη μορφή τετραγωνικού παλμού με περίοδο 1,3421 δευτερόλεπτα. Αυτό επιτυγχάνεται με την χρήση ενός μεγάλου μετρητή των 22-bits. Σε κάθε θετική ακμή του ρολογιού ο μετρητής αυξάνεται κατά ένα και στέλνει πίσω την τιμή του πιο σημαντικού bit. Επομένως, για 0,67105 δευτερόλεπτα το σήμα θα έχει την τιμή 0 και για τα υπόλοιπα την τιμή 1. Έτσι δημιουργήσαμε ένα πιο αργό ρολόι.

Επαλήθευση:

Για την επαλήθευση χρησιμοποιήσαμε το test_bench του δεύτερου μέρους αυτούσιο. Άλλαγή κάναμε στον κώδικα του DelayRotate μειώνοντας τα bits του μετρητή από 22-bits σε 4 για να χρειαστεί λιγότερος χρόνος τρεξίματος κατά την προσομοίωση. Μπορούμε να παρατηρήσουμε την αλλαγή των χαρακτήρων προς εμφάνιση.



Πειράματα/Τελική υλοποίηση :

Κατά την τελική δοκιμή στην πλακέτα δεν παρατηρήθηκε κανένα πρόβλημα και η εναλλαγή των χαρακτήρων με καθυστέρηση λειτουργεί κανονικά.

ΔΙΑΣΥΝΔΕΣΗ ΤΩΝ ΜΟΝΑΔΩΝ

