PROJECT1 REPORT

1. **ΤΙΤΛΟΣ**

7-Segment Display Driver

ΓΑΡΟΥΦΑΛΗ ΜΑΡΙΑ ΝΕΦΕΛΗ, ΑΕΜ: 3129, 11/5/22

1. **ΠΕΡΙΛΗΨΗ**

Στην συγκεκριμένη αναφορά παρουσιάζεται η 1η εργαστηριακή εργασία του μαθήματος.

Η εργασία αυτή αφορά στην κατασκευή ενός οδηγού τεσσάρων ενδείξεων 7-τμημάτων. Η υλοποίησή της έγινε σε στάδια, τα οποία θα αναφερθούν αναλυτικά. Κάθε στάδιο περιγράφεται από την Υλοποίηση, την Επαλήθευση και το Πείραμα.

1. **ΕΙΣΑΓΩΓΗ**

Ο στόχος της 1ης εργαστηριακής εργασίας είναι η υλοποίηση ενός οδηγού των τεσσάρων ενδείξεων 7-τμημάτων LED της πλακέτας Nexys A7-100T, προκειμένου να επιτευχθεί η περιστροφική παρουσίαση ενός μηνύματος μεγέθους 16 χαρακτήρων. Το μήνυμα θα παρουσιαστεί μετατοπίζοντας τους χαρακτήρες, έναν προς έναν, κατάλληλα προς τα αριστερά, είτε:

• (1) με το πάτημα ενός κουμπιού, είτε

• (2) μετά από ένα δεδομένο χρονικό διάστημα.

Μετά τον τελευταίο χαρακτήρα του μηνύματος, θα ακολουθεί ο πρώτος, έτσι το μήνυμα ουσιαστικά θα περιστρέφεται διαρκώς.

Ο στόχος της περιστροφής του μηνύματος με το πάτημα κουμπιού επιτεύχθηκε με την υλοποίηση ενός μετρητή που αυξάνεται με το πάτημα ενός κουμπιού. Η κατάσταση στην οποία βρίσκεται ο συγκεκριμένος μετρητής καθορίζει το μήνυμα που θα παρουσιαστεί από τα ψηφία της πλακέτας. Ένας επιπλέον μετρητής εξασφαλίζει την περιοδική εναλλαγή ενεργούς κατάστασης των ψηφίων. Ένας αποκωδικοποιητής παίρνει ως είσοδο τις παραπάνω πληροφορίες και αποτυπώνει στην έξοδο τον χαρακτήρα που πρέπει να απεικονιστεί, καθώς και ποιο ψηφίο είναι ενεργό την συγκεκριμένη χρονική στιγμή.

Ο στόχος της περιστροφής του μηνύματος μετά από δεδομένο χρονικό διάστημα επιτεύχθηκε με την χρήση ενός 23bit μετρητή. Ο συγκεκριμένος μετρητής εξασφαλίζει πως μεταξύ διαδοχικών αρχικών καταστάσεων (23’b0 (t0) -> 23’b0 (t1)) θα έχει περάσει χρόνος ίσος περίπου με 1 δευτερόλεπτο. Με είσοδο έναν παλμό κάθε φορά που περνάει το περίπου 1 δευτερόλεπτο, ένας μετρητής δημιουργεί 16 καταστάσεις. Η κατάσταση στην οποία βρίσκεται ο συγκεκριμένος μετρητής καθορίζει το μήνυμα που θα παρουσιαστεί από τα ψηφία της πλακέτας. Ένας επιπλέον μετρητής εξασφαλίζει την περιοδική εναλλαγή ενεργούς κατάστασης των ψηφίων. Ένας αποκωδικοποιητής παίρνει ως είσοδο τις παραπάνω πληροφορίες και αποτυπώνει στην έξοδο τον χαρακτήρα που πρέπει να απεικονιστεί, καθώς και ποιο ψηφίο είναι ενεργό την συγκεκριμένη χρονική στιγμή.

1. **Μέρος Α - Υλοποίηση Αποκωδικοποιητή 7-τμημάτων**

* **ΥΛΟΠΟΙΗΣΗ**

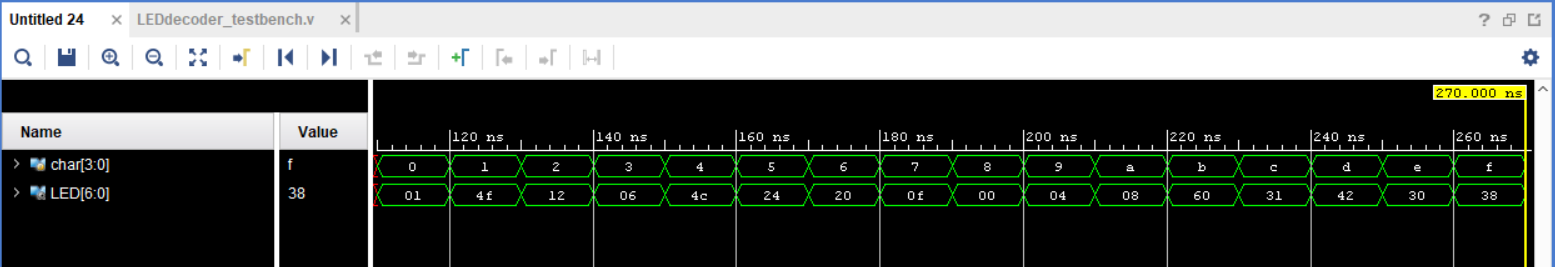
Σε αυτό το μέρος υλοποιήθηκε ένας αποκωδικοποιητής, ο οποίος παίρνει ως είσοδο έναν 4bit χαρακτήρα και έχει ως έξοδο ένα 7bit σήμα LED. Χρησιμοποιούμε 4bit χαρακτήρα διότι θέλουμε να αναπαραστήσουμε 16 διαφορετικές τιμές [log2(16)=4]. Ο συγκεκριμένος αποκωδικοποιητής είναι ένα συνδυαστικό κύκλωμα, οπότε και στο τμήμα always χρησιμοποιούμε blocking assignments.

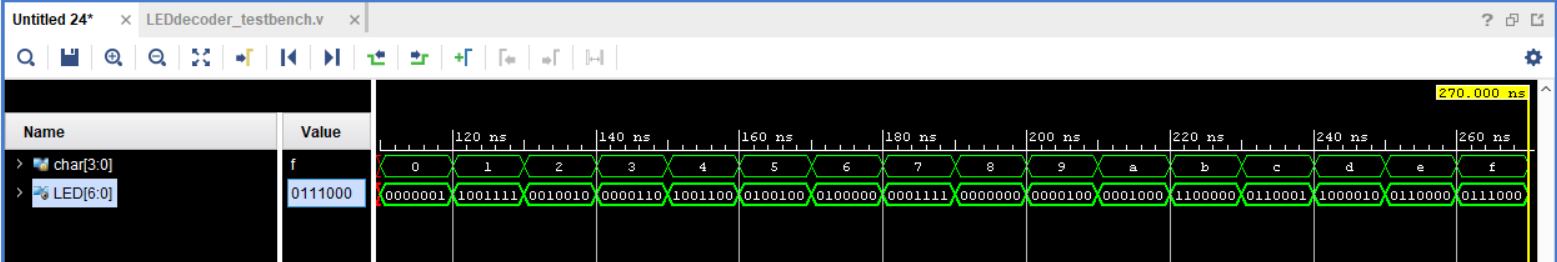
Μονάδες

**LEDdecoder**: Πρόκειται για έναν αποκωδικοποιητή. Είναι συνδυαστικό κύκλωμα και ως εκ τούτου χρησιμοποιήθηκαν blocking assignments.

* **ΕΠΑΛΗΘΕΥΣΗ**

Στο στάδιο της επαλήθευσης κατασκευάστηκε μια μονάδα ελέγχου (testbench). Σε αυτή χρησιμοποιήθηκε ένας 4bit χαρακτήρας εισόδου, ο οποίος με καθυστέρηση 10ns πήρε όλες τις τιμές που επιθυμούμε να αναπαραστήσουμε. Με τον τρόπο αυτό ελέγχθηκε ότι όλες οι τιμές απεικονίζονται σωστά με τα 7 τμήματα led. Το ποσοστό κάλυψης της μονάδας ελέγχου είναι 100%. Η μονάδα ελέγχου δεν κατέδειξε κανένα πρόβλημα στην υλοποίηση. Παρακάτω φαίνονται τα αποτελέσματα της προσομοίωσης σε 16αδική και σε δυαδική αναπαράσταση.





* **ΠΕΙΡΑΜΑ**

Το συγκεκριμένο μέρος δεν μπορεί να ελεγχθεί πειραματικά, αυτοτελώς.

**Μέρος Β – Οδήγηση Τεσσάρων Ψηφίων**

* **ΥΛΟΠΟΙΗΣΗ**

Στο στάδιο αυτό στόχος είναι να εμφανίζονται στην πλακέτα συνεχώς αναμμένα τα τέσσερα ψηφία με μια σταθερή διαφορετική τιμή το καθένα. Τα τέσσερα ψηφία μοιράζονται τα 7 συν 1 (δεκαδικό σημείο) σήματα A, B, C, D, E, F, G και DP. Για να ανάψει ένα συγκεκριμένο τμήμα LED ενός ψηφίου πρέπει:

• το σχετικό σήμα ανόδου, ΑΝ0-ΑΝ3, που επιλέγει το ψηφίο, να οδηγείται στο μηδέν (0),

• το σχετικό σήμα καθόδου τμήματος, CA-G/DP, που επιλέγει το τμήμα, να οδηγείται στο μηδέν (0).

Οι τέσσερις άνοδοι πρέπει να οδηγούνται εναλλάξ στο μηδέν, ενώ αλλάζουν κατάλληλα τα σήματα των 7 ενδείξεων και του δεκαδικού σημείου. Η διαδικασία πρέπει να επαναλαμβάνεται διαρκώς. Αν και τα LED ουσιαστικά αναβοσβήνουν, λόγω της γρήγορης εναλλαγής και της διατήρησης της φωτεινότητας τους, στο ανθρώπινο μάτι φαντάζουν σταθερά. Κατά την εναλλάξ διαδικασία σάρωσης, η άνοδος και τα σήματα των τμημάτων πρέπει να μένουν σταθερά για ένα σχετικά μεγάλο χρονικό διάστημα, ως προς το ρολόι της πλακέτας, λόγω της μεγάλης χωρητικότητας του κυκλώματος των LED. Ο ευκολότερος τρόπος οδήγησης των ανόδων στην ταχύτητα των 0.20μs είναι να πολλαπλασιαστεί η περίοδος του ρολογιού της FPGA (περιόδου 10ns) κατά είκοσι φορές, και κατόπιν η οδήγηση των σημάτων να είναι ορισμένη στη Verilog ανά κύκλο, βασισμένη στο νέο αργότερο ρολόι. Για τον πολλαπλασιασμό της περιόδου, χρησιμοποιήθηκε μια δομική μονάδα MMCM (Mixed-Mode Clock Manager). Τα σήματα των ανόδων δε θα πρέπει να επικαλύπτονται όσο βρίσκονται στο μηδέν, αλλιώς θα αλλοιωθούν οι ενδείξεις. Έτσι, απαιτείται και ένα περιθώριο ασφαλείας από την επιστροφή ενός σήματος ανόδου στο λογικό ένα, μέχρι την πτώση του επόμενου. Η μη-επικαλυπτόμενη οδήγηση των ανόδων επιτεύχθηκε με την χρήση ενός 4bit μετρητή (FourBit\_Counter), κάποιες καταστάσεις του οποίου ενεργοποιούν τα ψηφία ενώ, οι ενδιάμεσες τα απενεργοποιούν. Πρακτικά, ο μετρητής χρησιμοποιείται εμμέσως ως ΜΠΚ (Μηχανή Πεπερασμένων Καταστάσεων), όπου η μέτρηση του αντιστοιχεί πρακτικά σε μια κατάσταση. Στην συγκεκριμένη υλοποίηση συναντούμε και την μονάδα Digs\_Disp. Η συγκεκριμένη μονάδα αποτελεί ουσιαστικά συνέχεια του μετρητή. Πρόκειται ουσιαστικά για έναν αποκωδικοποιητή ο οποίος έχει ως είσοδο τις 16 διαφορετικές καταστάσεις του μετρητή και ως έξοδο παράγει κάθε στιγμή ποιο ψηφίο είναι αναμμένο καθώς και την τιμή που πρέπει να απεικονιστεί. Τέλος, ο LEDdecoder που υλοποιήθηκε στο προηγούμενο μέρος λαμβάνει ως είσοδο τον χαρακτήρα που πρέπει να απεικονιστεί και βγάζει ως έξοδο ποια από τα 7 Led τμήματα πρέπει να φωτιστούν για να παραχθεί ο συγκεκριμένος χαρακτήρας. Από τις παραπάνω μονάδες μόνο ο μετρητής είναι ακολουθιακό κύκλωμα. Για την αρχικοποίηση του χρησιμοποιήθηκε ασύγχρονο σήμα reset και non-blocking αναθέσεις. Αντίθετα, στα συνδυαστικά κυκλώματα χρησιμοποιήθηκαν blocking αναθέσεις.

Μονάδες

**FourBit\_Counter**: Πρόκειται για έναν μετρητή 16 καταστάσεων. Είναι ασύγχρονο ακολουθιακό κύκλωμα με reset. Ως ακολουθιακό κύκλωμα έχει non-blocking assignments. Χρησιμοποιείται για την εναλλάξ οδήγηση των ψηφίων και των τιμών τους.

**Digs\_Disp**: Πρόκειται για έναν αποκωδικοποιητή. Είναι συνδυαστικό κύκλωμα και ως εκ τούτου χρησιμοποιήθηκαν blocking assignments. Λαμβάνει ως είσοδο τις καταστάσεις του FourBit\_Counter και βγάζει στην έξοδο ποιο ψηφίο πρέπει να ανάψει, ποια πρέπει να είναι σβηστά αλλά και ποια τιμή πρέπει να απεικονιστεί.

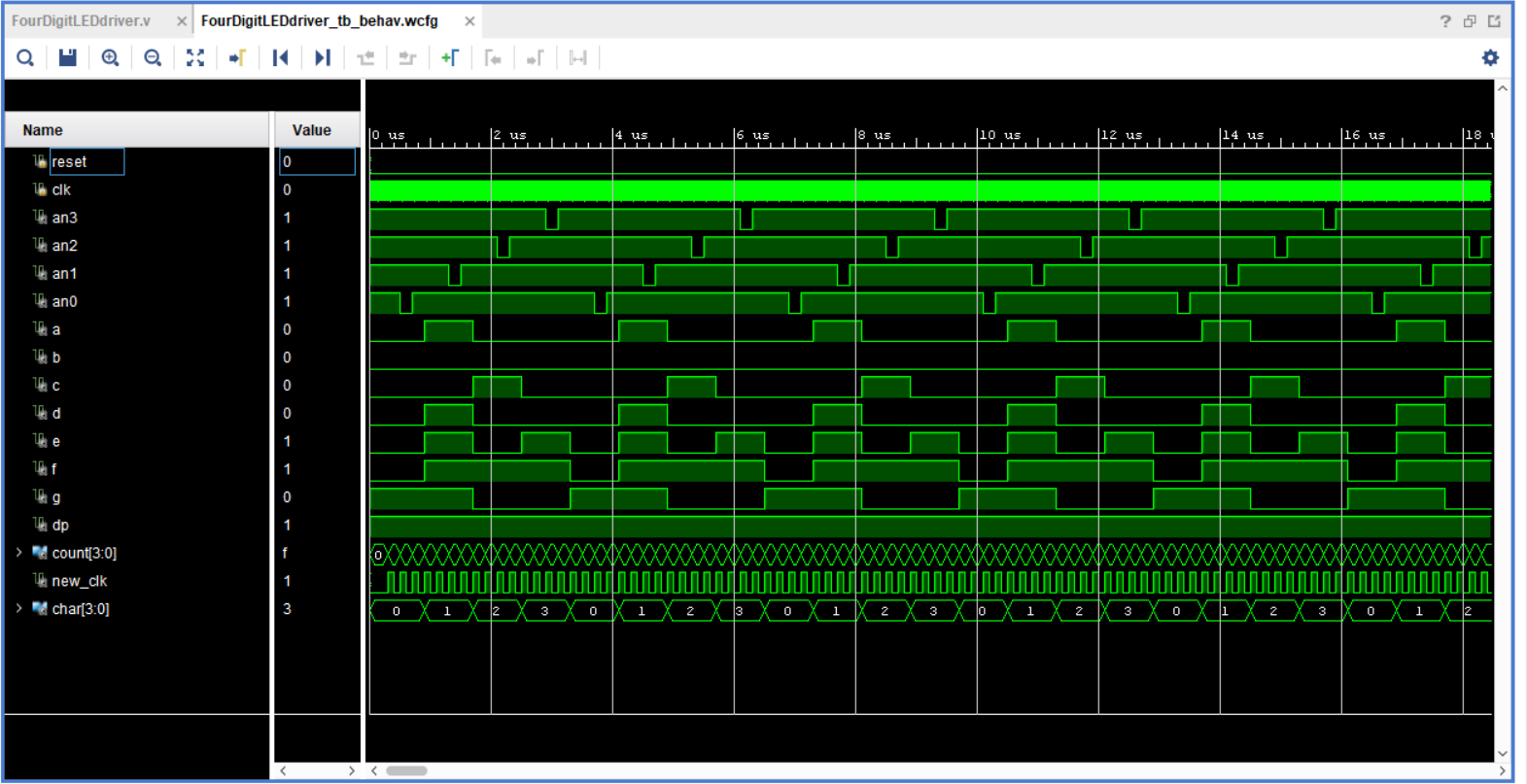
**LEDdecoder**: Πρόκειται για έναν αποκωδικοποιητή. Είναι συνδυαστικό κύκλωμα και ως εκ τούτου χρησιμοποιήθηκαν blocking assignments. Λαμβάνει ως είσοδο με 4 bit τον χαρακτήρα που θέλουμε να απεικονίσουμε και παράγει στην έξοδο με 7 bit τα σωστά φωτισμένα 7 τμήματα του 7segment display.

Εικόνα που περιέχει κείμενο, λευκός πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

* **ΕΠΑΛΗΘΕΥΣΗ**

Στο στάδιο της επαλήθευσης κατασκευάστηκε μια μονάδα ελέγχου (testbench). Σε αυτή ορίστηκε αρχικά το ρολόι με περίοδο σαν αυτή της πλακέτας και στην συνέχεια τέθηκε το reset κατάλληλα που ουσιαστικά αρχικοποιεί τον μετρητή. Έπειτα, έγινε η «τοποθέτηση» του module. Με τον τρόπο αυτόν ελέγχθηκε ότι στο πέρασμα του χρόνου, τα ψηφία γίνονται ενεργά εναλλάξ, καθώς και ότι απεικονίζονται διαφορετικές τιμές στο καθένα όταν αυτό είναι ενεργό. Ένα πρόβλημα που προέκυψε ήταν ο χρόνος της προσομοίωσης, ο οποίος αρχικά δεν επαρκούσε για να ελεγχθούν όλες οι περιπτώσεις. Για τον λόγο αυτόν, αυξήθηκε για να υπάρχει 100% κάλυψη. Προβλήματα στην υλοποίηση δεν αναδείχθηκαν. Το μήνυμα που επιλέχθηκε να εμφανίζεται είναι το 3 2 1 0 στις an3 an2 an1 an0 αντίστοιχα.



* **ΠΕΙΡΑΜΑ**

Το συγκεκριμένο μέρος ελέγχθηκε και μέσω της πειραματικής διαδικασίας. Συγκεκριμένα, προγραμματίστηκε η πλακέτα από το αρχείο bitstream που παρήγαγε ο κώδικας του συγκεκριμένου τμήματος. Αν και δεν υπήρχαν ηλεκτρικά προβλήματα, το αποτέλεσμα μετά την πρώτη προσπάθεια προγραμματισμού της πλακέτας δεν ήταν το επιθυμητό. Στην έξοδο δεν εμφανίζονταν οι χαρακτήρες που είχα προγραμματίσει να εμφανίζονται. Αφού αρχικά βεβαιώθηκα ότι οδηγώ σωστά τις καθόδους, παρατήρησα ότι κάποια από τα 7 τμήματα δεν φωτίζονταν καθόλου, άλλα ήταν πιο αχνά και άλλα πιο έντονα. Κοιτώντας ξανά την προσομοίωση παρατήρησα ότι στα πιο έντονα τμήματα, οι κάθοδοι ήταν περισσότερη ώρα στην συγκεκριμένη τιμή. Αντίθετα στα αχνά φωτισμένα τμήματα, οι κάθοδοι άλλαζαν συχνά τιμές, ενώ στα σβηστά τμήματα οι κάθοδοι ήταν μονίμως ανενεργές. Έτσι, σκέφτηκα ότι θα έπρεπε να μην αλλάζω τις τιμές των καθόδων μόνο για την κατάσταση που το συγκεκριμένο ψηφίο θα είναι αναμμένο, αλλά να τις αλλάζω δύο ή τρείς καταστάσεις νωρίτερα. Αφού έκανα αυτήν την αλλαγή, προγραμμάτισα ξανά την πλακέτα και το αποτέλεσμα αυτήν την φορά ήταν το αναμενόμενο.

**Μέρος Γ - Βηματική Περιστροφή του Μηνύματος με χρήση Κουμπιού**

* **ΥΛΟΠΟΙΗΣΗ**

Στο τρίτο μέρος έπρεπε να υλοποιηθεί η περιστροφή του μηνύματος στο πάτημα ενός κουμπιού. Η τιμή των τεσσάρων ψηφίων προς αποκωδικοποίηση στην συγκεκριμένη περίπτωση, θα πρέπει να είναι μεταβλητή και όχι σταθερή. Αυτό μπορεί να επιτευχθεί, χρησιμοποιώντας έναν μετρητή – δείκτη (FourBitEnableCounter). Ο μετρητής - δείκτης θα αυξάνεται στο πάτημα ενός κουμπιού της πλακέτας. Κάθε κατάσταση του μετρητή θα αντιστοιχεί σε διαφορετικές τιμές για τα τέσσερα ψηφία. Μέσω ενός αποκωδικοποιητή (Scrolling\_text) που ως είσοδο έχει τις καταστάσεις του μετρητή, παίρνουμε στην έξοδο την τιμή του κάθε ψηφίου για την συγκεκριμένη κατάσταση. Και σε αυτό το μέρος υπάρχει όπως και πριν η μονάδα MMCM, καθώς και ο μετρητής FourBit\_Counter που εξασφαλίζει την περιοδική ενεργή τιμή των ψηφίων. Επιπλέον, υπάρχει ένας αποκωδικοποιητής (Digit\_Disp) ο οποίος λαμβάνει ως είσοδο τις καταστάσεις που δημιουργεί ο FourBit\_Coutner και τις τιμές που πρέπει να αναπαραστήσει το κάθε ψηφίο στην συγκεκριμένη κατάσταση από τον Scrolling\_text. Με αυτές τις πληροφορίες αναπαριστά στην έξοδο ποιο ψηφίο είναι αναμμένο και ποια είναι η τιμή που πρέπει να απεικονιστεί τελικά. Η τιμή που βγαίνει από τον Digit\_Disp, όπως και πριν αποκωδικοποιείται στον LEDdecoder για να πάρουμε τελικά τα φωτισμένα τμήματα του 7 segment display. Επίσης, προστέθηκε στο αρχείο .xdc το pin στο οποίο πρέπει να συνδεθεί το σήμα enable που είναι ουσιαστικά το κουμπί. Τέλος, κατασκευάστηκε ένας debouncer για την καλύτερη λειτουργία του κουμπιού.

Μονάδες

**Debounce**: Η συγκεκριμένη μονάδα περιέχει όλο το debounce κύκλωμα του κουμπιού. Αποτελείται από 2 D Flip Flop και 1 πύλη AND. Ως είσοδο παίρνει το αργό ρολόι που παράχθηκε από τον MMCM και το σήμα που αντιστοιχεί στο κουμπί. Η έξοδος της συγκεκριμένης μονάδας είναι το νέο σήμα του κουμπιού.

**FourBitEnableCounter**: Πρόκειται για έναν μετρητή 4-bit ο οποίος αλλάζει στην θετική ακμή του ρολογιού την τιμή του, όποτε ο χρήστης πατήσει το κουμπί. Πρόκειται για ακολουθιακό ασύγχρονο κύκλωμα οπότε και χρησιμοποιήθηκαν non-blocking assignments.

**Scrolling\_text**: Πρόκειται για έναν αποκωδικοποιητή ο οποίος λαμβάνει ως είσοδο τις 16 καταστάσεις του μετρητή καθώς και το μήνυμα από την μνήμη. Στην έξοδο παράγει την τιμή κάθε ψηφίου για την κάθε κατάσταση εισόδου. Πρόκειται για συνδυαστικό κύκλωμα οπότε και χρησιμοποιήθηκαν blocking assignments.

**Digit\_Disp**: Πρόκειται επίσης για έναν αποκωδικοποιητή ο οποίος λαμβάνει ως είσοδο τις τιμές των 4 ψηφίων καθώς και την τιμή του FourBit\_Counter ο οποίος καθορίζει κάθε στιγμή ποιο ψηφίο πρέπει να ανάψει. Έτσι, στην έξοδο βγάζει ποιο ψηφίο είναι αναμμένο κάθε χρονική στιγμή καθώς και ποια τιμή πρέπει να απεικονιστεί τελικά. Επειδή είναι συνδυαστικό κύκλωμα έχει υλοποιηθεί με blocking assignments.

**LEDdecoder**: Πρόκειται για έναν αποκωδικοποιητή ο οποίος λαμβάνει με 4 bit τον χαρακτήρα που θέλουμε να απεικονίσουμε και παράγει στην έξοδο με 7 bit τα σωστά φωτισμένα 7 τμήματα του 7segment display. Επειδή είναι συνδυαστικό κύκλωμα έχει υλοποιηθεί με blocking assignments.

**FourBit\_Counter**: Πρόκειται για έναν μετρητή 16 καταστάσεων. Είναι ασύγχρονο ακολουθιακό κύκλωμα με reset. Ως ακολουθιακό κύκλωμα έχει non-blocking assignments. Χρησιμοποιείται για την εναλλάξ οδήγηση των ψηφίων και των τιμών τους.

**Memory\_init**: Η συγκεκριμένη μονάδα χρησιμοποιήθηκε για την αρχικοποίηση του μηνύματος message στην μνήμη. Πρόκειται για ένα ακολουθιακό ασύγχρονο κύκλωμα που λειτουργεί με το παλιό ρολόι, οπότε και χρησιμοποιήθηκαν non-blocking αναθέσεις για την υλοποίηση του.

Εικόνα που περιέχει κείμενο, λευκός πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

* **ΕΠΑΛΗΘΕΥΣΗ**

Για την επαλήθευση δημιουργήθηκε μια μονάδα ελέγχου. Αρχικά ορίσαμε το ρολόι, όμοια με αυτό της πλακέτας. Έπειτα, έγινε η «τοποθέτηση» του module. Στην συνέχεια χρησιμοποιήθηκε ένα σήμα enable το οποίο υλοποιεί την λειτουργία του κουμπιού. Η στρατηγική που ακολουθήθηκε ήταν το enable να ενεργοποιείται για μία περίοδο ρολογιού, προκειμένου να μεταφερθεί σίγουρα η αλλαγή και έπειτα να παραμένει ανενεργό για 3200ns όσο δηλαδή χρειάζεται για να ανάψουν όλα τα ψηφία της πλακέτας και να απεικονιστούν οι τιμές τους για την συγκεκριμένη κατάσταση που δημιούργησε το πάτημα του κουμπιού. Με τον τρόπο αυτόν ελέγχθηκαν όλες οι περιπτώσεις και η κάλυψη ήταν 100%. Και σε αυτήν την προσομοίωση χρειάστηκε να αυξήσω τον χρόνο. Στην προσομοίωση το σήμα enable\_in είναι το κουμπί μετά τον debouncer. Διαδοχικά φαίνεται πως με το πάτημα του κουμπιού αλλάζουν βηματικά οι έξοδοι του κάθε ψηφίου.

Εικόνα που περιέχει κείμενο, ηλεκτρονικές συσκευές

Περιγραφή που δημιουργήθηκε αυτόματα

* **ΠΕΙΡΑΜΑ**

Το συγκεκριμένο μέρος ελέγχθηκε και μέσω της πειραματικής διαδικασίας. Συγκεκριμένα, προγραμματίστηκε η πλακέτα από το αρχείο bitstream που παρήγαγε ο κώδικας του συγκεκριμένου τμήματος. Ηλεκτρικά προβλήματα δεν υπήρχαν και το κύκλωμα λειτούργησε κατευθείαν όπως αναμενόταν.

**Μέρος Δ - Βηματική Περιστροφή του Μηνύματος με σταθερή Καθυστέρηση**

* **ΥΛΟΠΟΙΗΣΗ**

Στο τέταρτο μέρος έπρεπε να αντικατασταθεί η λειτουργία του κουμπιού με μια σταθερή καθυστέρηση. Έτσι, κάθε 1 δευτερόλεπτο, ή λίγο περισσότερο, θα πρέπει να περιστρέφεται κατά ένα χαρακτήρα το μήνυμα. Η σταθερή καθυστέρηση μπορεί να υλοποιηθεί με έναν μετρητή κύκλων ρολογιού, ο μηδενισμός του οποίου θα επιδεικνύει το πέρας του χρόνου. Επιλέχθηκε μετρητής 23bit διότι 2^23 = 8388608. Εφόσον ο μετρητής αυτός λειτουργεί με τον παλμό του καινούριου αργού ρολογιού που έχει περίοδο 200ns ο χρόνος που θα περάσει θα είναι 200\*8388608 = 1,6777214 δευτερόλεπτα. Βάσει αυτού του μετρητή δημιουργήθηκε ένα σήμα click το οποίο ενεργοποιείται κάθε φορά που ο μετρητής παίρνει μια συγκεκριμένη τιμή και άρα περνάει αυτό το χρονικό διάστημα. Το σήμα click στην συνέχεια μπαίνει ως είσοδος σε έναν μετρητή. Κάθε φορά που γίνεται 1 ο μετρητής μεταβάλει την κατάσταση του. Με τον τρόπο αυτόν ουσιαστικά δημιουργούνται οι 16 διαφορετικές καταστάσεις που θέλουμε μετά από το πέρας 1 δευτερολέπτου. Οι 16 αυτές καταστάσεις αποτελούν είσοδο για έναν αποκωδικοποιητή, ο οποίος βάσει της τρέχουσας κατάστασης, βγάζει στην έξοδο τις τιμές που πρέπει να απεικονίσει το κάθε ψηφίο. Οι τιμές αυτές καθώς και η έξοδος του FourBit\_Counter εισέρχονται στον αποκωδικοποιητή Digit\_Disp, ο οποίος βγάζει στην έξοδο ποιο ψηφίο θα είναι αναμμένο, καθώς και ποιος χαρακτήρας θα πρέπει τελικά να αναπαρασταθεί. Η τιμή που βγαίνει από τον Digit\_Disp, όπως και πριν αποκωδικοποιείται στον LEDdecoder για να πάρουμε τελικά τα φωτισμένα τμήματα του 7 segment display.

Μονάδες

**Sec\_Counter**: Πρόκειται για έναν μετρητή 23-bit ο οποίος αλλάζει στην θετική ακμή του ρολογιού την τιμή του. Μπορεί να αναπαραστήσει 2^23 = 8388608 τιμές. Αν πολλαπλασιάσουμε τον αριθμό αυτόν με 200 ns που είναι η περίοδος του ρολογιού και η «καθυστέρηση» από κατάσταση σε κατάσταση του μετρητή, προκύπτει ότι μετά από έναν «κύκλο» του μετρητή θα έχουν περάσει 167772160 ns, ή 1,67 second. Αυτός είναι και ο λόγος που τον χρησιμοποιούμε στο κύκλωμα μας. Πρόκειται για ακολουθιακό ασύγχρονο κύκλωμα οπότε και χρησιμοποιήθηκαν non-blocking assignments.

**Scrolling\_text**: Πρόκειται για έναν αποκωδικοποιητή ο οποίος λαμβάνει ως είσοδο τις 16 καταστάσεις του μετρητή καθώς και το μήνυμα από την μνήμη. Στην έξοδο παράγει την τιμή κάθε ψηφίου για την κάθε κατάσταση εισόδου. Πρόκειται για συνδυαστικό κύκλωμα οπότε και χρησιμοποιήθηκαν blocking assignments.

**Digit\_Disp**: Πρόκειται επίσης για έναν αποκωδικοποιητή ο οποίος λαμβάνει ως είσοδο τις τιμές των 4 ψηφίων καθώς και την τιμή του FourBit\_Counter ο οποίος καθορίζει κάθε στιγμή ποιο ψηφίο πρέπει να ανάψει. Έτσι, στην έξοδο βγάζει ποιο ψηφίο είναι αναμμένο κάθε χρονική στιγμή καθώς και ποια τιμή πρέπει να απεικονιστεί τελικά. Επειδή είναι συνδυαστικό κύκλωμα έχει υλοποιηθεί με blocking assignments.

**LEDdecoder**: Πρόκειται για έναν αποκωδικοποιητή ο οποίος λαμβάνει με 4 bit τον χαρακτήρα που θέλουμε να απεικονίσουμε και παράγει στην έξοδο με 7 bit τα σωστά φωτισμένα 7 τμήματα του 7segment display. Επειδή είναι συνδυαστικό κύκλωμα έχει υλοποιηθεί με blocking assignments.

**FourBit\_Counter**: Πρόκειται για έναν μετρητή 16 καταστάσεων. Είναι ασύγχρονο ακολουθιακό κύκλωμα με reset. Ως ακολουθιακό κύκλωμα έχει non-blocking assignments. Χρησιμοποιείται για την εναλλάξ οδήγηση των ψηφίων και των τιμών τους.

**Count\_case**: Πρόκειται για έναν μετρητή ο οποίος παίρνει ως είσοδο την έξοδο του Sec\_Counter και παράγει 16 διαφορετικές καταστάσεις στην έξοδο, κάθε φορά που ο Sec\_Counter κάνει έναν «κύκλο» δηλαδή κάθε φορά που περνάει 1 περίπου δευτερόλεπτο. Είναι ένα ακολουθιακό κύκλωμα με ασύγχρονο reset και ως τέτοιο υλοποιήθηκε με non-blocking assignments.

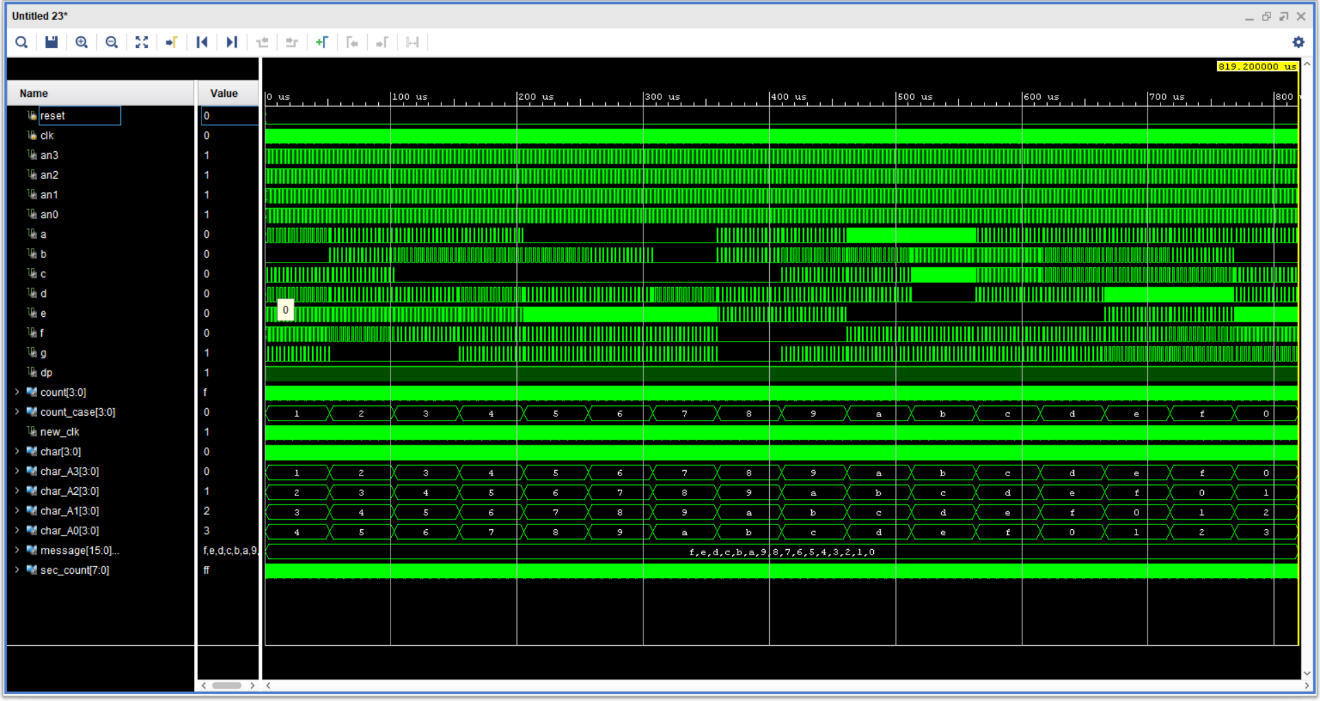
**Memory\_init**: Η συγκεκριμένη μονάδα χρησιμοποιήθηκε για την αρχικοποίηση του μηνύματος message στην μνήμη. Πρόκειται για ένα ακολουθιακό ασύγχρονο κύκλωμα που λειτουργεί με το παλιό ρολόι, οπότε και χρησιμοποιήθηκαν non-blocking αναθέσεις για την υλοποίηση του.

Εικόνα που περιέχει κείμενο, λευκός πίνακας

Περιγραφή που δημιουργήθηκε αυτόματα

* **ΕΠΑΛΗΘΕΥΣΗ**

Για την επαλήθευση δημιουργήθηκε μια μονάδα ελέγχου. Αρχικά ορίσαμε το ρολόι, όμοια με αυτό της πλακέτας. Έπειτα, έγινε η «τοποθέτηση» του module. Σε αυτή τη περίπτωση το σύστημα πρέπει να τρέχει ανεξάρτητα από κάποιο κουμπί. Εφόσον η μόνη εξάρτιση είναι ο χρόνος, η στρατηγική είναι να αρχικοποιήσουμε το κύκλωμα θέτοντας κατάλληλα το reset και να το αφήσουμε να τρέξει για επαρκές χρονικό διάστημα. Εδώ εμφανίστηκε ένα πρόβλημα. Η προσομοίωση για 20 seconds αργούσε υπερβολικά, οπότε προκειμένου να ελεγχθεί το κύκλωμα, άλλαξα την τιμή του μετρητή μειώνοντας ουσιαστικά το χρονικό διάστημα μεταξύ των αλλαγών. Η κάλυψη της συγκεκριμένης μονάδας ελέγχου είναι 100% αφού για το χρονικό διάστημα που επιλέχθηκε άλλαξαν όλες οι καταστάσεις.



* **ΠΕΙΡΑΜΑ**

Το συγκεκριμένο μέρος ελέγχθηκε και μέσω της πειραματικής διαδικασίας. Συγκεκριμένα, προγραμματίστηκε η πλακέτα από το αρχείο bitstream που παρήγαγε ο κώδικας του συγκεκριμένου τμήματος. Αν και δεν υπήρχαν ηλεκτρικά προβλήματα, το σύστημα δεν λειτούργησε όπως θα έπρεπε. Αντί να υπάρχει βηματική κύλιση των ψηφίων, δηλαδή (0 1 2 3 -> 1 2 3 4 -> 2 3 4 5) υπήρχε κύλιση των ψηφίων ανά δύο, δηλαδή (0 1 2 3 -> 2 3 4 5 -> 4 5 6 7). Αυτό το πρόβλημα δεν εμφανιζόταν στην επαλήθευση οπότε και μου πήρε κάποια ώρα να συνειδητοποιήσω τι έφταιγε. Κάνοντας τις απαραίτητες αλλάγες, περίμενα να δουλέψει αλλά και πάλι προέκυψαν προβλήματα. Πλέον το μήνυμα μετακινούνταν ανά 4 ψηφία δηλαδή (3 4 5 6 -> 7 8 9 a -> b c d e). Όντας σίγουρη για την ορθότητα της αλλαγής που έκανα έλεγξα τα instantiations των μονάδων και εντόπισα ένα λάθος σήμα στην κλήση μιας μονάδας. Αφού έκανα αυτήν την αλλαγή, προγραμμάτισα ξανά την πλακέτα και το αποτέλεσμα αυτήν την φορά ήταν το αναμενόμενο.

1. **ΣΥΜΠΕΡΑΣΜΑΤΑ**

Η υλοποίηση της 1ης εργαστηριακής εργασίας ήταν πολύ βοηθητική για την εισαγωγή στο περιβάλλον του Vivado και τον προγραμματισμό μιας FPGA σε Verilog. Ιδιαίτερα σημαντική είναι η καλή κατανόηση της εκφώνησης καθώς και η σχεδίαση διαγράμματος πριν την υλοποίηση του κώδικα. Όσον αφορά την υλοποίηση εφόσον είχα αποφασίσει τι έπρεπε να σχεδιάσω μελετούσα από την αντίστοιχη θεωρεία του μαθήματος. Παρόλα αυτά συναντήθηκαν κάποιες δυσκολίες στο πως για παράδειγμα θα περαστεί πίνακας σε module ή στην παραμετροποίηση του MMCM. Τα προβλήματα αυτά επιλύθηκαν με αναζήτηση στο διαδίκτυο και μελέτη των εγχειριδίων. Άλλα προβλήματα στην υλοποίηση ή λάθη γινόταν συνήθως κατευθείαν αντιληπτά από τα μηνύματα λάθους του Vivado. Το κομμάτι της επαλήθευσης ήταν αισθητά πιο εύκολο από αυτό της υλοποίησης. Η συγγραφή του testbench (αν και δεν το έκανα αυτόματο) μου φάνηκε σχετικά απλή και η μόνη μέριμνα μου ήταν να ελέγχω κάθε φορά πλήρως την υλοποίηση. Εξαιρώντας την διαίρεση της συχνότητας όπου η επαλήθευση έδειχνε συχνά λάθη στην υλοποίηση, όλες οι υπόλοιπες μονάδες ελέγχου δεν παρουσίασαν προβλήματα. Δυσκολίες συναντήθηκαν σε τεχνικά ζητήματα όπως το πώς θα προστεθούν επιμέρους σήματα στις κυματομορφές και πως θα αυξηθεί ο χρόνος επαλήθευσης. Και πάλι αυτά επιλύθηκαν με την βοήθεια του διαδικτύου και των βοηθών του εργαστηρίου. Η δοκιμή του συστήματος στην πλακέτα κάποιες φόρες κατέδειξε λάθη της υλοποίησης που δεν είχαν διαπιστωθεί στο στάδιο της επαλήθευσης. Αυτά τα λάθη ήταν και αυτά που πήραν περισσότερη ώρα στην διόρθωση γιατί ήταν πιο δύσκολη η ανίχνευσή τους. Η σωστή αντιμετώπιση ήταν η εστίαση στα modules εκείνα που σχετίζονταν με κάποιον τρόπο με το πρόβλημα που παρουσιαζόταν.