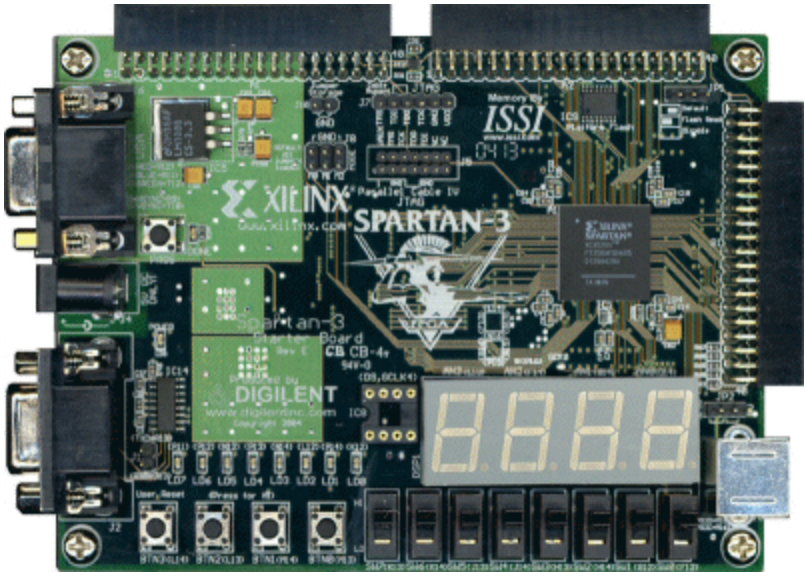
Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Πανεπιστήμιο Θεσσαλίας

CE430
ΕΡΓΑΣΤΗΡΙΟ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ



ΑΣΚΗΣΗ 4:
ΣΧΕΔΙΑΣΗ LCD CONTROLLER

PATSIANOTAKIS CHARALAMPOS
2116

Εισαγωγή

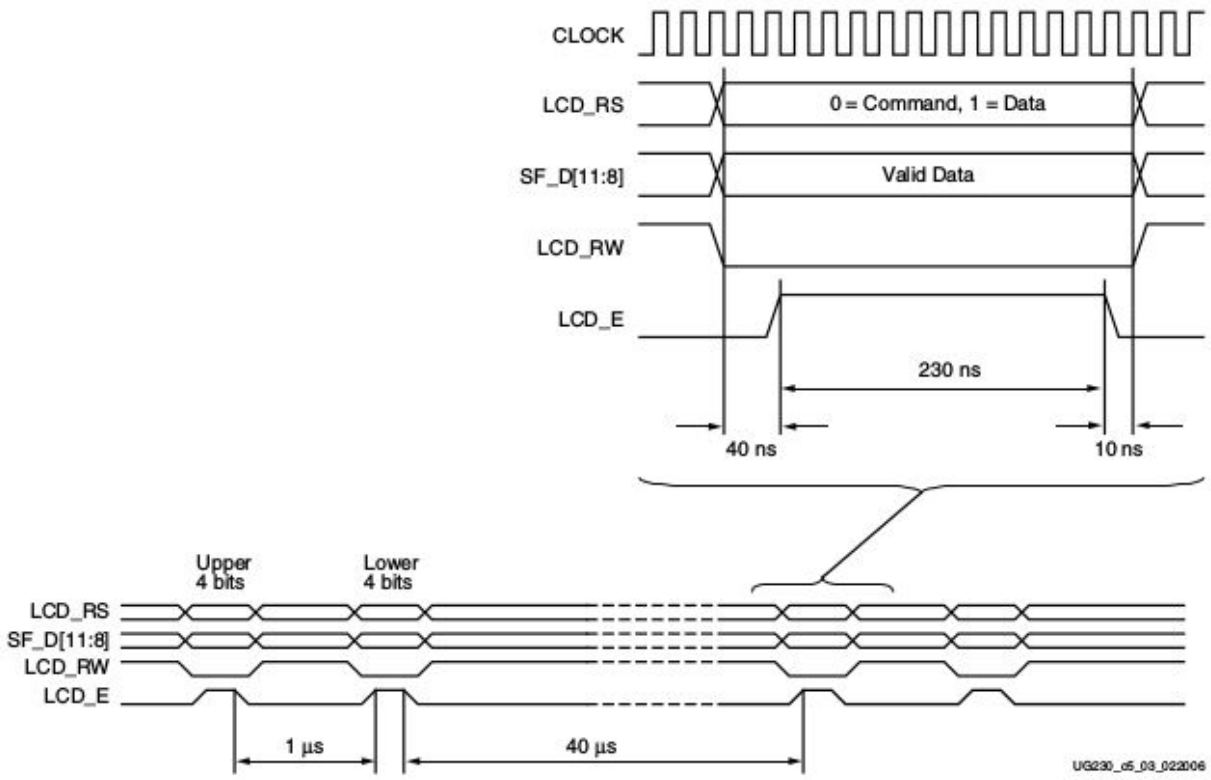
Σε αυτήν την εργασία στόχος ήταν η υλοποίηση ενός LCD Controller. Ο ελεγκτής αυτός, αν και θα μπορούσε να υλοποιηθεί με την χρήση ενός επεξεργαστή, για λόγους κόστους σε χρόνο, χώρο στην FPGA και ενέργεια θα υλοποιηθεί με 2 ομάδες από FSM. Η πρώτη αποτελεί τον αποστολέα, δηλαδή δέχεται εντολές και στέλνει τα σήματα στην LCD σύμφωνα με τα πρωτότυπα της τελευταίας. Η δεύτερη είναι εκείνη που παράγει τις εντολές, οι οποίες στέλνονται στον επεξεργαστή της LCD.
Η εργασία δυστυχώς δεν επιτεύχθηκε με επιτυχία, καθώς ένα crash του λειτουργικού συστήματος είχε αποτέλεσμα την απώλεια του κώδικα της εργασίας οπότε και πήγε αναγκαστηκά πίσω, μην έχοντας αρκετό χρόνο για debug. Η τεχνική αναφορά χωρίζεται σε 2 μέρη, ένα μέρος για κάθε fsm όπως αναφέρθηκε πριν.

ΜΕΡΟΣ Α

Instruction Sender

Υλοποίηση

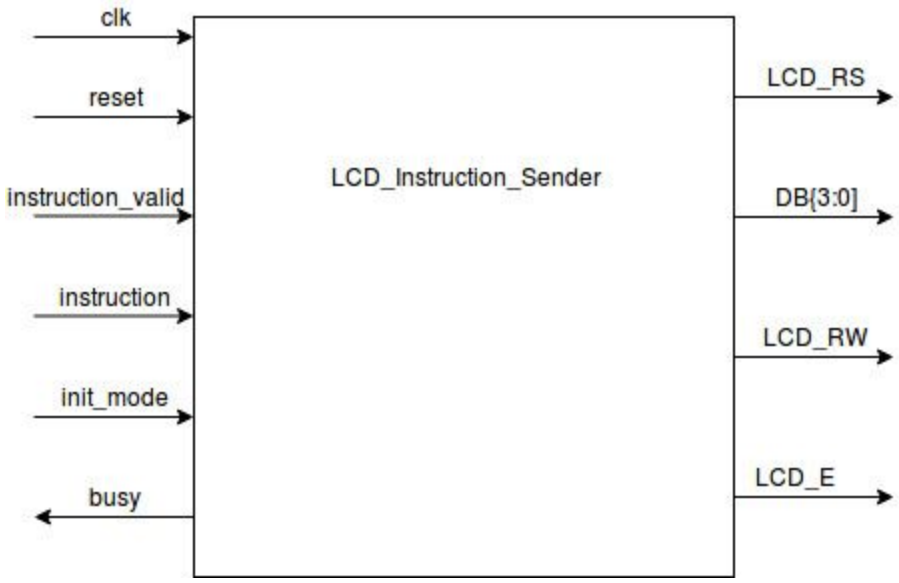
Για να κατανοήσει η LCD τα δεδομένα που της στέλνονται και να τα μετατρέψει σε πληροφορία, αυτά πρέπει να αποστέλλονται με συγκεκριμένους χρονισμούς, πράγμα που η Instruction_Sender είναι υπεύθυνη.
Οι χρόνοι που επιβάλλει το πρωτόκολλο είναι οι εξής:



Όπως διαπιστώθηκε σε περαιτέρω μελέτη αυτό ακριβώς πρέπει να εκπληρωθεί, εφόσον δεν η LCD δεν βρίσκεται σε φάση αρχικοποίησης.

Η FSM που υλοποιείται έχει τις εξόδους προς την LCD, όπως βλέπουμε στο προηγούμενο σχήμα και μία έξοδο προς το κύκλωμα

ότι είναι απασχολημένο. Εισόδους έχει την εντολή που πρέπει να αποστείλει, ένα σήμα που αποδεικνύει την εγκυρότητα της εντολής και ένα σήμα που δείχνει αν η LCD βρίσκεται σε φάση αρχικοποίησης. Πέρα από αυτές εννοείται υπάρχει το ρολόι και το reset.



Τεχνικά, η FSM παραλληλοποιείται σε 2 FSM. Η πρώτη είναι υπεύθυνη να σηκώνει το LCD_E όπως βλέπουμε στο πρώτο σχήμα, ενώ η δεύτερη ελέγχει αν στέλνονται τα lower ή τα upper bits. Το reset μας τοποθετεί την πρώτη σε IDLE_STATE και την δεύτερη σε UPPER_BITS_STATE.

Στην περίπτωση του UPPER BITS_STATE:

-Η πρώτη fsm βρίσκεται αρχικά σε IDLE_STATE όπου το LCD_RS και LCD_RW δίνουν σήμα 1, ενώ το LCD_E και το DB είναι στο 0. Το IDLE_STATE σε συνδυασμό με το UPPER_BITS_STATE είναι η μοναδική κατάσταση όπου το busy είναι στο 0. Εφόσον ενεργοποιηθεί το instruction valid περνάμε στο SETUP_STATE όπου για 40 ns (2 κύκλοι) κρατάμε τα LCD_RW στο 0 ενώ το LCD_RS παίρνει την τιμή του instruction [8] καθώς και τα DB τις τιμές του instruction [7:4]. Έπειται για 240 ns αυτή η fsm κρατάει σταθερά τις ίδιες τιμές της προηγούμενης κατάστασης εκτός του LCD_E που

ανεβαίνει στο 1. Επειδή όμως σύμφωνα με τις προδιαγραφές το Enable πρέπει να παραμένει σταθερό για 230 ns αντί 240 που δεν είναι πολλαπλάσιο του ρολογιού, χρησιμοποιήθηκε ένα τρικ. Παράλληλα με τις fsm που βλέπουμε, υπάρχει ένας καταχωρητής που ενεργοποιείται στην αρνητική ακμή του ρολογιού. Σε κάθε αρνητική ακμή, έχει ελέγξει αν η τιμή του μετρητή είναι ίση με την μέγιστη τιμή που πρέπει να έχει όσο μένει στο ENABLE_STATE και ελέγχει αν βρισκόμαστε στην αναφέρουσα κατάσταση. Αν ισχύει η συνθήκη που αναφέρθηκε βγάζει έξοδο 0, διαφορετικά 1. Αυτή η έξοδος συνδέεται με την LCD_E έξοδο της FSM με μία πύλη AND. Με αυτό το κόλπο κρατάμε το τελικό LCD_E στο 0 τα τελευταία 10 ns των 240 ns, δηλαδή παραμένει στο 1 για 230 ns. Με την πάροδο των 240 ns ελέγχουμε αν βρισκόμαστε σε init mode. Αν όντως βρισκόμαστε το bits state παραμένει όπως έχει (UPPER BITS STATE), διαφορετικά μεταβαίνουμε στο LOWER BITS STATE. Σε κάθε περίπτωση γυρνάμε σε IDLE_STATE.

Στην περίπτωση του LOWER BITS_STATE:

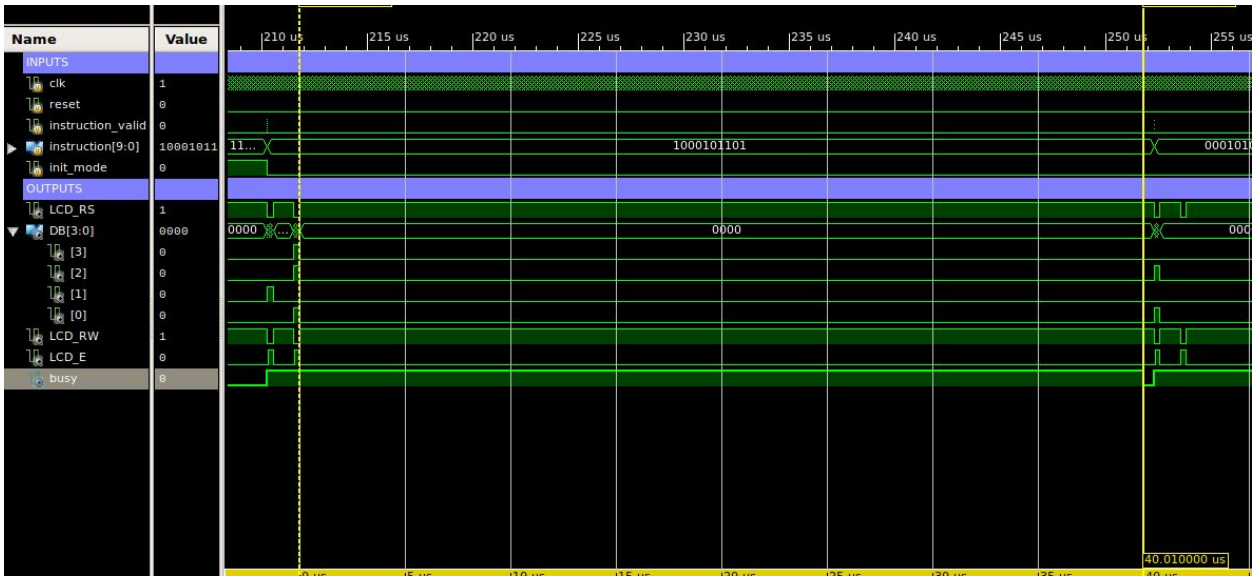
Ξεκινάμε πάλι με το IDLE_STATE, στο οποίο απλά περιμένουμε να περάσει 1 μs. Οι έξοδοι (εκτός του busy) ταυτίζονται με αυτές που υπήρχαν στο IDLE_STATE σε συνδυασμό με το UPPER_BITS_STATE. Εφόσον τελειώσει το 1μs μεταβαίνουμε στο SETUP_STATE. Ότι ισχύει για SETUP και ENABLE του UPPER BITS STATE ισχύει και στο LOWER, με την διαφορά ότι το DB παίρνει τώρα τις τιμές instruction [3:0]. Αφού τελειώσουμε το ENABLE_STATE, μεταβαίνουμε στο WAIT_STATE, όπου περιμένουμε να περάσουν 40 μs, ώστε να κατέβει το busy στο 0 και η FSM να είναι έτοιμη να δεχτεί την επόμενη εντολή.

Επαλήθευση

Αρχικά επαληθεύτηκε το κύκλωμα για την περίπτωση που το init_mode είναι στο 1. Διαπιστώθηκε ότι το busy κατεβαίνει έναν κύκλο πιο πριν από ότι έπρεπε, όμως δεδομένου ότι ο ελεγκτής της αρχικοποίησης δεν επιτρέπει την επόμενη εντολή χωρίς κάποια καθυστέρηση (και λόγω της τελευταίας στιγμής, όπως αναφέρθηκε στην εισαγωγή)δεν δόθηκε ιδιαίτερη σημασία. Ένα παράδειγμα:

Έπειτα, δοκιμάστηκε για init mode στο 0. Σε αυτήν την περίπτωση όλα πήγαν κατ' ευχήν. Ένα χαρακτηριστικό παράδειγμα είναι η επόμενη εικόνα. Διακρίνουμε να μεταφέρεται πρώτα η πληροφορία των 4 upper bits και μετά από περίπου 1 μs των 4 lower bits. Επίσης βλέπουμε ότι η χρονική απόσταση είναι 1 μs και μισός κύκλος (έχει να κάνει με το ότι το enable σήμα είναι στο 1 για 11 κύκλους και μισό κύκλο), πράγμα που δεν επηρεάζει, εφόσον το 1μs είναι ο ελάχιστος χρόνος και όχι ο ακριβής.

Και για την ίδια εντολή τέλος βλέπουμε ότι το busy είναι στο 1 για τουλάχιστον 40 μs μετά το πέρας που σταλούν τα lower bits (οπότε δεν επιτρέπει νέα εντολή).

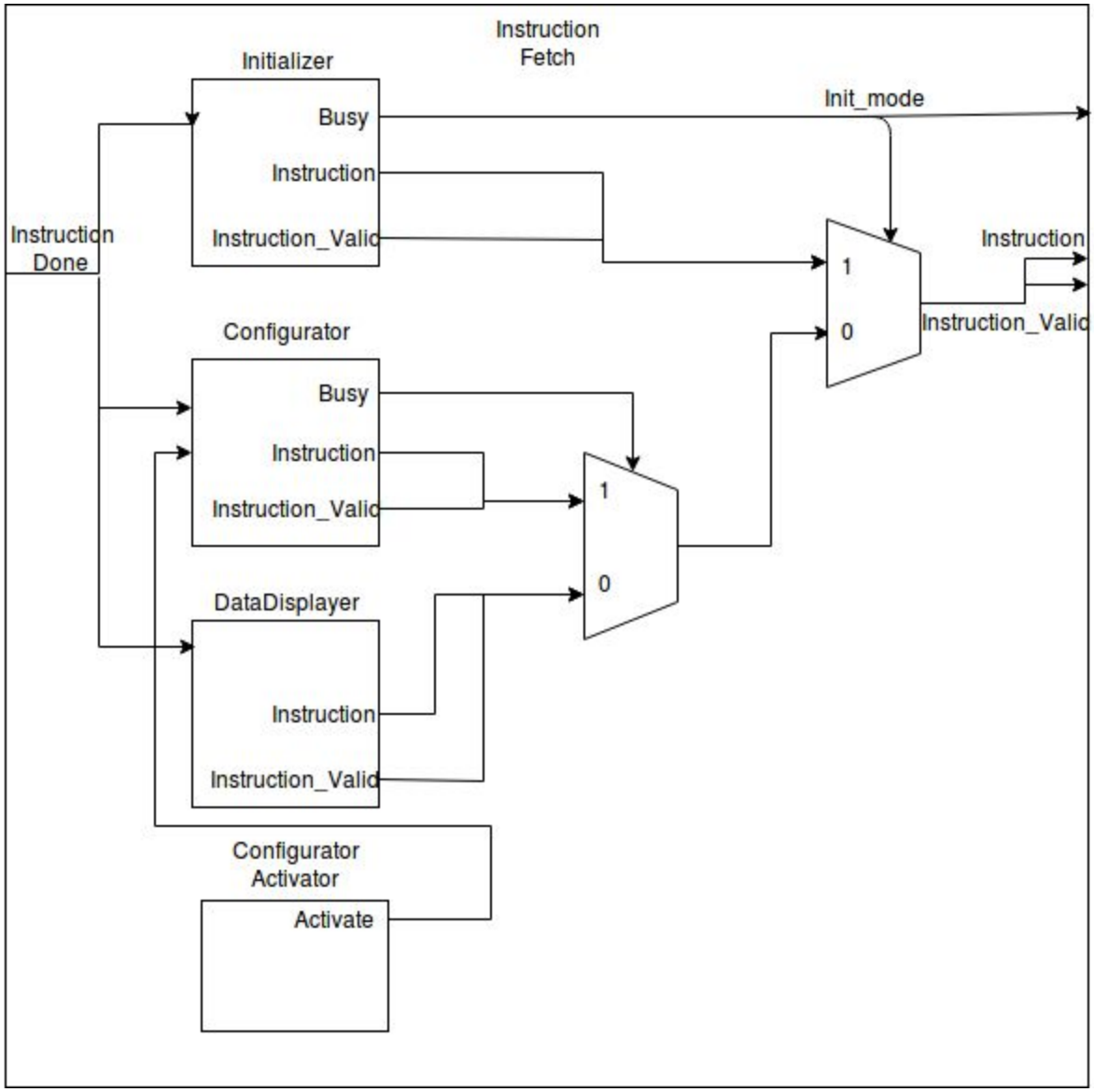


ΜΕΡΟΣ Β

Instruction Fetch

Υλοποίηση

Η συγκεκριμένη μονάδα είναι υπεύθυνη για την παραγωγή των σωστών εντολών και να τις στέλνει στο Instruction Sender. Η μονάδα χωρίζεται σε 3 fsm, τον Initializer, τον Configurator και τον DataDisplayer. Επίσης υπάρχει ο Configurator_Activator που στέλνει σήμα ενεργοποίησης ανά 1 sec για να γίνει το configuration. Την επιφανειακή λογική της μονάδα και τα βασικά σήματα τα διακρίνουμε στο παρακάτω σχήμα:
(δεν έχει σημειωθεί το enable για το DataDisplayer που είναι στο 1 μόνο όταν το config_busy και το init_mode είναι ταυτόχρονα 0)



Initializer:

Αποτελείται από 9 στάδια, τα οποία εναλλάσονται αν στέλνουν μήνυμα ή υπάρχει περίοδος υπομονής. Ξεκινάμε με υπομονή 15 ms και βάζουμε το instruction σε 00_0011_0000 (το οποίο θα διατηρηθεί ως και το στάδιο 6) με valid 0. Έπειτα κρατάμε το valid στο 1 για 240 ns. Ξαναπεριμένουμε 4.1 ms κρατώντας το valid 0 και μετα το ξαναβάζουμε στο 1 για 240 ns. Περιμένουμε 100 us με το valid στο 0 και το ξαναβάζουμε στο 1 για 240 ns πάλι. Τώρα περιμένουμε 40 us με το valid στο 0 έχοντας αλλάξει το instruction

σε 00_0010_0000. Ενεργοποιούμε το valid για 240 ns και το απενεργοποιούμε για 40 us πάλι. Τώρα ρίχνουμε το busy ώστε να συνεχίσει ο Configurator και ο DataDisplayer. Ο Initializer παραμένει στην τελευταία κατάσταση, μέχρι να ξαναπατηθεί το reset.

Configurator

Αφού έχει πέσει το init_mode (το busy του Initializer) ο Configurator_Activator στέλνει σήμα ενεργοποίησης το οποίο θα κάνει από εδώ και στο εξής για κάθε δευτερόλεπτο.

Ο Configurator αποτελείται από τις καταστάσεις STATE_IDLE, STATE_A, STATE_B, STATE_C, STATE_D, STATE_E. Έχοντας μπει μεσω του reset στο STATE_IDLE, περιμένει το σήμα ενεργοποίησης έχοντας βάλει το valid σε 0 και το instruction σε 00_0010_1000 (η επόμενη εντολή). Αφού μπει σε μία από τις επόμενες καταστάσεις, δίνει valid για ένα κύκλο (γίνεται με την βοήθεια flag), και περιμένει το done για να πάει στην επόμενη κατάσταση. Αυτό δεν ισχύει μόνο στο STATE_E, αφού είναι κατάσταση αναμονής. Αφού περάσουν 82000 κύκλοι σε αυτή την κατάσταση, επανέρχεται στο STATE_IDLE και κατεβάζει το busy, ώστε να λειτουργήσει ο DataDisplayer. Οι εντολές ανά κατάσταση είναι:

STATE_A: 00_0010_1000
STATE_B: 00_0000_0110
STATE_C: 00_0000_1100
STATE_D: 00_0000_0001

DataDisplayer:

Εφόσον δεν λειτουργεί ο Initializer ή ο Configurator είναι ώρα να δουλέψει ο DataDisplayer. Ο DataDisplayer όσο είναι ενεργοποιημένος στέλνει εναλλάξ εντολή ανάθεσης DDRAM και εντολή εγγραφής δεδομένων στην DDRAM.

Κατάσταση ανάθεσης DDRAM:

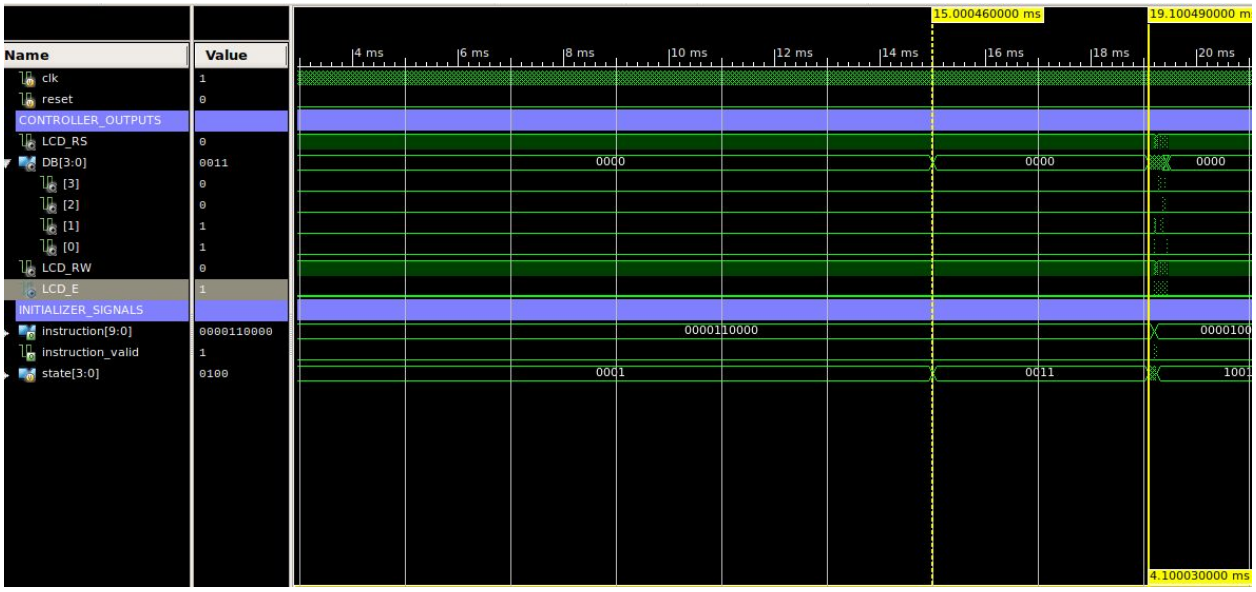
Το instruction [9:7] λαμβάνει την τιμή 001 και το [6:0] την τωρινή DDRAM διεύθυνση. Παράλληλα αλλάζει η τιμή της επόμενης διεύθυνσης. Αν η DDRAM διεύθυνση έχει τιμή 000_1111 επόμενη διεύθυνση είναι η 100_0000, ενώ αν η διεύθυνση έχει τιμή 100_1111 επόμενη τιμή είναι η 000_0000. Σε διαφορετική περίπτωση, επόμενη τιμή διεύθυνσης είναι η σύγχρονη αυξημένη κατά 1.

Κατάσταση γραψίματος DDRAM:

Το instruction [9:8] λαμβάνει την τιμή 10 και το [7:0] την τιμή δεδομένων που βρίσκονται στην διεύθυνση της BRAM που έχει ορισθεί. Για χάρη απλότητας, μέχρι να πραγματοποιηθεί σωστά ο ελεγκτής, η επόμενη διεύθυνση της BRAM ορίζεται η τρέχουσα αυξημένη κατά 1, εκτός αν τιμή είναι η 1000, οπότε και μηδενίζεται. Στόχος είναι να εμφανιστεί στην LCD το βαρετό μήνυμα 0123456789ABCDEF.

Επαλήθευση

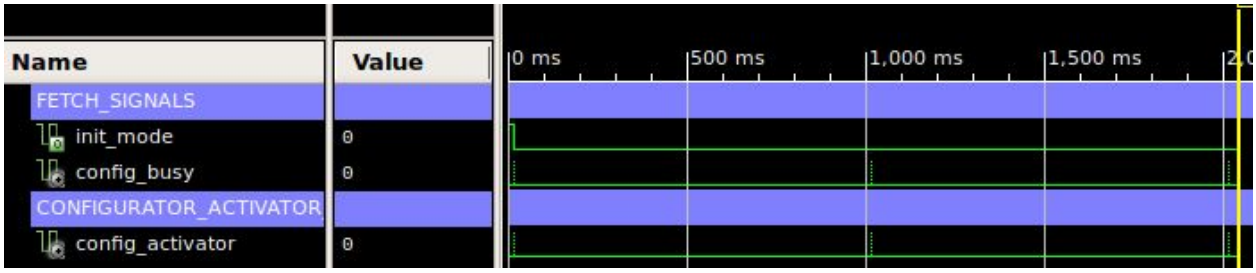
Initializer:
Περιμένει αρχικα τουλαχιστον 15 ms, και έπειτα αφου στείλει την πρωτη εντολή περιμένει τουλάχιστον 4.1 ms



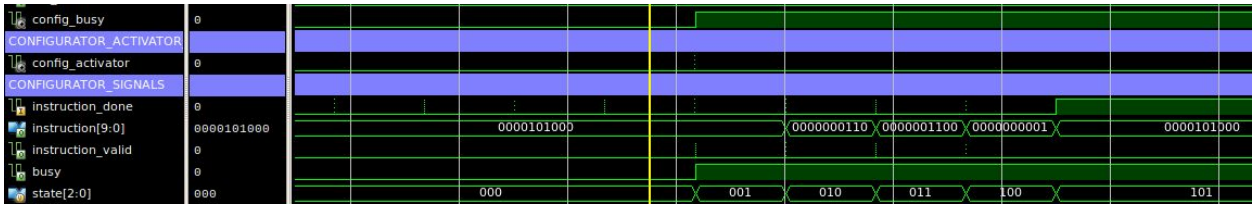
Ενώ ενδιάμεσα έχει στείλει την εντολή 0x3 για 12 κύκλους

Και μετά από 40 μs στην 9 κατάσταση, πέφτει το busy, ώστε να πάρει σειρά ο Configurator

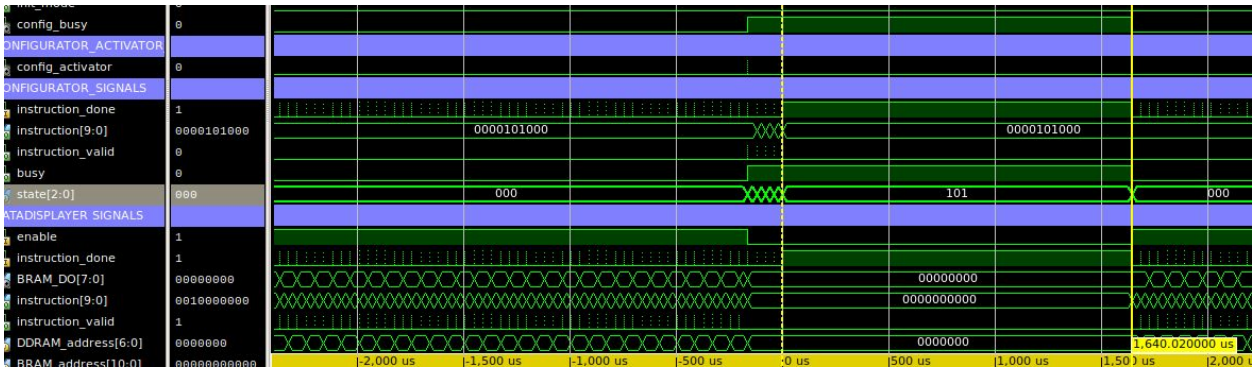
Ο activator του Configurator ενεργοποιείται καθε 1 sec.



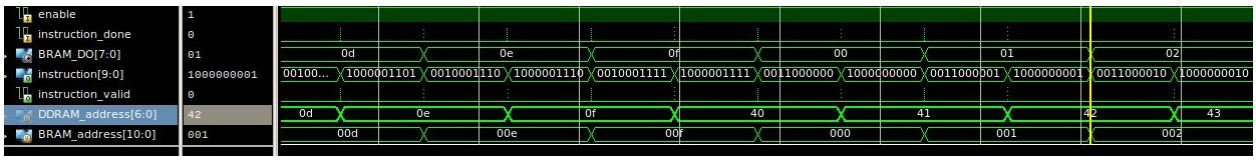
Καθε φορα που ενεργοποιείται ο activator, ο configurator στέλνει τις εντολές που αναφέρθηκαν.



Και περιμένει 82000 κύκλους (1.64 ms) (διακρίνεται και το γεγονός ότι τα σήματα στο DataDisplayer, όσο δουλεύει ο Configurator, είναι σε κατάσταση ηρεμίας).



Τυχαίο στιγμιότυπο από το DataDisplayer όπου βλέπουμε στην 16δική τιμή διεύθυνσης της DDRAM 0x0F, μεταβαίνουμε στην 40.



Ενώ στην τιμή της DDRAM 0x4F μεταβαίνουμε στην 0x0