Εργαστηριακή άσκηση 2

Τομαράς Παναγιώτης-Δημήτριος en1200002

Σύστημα δειγματοληψίας δεδομένων και καταχώρησης σε μνήμη τύπου block ram.

Περιεχόμενα

[Εισαγωγή 1](#_Toc55745352)

[Σχεδιασμός – αναλυτική περιγραφή μονάδων και λειτουργιών 2](#_Toc55745353)

[Sync\_stage 2](#_Toc55745354)

[Median\_filter 2](#_Toc55745355)

[Fsm\_wr 2](#_Toc55745356)

[Υποθέσεις – σχεδιαστικές επιλογές 3](#_Toc55745357)

[Fsm\_rd 3](#_Toc55745358)

[Υποθέσεις – σχεδιαστικές επιλογές 3](#_Toc55745359)

[Interface 4](#_Toc55745360)

[Έλεγχος σωστής λειτουργίας 5](#_Toc55745361)

[Sync\_stage 5](#_Toc55745362)

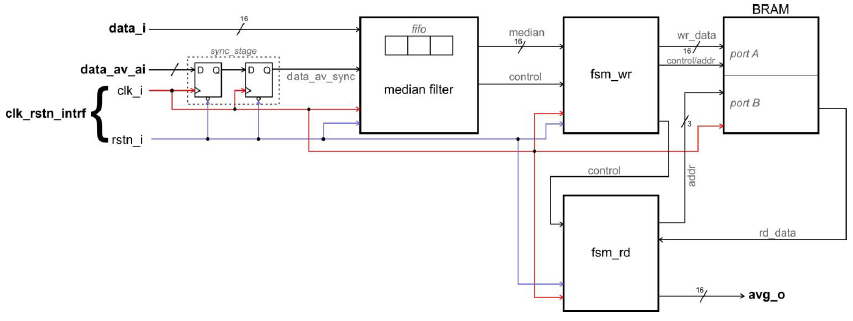
[Median\_fliter 5](#_Toc55745363)

[Fsm\_wr 6](#_Toc55745364)

[Fsm\_rd 7](#_Toc55745365)

# Εισαγωγή

Στην άσκηση αυτή υλοποιούμε ένα σύστημα δειγματοληψίας δεδομένων και καταχώρησης σε μνήμη τύπου block RAM. Τα δεδομένα 16-bit παράγονται και δειγματοληπτούνται με ασύγχρονο τρόπο και επεξεργάζονται πριν την καταχώρηση τους από ένα φίλτρο τύπου median μήκους 3 θέσεων. Τα δεδομένα ολισθαίνουν σειριακά σε μία μνήμη FIFO 3 θέσεων και σε κάθε νέα εισαγωγή το φίλτρο υπολογίζει την median τιμή και την προωθεί για καταχώρησης τη μνήμη. Στην συνέχεια οι τιμές καταχωρούνται, με τη χρήση μηχανής καταστάσεων σε μία μνήμη Block RAM 8x16-bit και η διαδικασία δειγματοληψίας σταματάει όταν γεμίσουν και οι 8 θέσεις της μνήμης. Η μνήμη είναι dual port και όταν γεμίσει μία δεύτερη μηχανή καταστάσεων διαβάζει τα δεδομένα και υπολογίζει την μέση τιμή τους και σταματάει η λειτουργία του κυκλώματος, ενώ επανεκκίνηση μπορεί να γίνει μέσω του σήματος reset.



# Σχεδιασμός – αναλυτική περιγραφή μονάδων και λειτουργιών

## Sync\_stage

Επειδή τα δεδομένα δειγματοληπτούνται με ασύγχρονο τρόπο θέλουμε ένα αρχικό στάδιο sync stage. Στο στάδιο αυτό έχουμε δύο D flip flop δηλαδή ένα Pipeline δύο σταδίων για να αντιμετωπίσουμε τη μετασταθή κατάσταση, η οποία θέλει 2 κύκλους για κάθε είσοδο. Με τον τρόπο αυτό παίρνουμε στο σύστημά μας ένα σύγχρονο σήμα.

## Median\_filter

Έπειτα έχουμε τη μονάδα median filter. Η λειτουργία αυτής της μονάδας είναι να υπολογίζει το median αριθμό μίας FIFO.

Για να υλοποιήσω αυτή τη μονάδα επέλεξα να δημιουργήσω έναν πίνακα 3 θέσεων, ένα δείκτη στη θέση πίνακα που πρέπει να γράψω ανά χρονική στιγμή, χωρίς να κάνω ολίσθηση.

Έχουμε 3 processes που τρέχουν. Τα 2 είναι σύμφωνα με το ρολόι και το άλλο είναι συνδυαστικό. Στο πρώτο Process υλοποιώ την αναμονή για το σήμα ενεργοποίησης data\_av\_sync, το οποίο πυροδοτεί την καταχώρηση στον πίνακα του αριθμού που παίρνω ως είσοδο και την αύξηση του δείκτη θέσης του πίνακα.

Ταυτόχρονα, το 2ο process παρακολουθεί πότε θα πάρω σήμα ενεργοποίησης και τη στιγμή εκείνη βγάζει στην έξοδο median\_en=1 δηλώνοντας πως η τιμή που βρίσκεται στην έξοδο median είναι αποδεκτή. Αυτό βασίζεται στη λειτουργία του 3ου process το οποίο είναι συνδυαστικό και πυροδοτείται από κάθε αλλαγή που πραγματοποιείται στο array που κρατάω τις τιμές. Ο τρόπος που υπολογίζει το median αριθμό είναι με επαναλαμβανόμενα if, όπου αρχικά βρίσκει το Index του μικρότερου αριθμού από τους 3, και ύστερα συγκρίνει τους 2 εναπομείναντες αριθμούς κρατώντας το μικρότερο, ο οποίος είναι και ο μεσαίος.

## Fsm\_wr

Επόμενο stage στη μονάδα είναι το fsm\_wr. Στο στάδιο αυτό θέλουμε να λαμβάνουμε έγκυρες τιμές median αριθμών και να τις καταχωρούμε στη block ram. Για το σκοπό αυτό έχουμε τη λειτουργία μιας fsm μηχανής 3 καταστάσεων. Οι καταστάσεις αυτές είναι οι IDLE, WRITE, PREPARE.

Στην πρώτη έχουμε το σημείο ηρεμίας της μηχανής, όπου οι έξοδοι write enable=0, read enable=0 εφόσον δε γράφουμε ούτε ενεργοποιούμε την fsm\_rd. Αναμένουμε το σήμα median\_en, δηλαδή έναν έγκυρο αριθμό ώστε να προχωρήσουμε στο στάδιο WRITE για να το γράψουμε στη μνήμη. Αμέσως βγάζουμε έξω την τιμή βασιζόμενοι στο ότι η μνήμη γράφει μόνο όταν παίρνει write enable σήμα, το οποίο ενεργοποιούμε στον επόμενο κύκλο, κι έτσι αποφεύγουμε τη χρήση ενδιάμεσου καταχωρητή για την αποθήκευση της τιμής του median.

Στην κατάσταση WRITE ενεργοποιούμε το σήμα write enable και στην έξοδο address to write βγάζουμε τη διεύθυνση μνήμης που θέλουμε να γραφτεί η τιμή.

Στο 3ο και τελευταίο στάδιο, PREPARE, αυξάνουμε το μετρητή της διεύθυνσης για το γράψιμο της μνήμης και ελέγχουμε αν έχουμε φτάσει στις 8 εγγραφές, γεγονός που μας κάνει να ενεργοποιήσουμε με σήμα read enable τη μονάδα fsm rd ώστε να αρχίσει το διάβασμα των 8 αριθμών που έχουν αποθηκευτεί και να υπολογίσει το μέσο όρο τους.

### Υποθέσεις – σχεδιαστικές επιλογές

Σύμφωνα με αυτή την υλοποίηση γνωρίζω πως η κάθε εγγραφή στη μνήμη μπορεί να πραγματοποιηθεί κάθε 3 κύκλους. Επομένως, τα ασύγχρονα δεδομένα θέλω να τα λαμβάνω ανά 3 κύκλους του ρολογιού.

## Fsm\_rd

Τελευταίο module της μονάδας μας είναι η μηχανή καταστάσεων fsm\_rd. Στη μονάδα αυτή θέλουμε να διαβάσουμε τις 8 αποθηκευμένες τιμές στη ram και να υπολογίσουμε το μέσο όρο τους.

Αυτό το επιτυγχάνουμε με τη μηχανή 6 καταστάσεων IDLE, ASK, WAITING, WAITING2, READ, CALCULATE, έναν πίνακα 8 θέσεων και ένα μετρητή για τη διεύθυνση διαβάσματος που ζητάμε.

Στην κατάσταση ηρεμίας επιστρέφουμε μετά από κάθε υπολογισμό μέσου όρου ή είναι η αφετηρία μας. Επομένως, έχουμε την αρχικοποίηση του δείκτη του πίνακα 8 θέσεων που κρατάμε τους αριθμούς και του μετρητή διεύθυνσης.

Στο στάδιο ASK ζητάμε μια τιμή από τη RAM, δηλαδή βγάζουμε τη διεύθυνση που θέλουμε να διαβάσουμε. Η μνήμη θα μας απαντήσει με έγκυρη τιμή σε 2 κύκλους και γι αυτό εισάγουμε 2 καταστάσεις αναμονής μέχρι το διάβασμα της τιμής. Μετά αυξάνουμε και κατά 1 το μετρητή για τη διεύθυνση ώστε να προετοιμαστούμε για την επόμενη ανάγνωση.

Στο στάδιο READ καταχωρώ στον πίνακα την τιμή που μου δίνει η μνήμη. Όταν γεμίσει ο πίνακας, δηλαδή έχω γράψει 8 τιμές περνάω στο στάδιο του υπολογισμού.

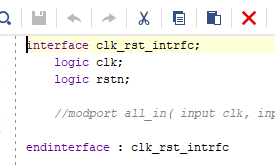
Στο CALCULATE μέσα σ’ έναν κύκλο υπολογίζω το άθροισμα των 8 αριθμών και με το assign έχω την ολίσθηση του αποτελέσματος κατά 3 θέσεις δεξιά, δηλαδή τη διαίρεση κατά 8.

### Υποθέσεις – σχεδιαστικές επιλογές

* 1. Αρχικά είχα υλοποιήσει το CALCULATE να διαρκεί 9 γύρους, όπου σε κάθε χτύπο του ρολογιού άθροιζα μια τιμή από τον πίνακα συσσωρευτικά στη μεταβλητή average\_calc και στον 9ο κύκλο έκανα ολίσθηση δεξιά. Αυτό απαιτούσε και ακόμα μια κατάσταση την PRINT ώστε να ολισθαίνει την τιμή της ενδιάμεσης μεταβλητής average\_calc στην έξοδο. Τελικά, έβαλα το άθροισμα σε έναν κύκλο και με συνδυαστικό τρόπο την ολίσθηση του αποτελέσματος στο assign. Αυτό είναι πιο γρήγορο και ίσως πιο ώριμη υλοποίηση με τη σκέψη ότι ένας αθροιστής είναι μια δεδομένη μονάδα που χρησιμοποιείται από το πρόγραμμα. Μια ακόμα σκέψη ήταν ότι θα ήθελα την ύπαρξη ενός σήματος average output enable, με το σκεπτικό ότι εγώ κρατάω στο output την έγκυρη τιμή μόνο για έναν κύκλο, αφού στο IDLE ξαναβγάζω avg\_o<=0.
  2. Για τη διαδικασία του READ είχα σκεφτεί τη λύση να κρατάω σαν previous data input την τιμή που στέλνει η μνήμη RAM και να τη συγκρίνω με την παρούσα τιμή, με αφορμή το σχηματικό που παρουσιάζει τη λειτουργία της και δείχνει ότι κάθε στιγμή βγάζει στο doutb την τιμή από το προηγούμενο διάβασμα. Οπότε θα απέφευγα τις 2 επιπλέον καταστάσεις με ένα if(data\_in!=previous\_data). Όμως αυτό δε λειτούργησε καθώς έχανα την 1η τιμή, αφού η ram μου έδινε την τιμή της διεύθυνσης 0 αφού στο idle κάνω addr\_rd\_mem<=0 και την πρώτη φορά αυτό το if αποτυγχάνει γιατί ήδη η RAM μου στέλνει αυτή την τιμή.
  3. Όταν βρίσκεται σε διαδικασία ανάγνωσης, ξοδεύει 8\*(4 κύκλους/εγγραφή) +1 κύκλο για υπολογισμό μέσου όρου. Επομένως, στο διάστημα αυτό αν δεχτεί σήμα ενεργοποίησης read\_en το αγνοεί και δεν το λαμβάνει υπόψιν.

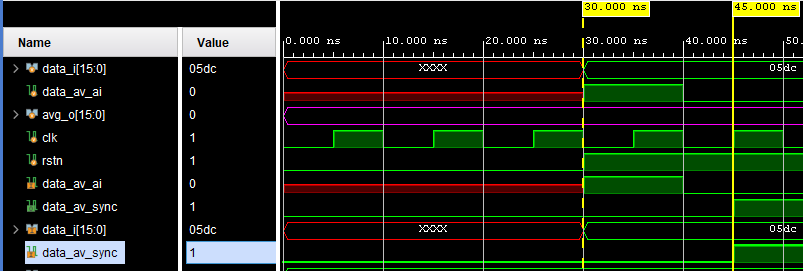
## Interface

Σύμφωνα με την παρακάτω εικόνα έχω υλοποιήσει το interface επικοινωνίας ανάμεσα στα διαφορετικά κομμάτια του συστήματός μας.



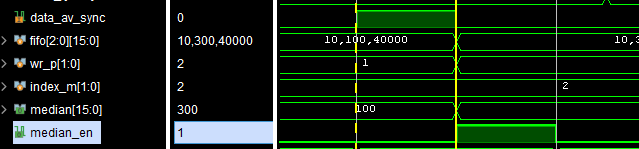
# Έλεγχος σωστής λειτουργίας

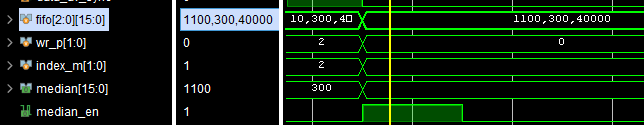
## Sync\_stage



Βλέπουμε τη λειτουργία του sync stage όπου σε 2 κύκλους ενεργοποιείται το σήμα data av sync έπειτα από την ενεργοποίηση του data\_av\_ai

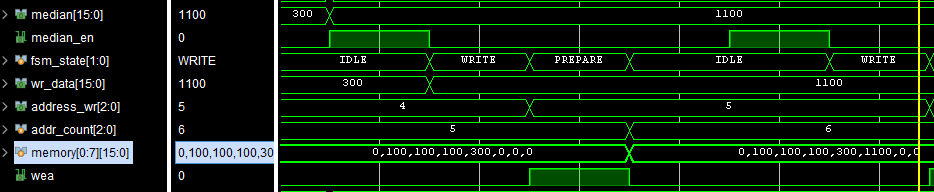
## Median\_fliter





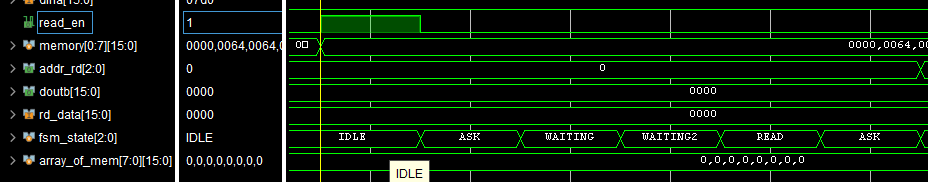
Εδώ βλέπουμε τη λειτουργία του median φίλτρου. Παίρνει σήματα ενεργοποίησης data\_av\_sync και στον επόμενο κύκλο βγάζει σήμα median\_en με την median=300 ως έγκυρη τιμή όταν η FIFO είναι η 10,300,40000. Αντίστοιχα βλέπουμε πως όταν έχουμε 1100,300,40000 ο μέσος είναι ο 1100.

## Fsm\_wr

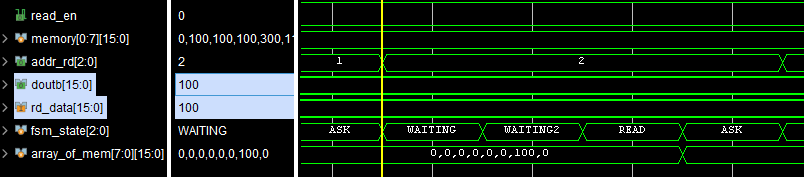


Εδώ έχουμε την fsm\_wr. Λαμβάνει median=1000 και σήμα ενεργοποίησης median\_enable. Αυτό βγάζει αμέσως στο wr\_data την τιμή 1000, στον επόμενο κύκλο δηλαδή στο WRITE ενεργοποιεί το write enable και δίνει τη διεύθυνση address\_wr = 4 για να γραφτεί στην 5η θέση. Στο PREPARE απλώς αυξάνω το μετρητή για τη διεύθυνση εγγραφής για την επόμενη εγγραφή. Βλέπουμε πως η μνήμη στον επόμενο κύκλο ύστερα από την παροχή write enable γράφει τον πίνακα που κρατάει τις τιμές, αλλάζοντας σε 0 100 100 100 300 1100 0 0 από αρχικά 0 100 100 100 300 0 0 0.

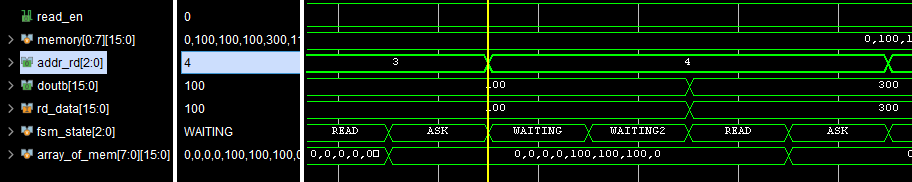
## Fsm\_rd



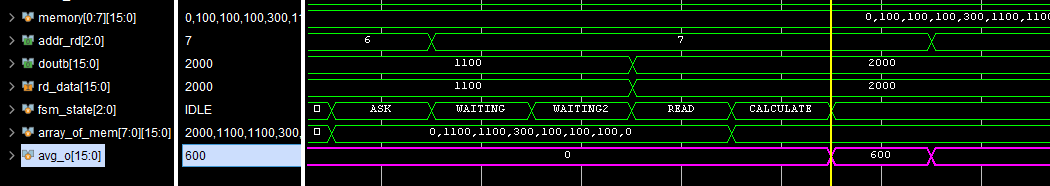
Το fsm\_rd από τη στιγμή που παίρνει read\_en ενεργοποιείται και ξεκινάει τη ζήτηση 8 αριθμών από τη μνήμη. Στο ask ζητάει την πρώτη τιμή στη addr\_rd διεύθυνση μνήμης 0 και στο READ παίρνει την απάντηση που είναι η τιμή 0 (στο memory έχουμε τον πίνακα τιμών της μνήμης).



Σε μεταγενέστερο χρόνο βλέπουμε πως ζητάει δεδομένα από τη διεύθυνση 2 (3η θέση) και λαμβάνει ως απάντηση το 100 .



Τέλος, ζητώντας την τιμή της διεύθυνσης 4 (5ο στοιχείο), ύστερα από αναμονή 2 κύκλων παίρνει την απάντηση του 300 κατά το στάδιο READ που είναι πράγματι η 5η τιμή όπως φαίνεται από το memory.



Βλέπουμε πως στο ASK ζητάμε τη διεύθυνση 7 (8ο στοιχείο), διαβάζουμε στο read το 2000 και περνάμε στο CALCULATE όπου σε έναν κύκλο μας βγάζει το αποτέλεσμα 600.