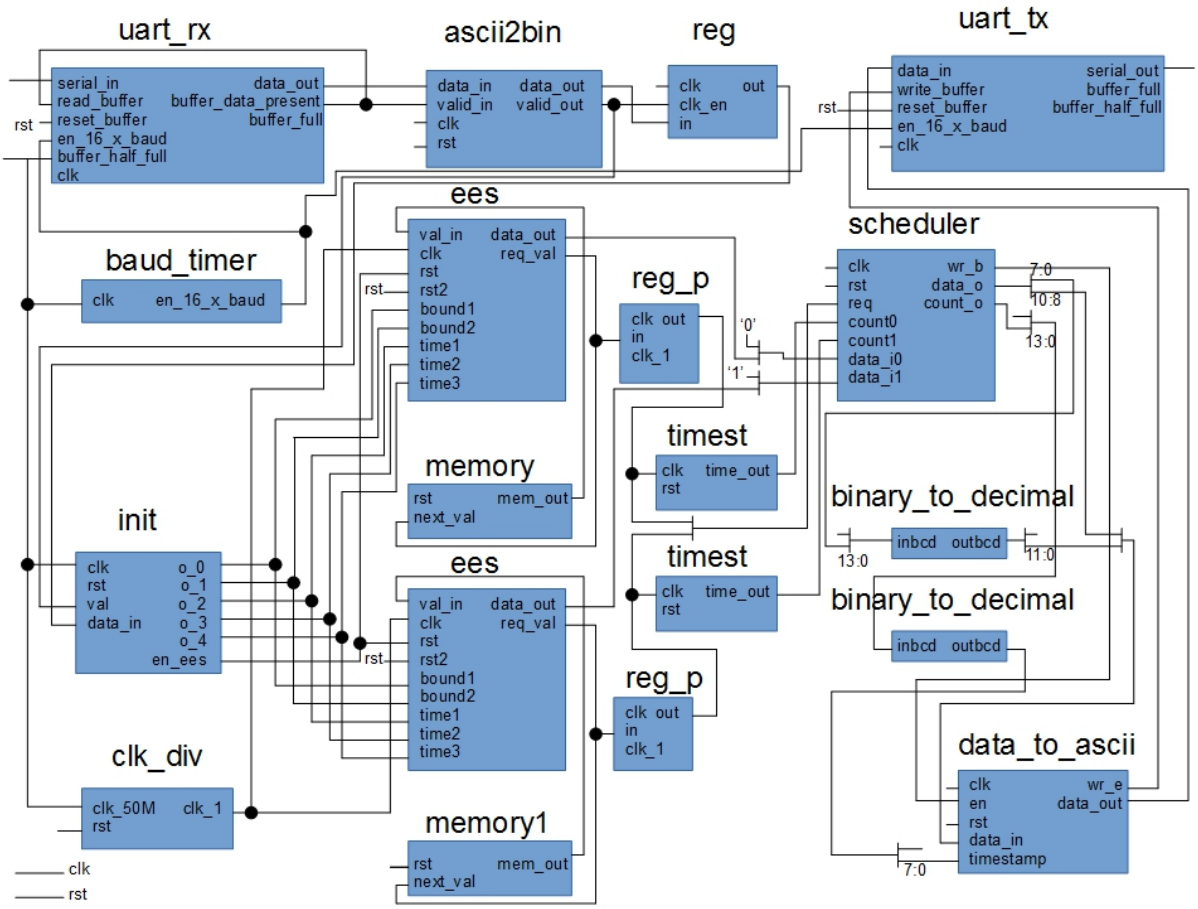
ΕΝΣΩΜΑΤΩΜΕΝΑ ΣΥΣΤΗΜΑΤΑ

ΑΣΚΗΣΗ 3η

ΒΑΣΙΛΗΣ ΚΙΤΣΑΚΗΣ
ΜΔΕ-ΗΑ 2014509

Στην άσκηση αυτή αναπτύχθηκε ένα κύκλωμα το οποίο έπρεπε να περιέχει δύο επεξεργαστές ειδικού σκοπού, που λειτουργούν παράλληλα, μία μονάδα που να παίρνει παράλληλα τα δεδομένα από τους επεξεργαστές και να τα εξάγει σε σειρά, χωρίς να χαθεί κάποια τιμή. Η είσοδος και η έξοδος απ'ο την κάρτα γίνεται με σειριακή σύνδεση. Η επικοινωνία γίνεται με το πρόγραμμα terminal το οποίο στέλνει 8-bit σήματα σε ASCII κωδικοποίηση και δέχεται 8-bit και τα διαβάζει σαν χαρακτήρες ASCII. Το κύκλωμα είναι το εξής



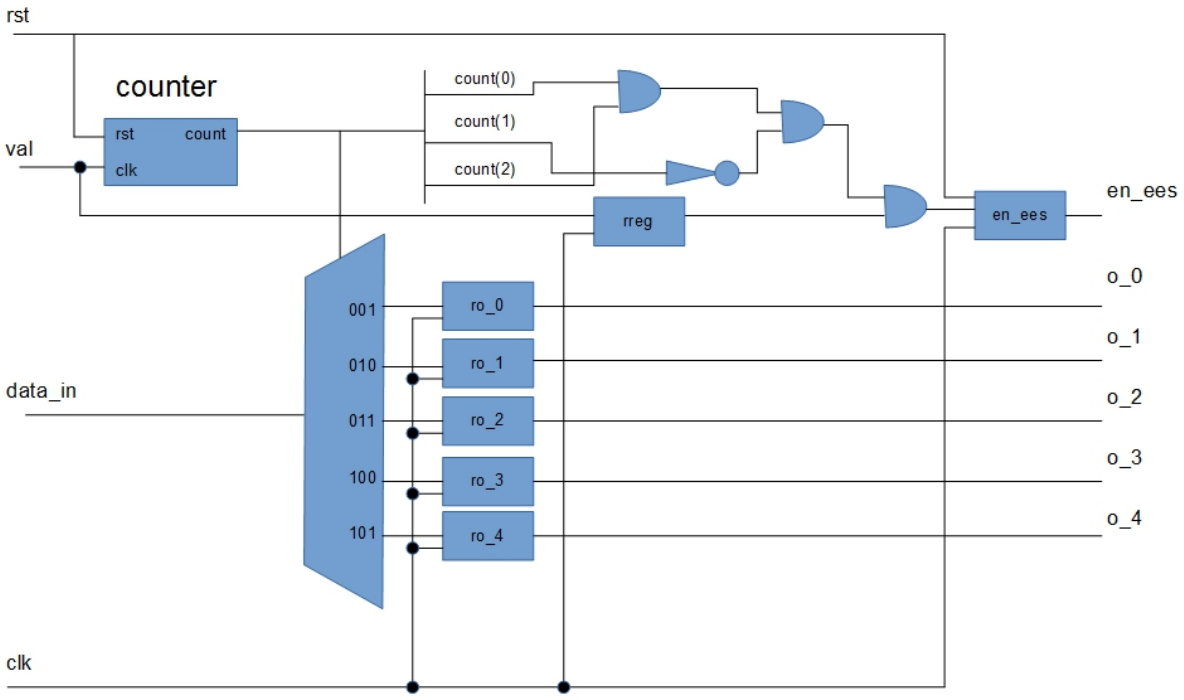
uart_rx
Η μονάδα αυτή δέχεται 1-bit εισόδου και σχηματίζει 8-bit λέξεις, κάθε φορά που σχηματίζεται μία λέξη στο data_out η έξοδος buffer_data_present δίνει '1'. Το buffer_data_present έχει συνδεθεί με την είσοδο read_buffer, έτσι κάθε φορά που ολοκληρώνεται η μία λέξη αρχίζει να σχηματίζεται η επόμενη.

baud_timer
Η μονάδα αυτή ορίζει τον ρυθμό με τον οποίο διαβάζονται τα bit της εισόδου και τον ρυθμό που μεταδίδονται στην έξοδο.

ascii2bin
Η μονάδα αυτή δέχεται σαν είσοδο δεδομένα από την uart_rx, κάθε φορά διαβάζει τρεις αποδεκτές (valid) λέξεις και τις μετατρέπει σε μία 8-bit δυαδική λέξη.

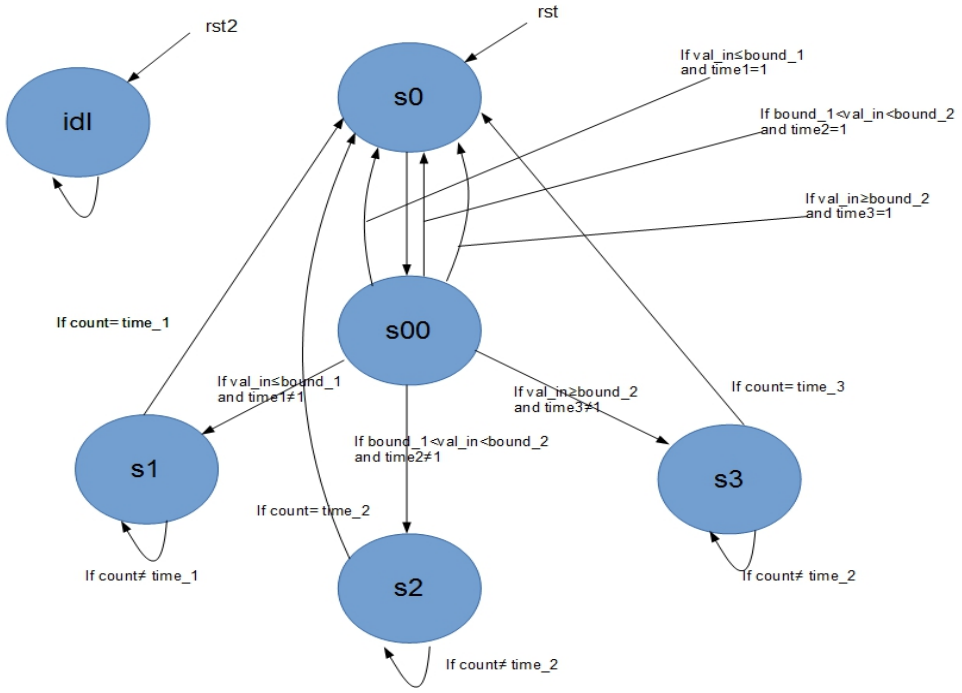
reg
Είναι registers που έχουν περιγραφεί σε behavioral.

init
Σκοπός αυτής της μονάδας είναι να αποθηκεύσει τα όρια που δίνουμε σαν είσοδο, συνολικά πέντε τιμές. Κάθε valid έξοδος της ascii2bin αποθηκεύεται σε έναν register με clock enable το σήμα valid. Έτσι μέχρι να σχηματιστεί η επόμενη λέξη στην έξοδο του ascii2bin στον register, μένει αποθηκευμένη η προηγούμενη. Κάθε λέξη (από τις πέντε συνολικά που θέλουμε) που αποθηκεύεται στον register αυτόν πρέπει να αποθηκευτεί σε έναν από τους πέντε παράλληλους registers, που θα δώσουν τα όρια στους επεξεργαστές. Αυτό γίνεται με έναν αποπλέκτη πέντε εξόδων, με κάθε έξοδο να είναι συνδεδεμένη με έναν register η κάθε μία. Το select (count) του αποπλέκτη είναι συνδεδεμένο με έναν counter που μετράει τα valid (τιμές) του ascii2bin. Έτσι κάθε καινούρια λέξη γράφεται σε διαφορετικό register. Το select ξεκινάει από το 001 επειδή η είσοδος καθυστερεί κατά έναν κύκλο λόγο του reg. Όταν το count γίνει 101 έχουν γραφεί όλες οι τιμές στους registers. Στον επόμενο κύκλο δίνει για έναν κύκλο δίνει ‘1’ το οποίο θα θέσει τους επεξεργαστες σε κατάσταση αρχικοποίησης. Αυτό γίνεται ως εξής. Αποθηκεύεται το valid (μας ενδιαφέρει μόνο το τελευταίο) σε εναν register (rreg), όταν το count γίνει 101 (θα παραμείνει έτσι μιας και δεν πρόκειται να έρθει άλλο valid) παράγουμε την τιμή ‘1’. Αυτή την τιμή την περνάμε από μία πύλη or με την έξοδο του rreg. Έτσι έχουμε ‘1’ για έναν κύκλο όταν γραφτούν όλοι οι registers. Την τιμή αυτήν την καθυστερούμε κατά έναν κύκλο με τον register en_ees. Στο ακόλουθο σχήμα φαίνεται η λογική αυτή.



memory/memory1
Είναι δύο μνήμες rom που δίνουν τιμές στους επεξεργαστές. Κάθε επεξεργαστής έχει την δικιά του. Σε έναν κύκλο ζητάνε τιμή και στον επόμενο κύκλο την δέχονται.

EES
Η μονάδα αυτή είναι ο επεξεργαστής ειδικού σκοπού. Σαν είσοδο δέχεται μια τιμή που τον θέτει σε κατάσταση αδράνειας (rst2), πέντε τιμές που είναι τα όρια (bound1,bound2,time1,time2 και time3) και ένα σήμα που αρχικοποιεί τα όρια που αναφέρθηκαν (rst). Τα όρια του χρόνου αναπαριστούν δευτερόλεπτα. Ο EES λειτουργεί σε ρολόι 0.5s, έτσι δύο κύκλοι αναπαριστούν 1s. Οπότε αρχικά τα όρια αυτά πρέπει να διπλασιαστούν. Σε κάθε περίπτωση θα πρέπει να πάει απο την κατάσταση s0 στην κατάσταση s00 (2 κύκλοι, 1s είδη φαίνεται ότι πρέπει να αφαιρέσουν τουλάχιστον 2) και να παραμείνει στην επόμενη κατάσταση για όσο χρειαστεί (εκτός αν το όριο είναι 1s). Έτσι μιας και μετράμε απο το μηδέν θα πρέπει στην συνθήκη ελένχου να αφαιρέσουμε τρία από το διπλάσιο του ορίου. Τα πραγματικά όρια χρόνου (τιμή ορίου επί 2 μείων τρία) και τα όρια θερμοκρασίας αποθηκεύονται σε πέντε latches ασύγχρονα όταν έρθει ‘1’ στην τιμή της αρχικοποίησης (rst) και ορίζεται σαν επόμενη κατάσταση η s0. Στην s0 ζητάει νέα τιμή από την μνήμη και η επόμενη κατάσταση είναι η s00. Στην s00 έχει λάβει την νέα τιμή από την μνήμη και ελέγχονται τρεις περιπτώσεις για το που ανήκει η τιμή σε σχέση με τα όρια της θερμοκρασίας, που είναι αποθηκευμένα στα latches. Ανάλογα με την περίπτωση, στην τιμή μπροστά μπαίνει ο κωδικός του mode και επιλέγεται η ανάλογη κατάσταση. Για low ο κωδικός είναι 01 και η κατάσταση s1, για high ο κωδικός είναι 10 και η κατάσταση s2, τέλος για emergency ο κωδικός είναι 11 και η κατάσταση s3. Σε κάθε περίπτωση αν το όριο του χρόνου είναι 1 τότε η επόμενη κατάσταση είναι η s0 (ζητάει νέα τιμή). Στις καταστάσεις s1, s2 και s3 (low, high και emergency αντίστοιχα) μένει για τόσους κύκλους όσους ορίζουν τα πραγματικά όρια χρόνου (latches) και όταν η συνθήκη ελένχου γίνει αληθής ορίζεται σαν επόμενη κατάσταση η s0 και ολοκληρώνεται ο “κύκλος” μίας τιμής. Ακολουθεί το fsm.



req_p

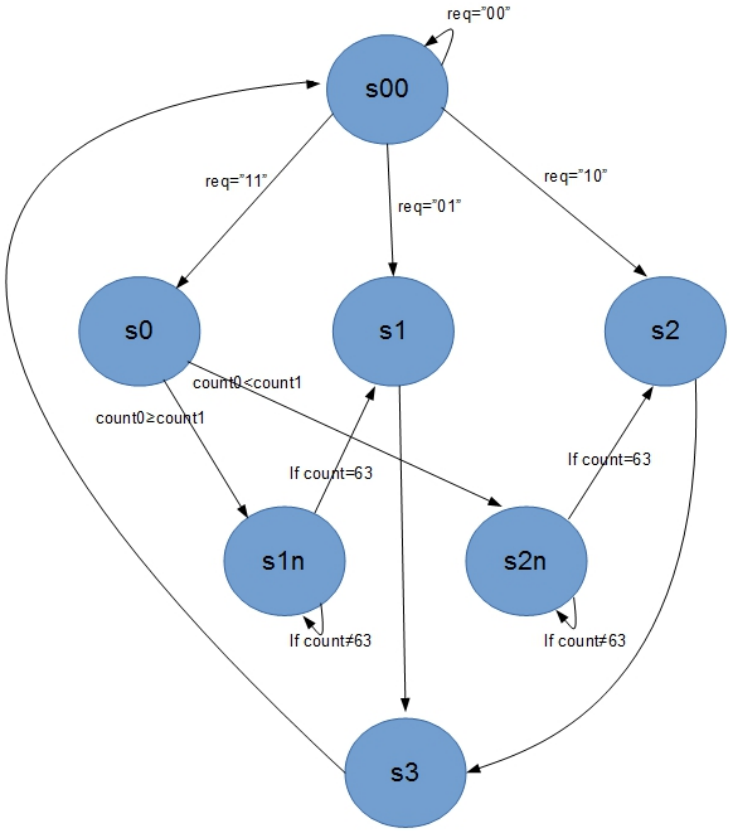
Το σχήμα αυτό αποτελείται από δυο μονάδες την req_to_pulse και έναν παράγει έναν παλμό που ακολουθεί την έξοδο του EES, σαν valid. Πρώτα το req_val του EES καθυστερείτε κατα έναν κύκλο (0.5s) ώστε να ακολουθεί την τωρινή τιμή και όχι την προηγούμενη. Μετά η έξοδος του register μπαίνει στην μονάδα req_to_pulse και αν η είσοδος είναι '1' δινει '1' για έναν κύκλο (20ns). Και τα δύο σήματα μαζί θα χρησιμοποιηθούν σαν ένα σήμα ελενχου για την μονάδα scheduler.

timest

Η μονάδα αυτή μετράει τα requests ενός επεξεργαστή. Με την λογική ότι έαν ένας επεξεργαστής βρίσκετε σε πιο κρίσιμη κατάσταση από τον αλλό θα έχει περισσότερα requests. Έτσι όποιος έχει μεγαλύτερη μέτρηση στον counter του timest θα έχει προτεραιότητα στο scheduling. Για να είναι κάπως πιο δίκαιο και να μην κερδίζει πάντα ο ένας, ο counter είναι 4-bit για να παθένει πιο εύκολα overflow και να προηγηθεί ο άλλος. Αν είναι σε emergency αυτός που έπαθε overflow θα προηγηθεί και πάλι είτε μετά από μερικές μετρήσεις (μιας και θα τις ζητάει συχνότερα) είτε απο overflow του άλλου.

scheduler

Η μονάδα αυτή επιλέγει ποιου επεξεργαστή η μέτρηση θα προηγηθεί. Σαν είσοδο δέχεται τη μέτρηση με το id του κάθε επεξεργαστή, τη μέτρηση των request (timestamp) κάθε επεξεργαστή και ένα σήμα ελέγχου (req). Το req είναι 2-bit και το κάθε bit αντιστοιχεί σε έναν επεξεργαστή, αν είναι '1' τότε ο αντίστοιχος επεξεργαστής ζητάει να στείλει τιμή. Διακρίνονται τέσσερεις περιπτώσεις. Αν 00 τότε δεν ζητάει κανείς να στείλει τιμή. Αν 01 ζητάει μόνο ο δεύτερος. Αν 10 μόνο ο πρώτος. Τέλος 11 ζητάνε και οι δύο. Στην κατάσταση s00 γίνεται ο παραπάνω έλεγχος και ορίζετε η επόμενη κατάσταση η s00, s1, s2 ή s0 αντίστοιχα με τις προηγούμενες περιπτώσεις. Στην περίπτωση που πάει στην s0 (11) τότε συγκρίνονται τα timestamps, αν προηγήτε ο πρώτος στην έξοδο δίνει την μέτρηση του με κάποιο valid (wr_e) και επόμενη κατάσταση την s1n, σε αντίθετη περίπτωση η έξοδος είναι η μέτρηση του δεύτερου και η επόμενη κατάσταση η s2n. Στις καταστάσεις s1n και s2n θα μείνει για μερικούς κύκλους για να προλάβει να εκτυπωθεί τη τιμή που πέρασε, και θα πάει στη κατάσταση s1 ή s2 αντίστοιχα. Η κατάσταση s1 δίνει σαν έξοδο την τιμή του δευτέρου επεξεργαστή με κάποιο valid και επόμενη κατάσταση την s3, μπορεί να βρεθεί κατευθείαν σε αυτήν την κατάσταση με req=01. Η κατάσταση s2 δίνει σαν έξοδο την τιμή του πρώτου επεξεργαστή με κάποιο valid και επόμενη κατάσταση την s3, μπορεί να βρεθεί κατευθείαν σε αυτήν την κατάσταση με req=10. Η s3 στέλνει στην κατάσταση s00.



binary_to_ascii
Η μια μονάδα σαν είσοδο δέχεται την τιμή του scheduler και η άλλη το timestamp, με κάποιο padding με μηδενικά. Σαν έξοδο έχουμε λέξεις που ανά 4-bit αναπαριστούν ένα δεκαδικό ψηφίο, δύο για το timestamp και τρία για την μέτρηση.

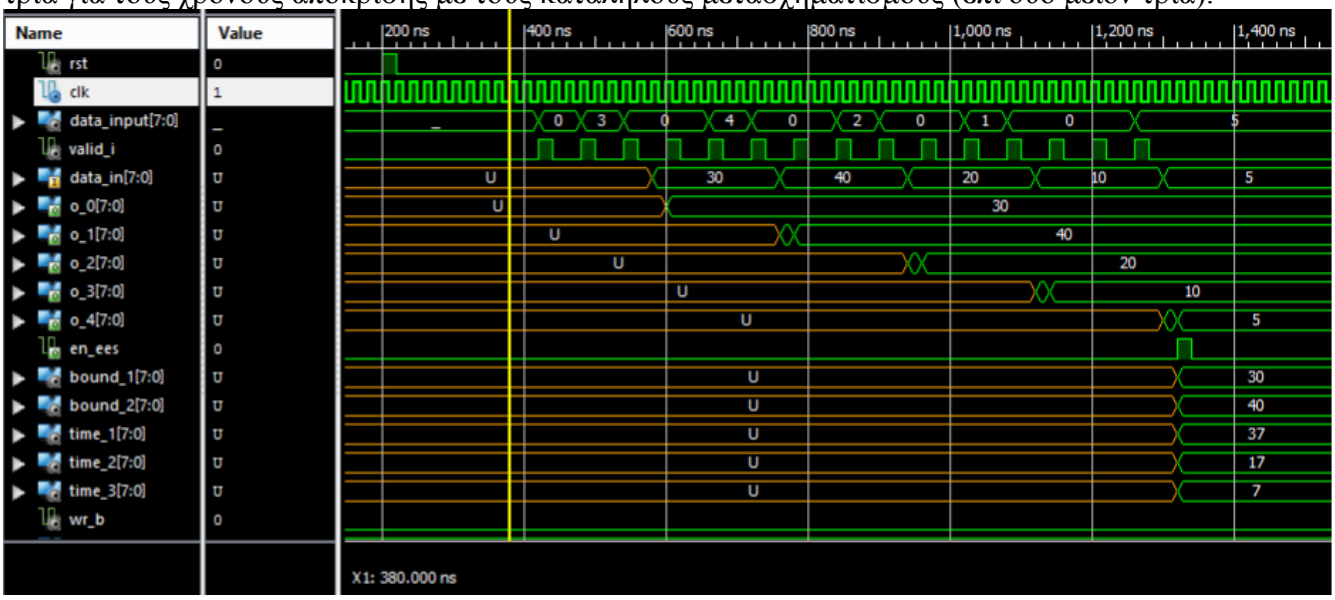
data_to_decimal
Η μονάδα αυτή δέχεται τις εξόδους από τις μονάδες binary_to_ascii και ανά δύο κύκλους παίρνει ένα κομμάτι 4-bit (ένα ψηφίο) το μετατρέπει σε ASCII (προσθέτει 48) και το στέλνει στην έξοδο με ένα σήμα valid. Η μορφή των εξόδων είναι id_mode_digit2_digit1_digit0_#_time1_time0_nl όπου η κάτω παύλα είναι νεκρός κύκλος, id ο επεξεργαστής, mode η κατάσταση της μέτρησης (low,high,emergency), digit2 το αριστερό ψηφίο της μέτρησης, digit1 το μεσαίο, digit0 το δεξί, # χωρίζει την μέτρηση από το timestamp, time1 το αριστερό ψηφίο του timestamp και time0 το δεξί.

uart_tx
Η μονάδα αυτή δέχεται σαν είσοδο τα 8-bit ψηφία από την data_to_ascii και αν το write_buffer είναι '1' (το valid της data_to_ascii) τότε γράφει την είσοδο στην fifo της και αρχίζει να μεταδίδει στην έξοδο ανά bit με τον ρυθμό που ορίζει το baud_timer.

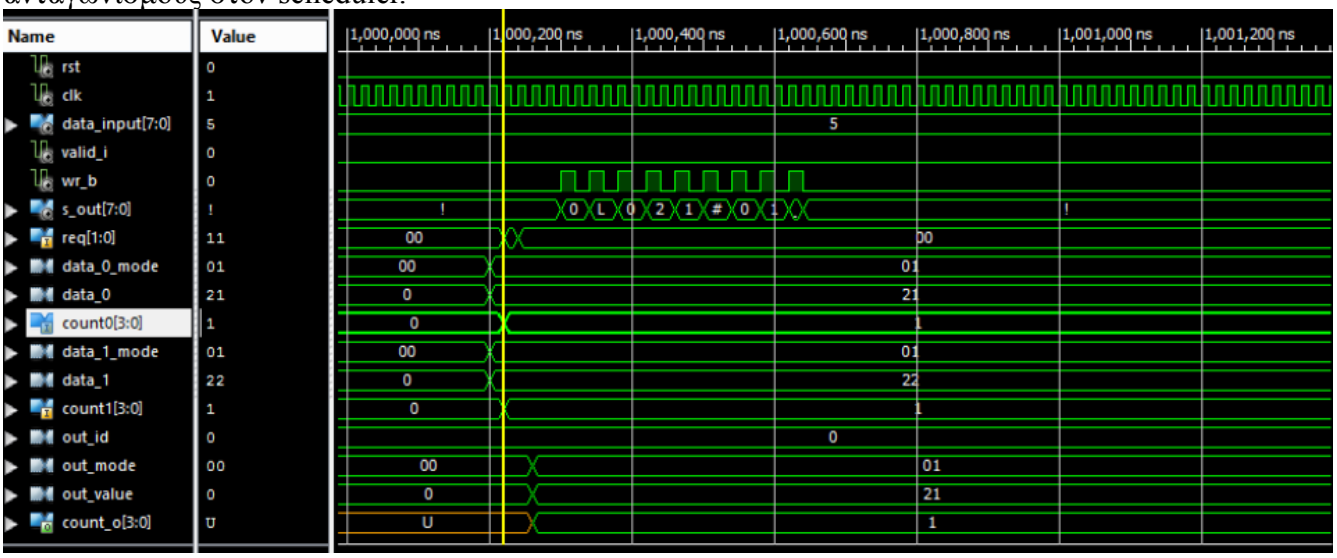
behavioral simulation (top_prev.vhd)

Το behavioral simulation παρουσιάζεται σε κλίμακα των ms, δηλαδή 1000 φορές κάτω από αυτό που ελένχθηκε στο εργαστήριο. Η προσομοίοση έγινε χωρις τις μονάδες uart, έτσι σαν είσοδο του κυκλώματος έχουμε ότι θα έδινε σαν έξοδο η μονάδα uart_rx (8-bit λέξη και 1-bit valid/data_present) και σαν έξοδο του κυκλώματος ότι θα δεχόταν σαν είσοδο η μονάδα uart_tx (8-bit λέξη και 1-bit valid/write_buffer).

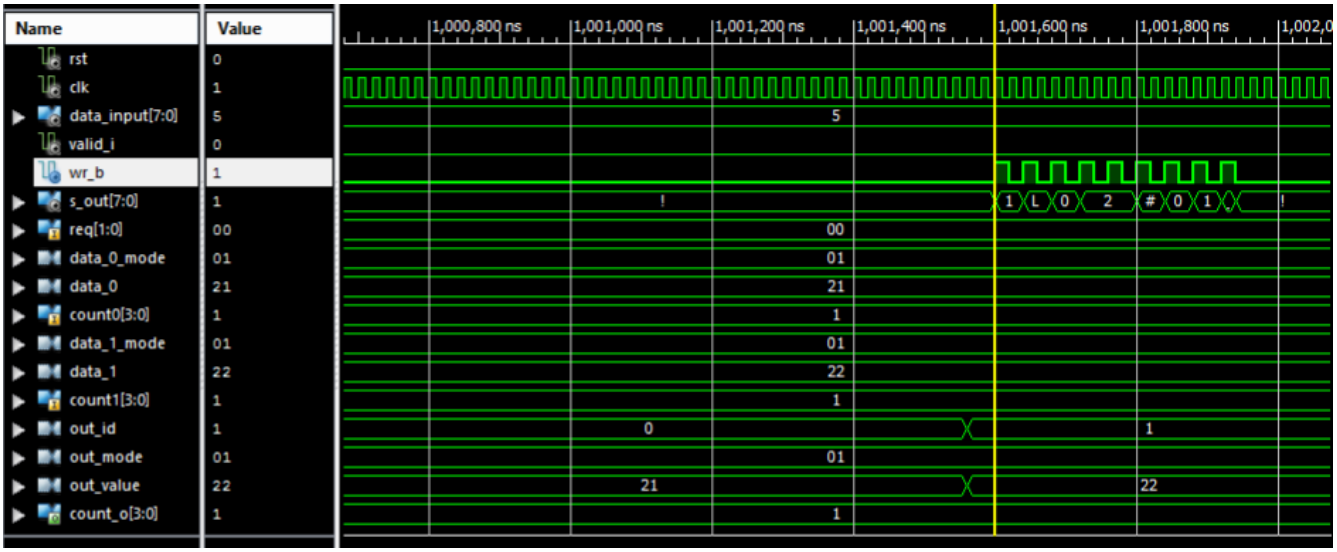
Στην πρώτη εικόνα βλέπουμε την αρχικοποίηση για τιμές 030 040 020 010 005. Τα σήματα o_0, o_1, o_2, o_3, o_4 είναι οι έξοδοι της μονάδας init που είναι τα όρια και το en_ees είναι το σήμα που θέτει τους επεξεργαστές σε κατάσταση αρχικοποίησης. Τα bound_1, bound_2, time_1, time_2, time_3 είναι τα latches που αποθηκεύουν οι επεξεργαστές τα όρια, τα δύο πρώτα είναι για θερμοκρασία και τα άλλα τρία για τους χρόνους απόκρισης με τους κατάληλους μετασχηματισμούς (επί δύο μείον τρία).



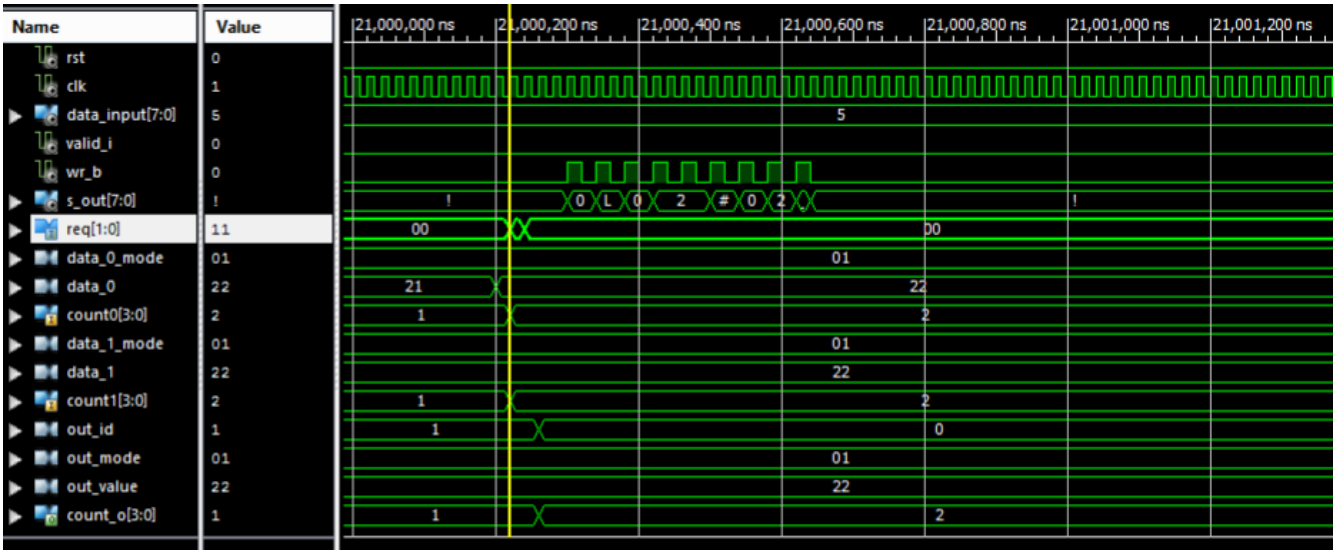
Στις επόμενες εικόνες φαίνεται η λειτουργεία του επεξεργαστή. s_out είναι η τιμή της εξόδου, wr_b είναι το valid που θα έπερνε το uart_tx για να γράψει στη fifo του. Το req είναι αυτό που δίχνει ποιός επεξεργαστής ζητάει να περάσει από τον scheduler Το data_0_mode είναι tomode του πρώτου επεξεργαστή, το data_0 είναι η μετρηση του πρώτου, count0 το timestamp του και data_0_mode, data_0, count0 τα αντίστοιχα του δευτέρου. Τα επόμενα σήμματα είναι η έξοδος του scheduler (το id του επεξεργαστή, το mode , η μέτρηση και το timestamp). Έτσι μπορούμε να δούμε τους ανταγωνισμούς στον scheduler.



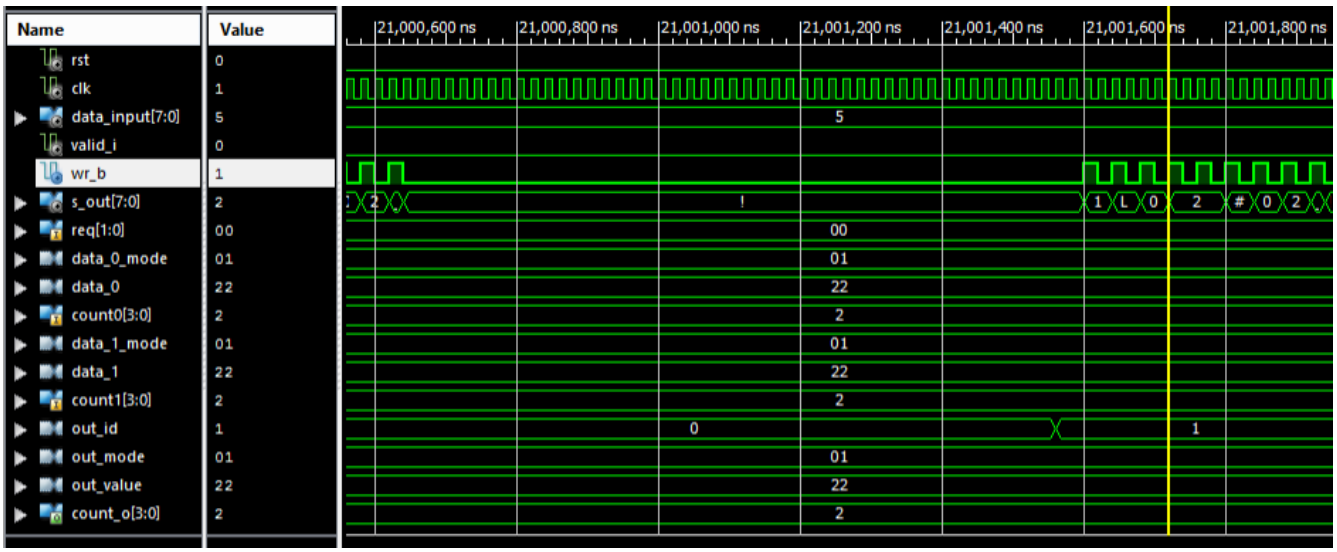
req=11 περνάει αρχικά ο πρωτος.



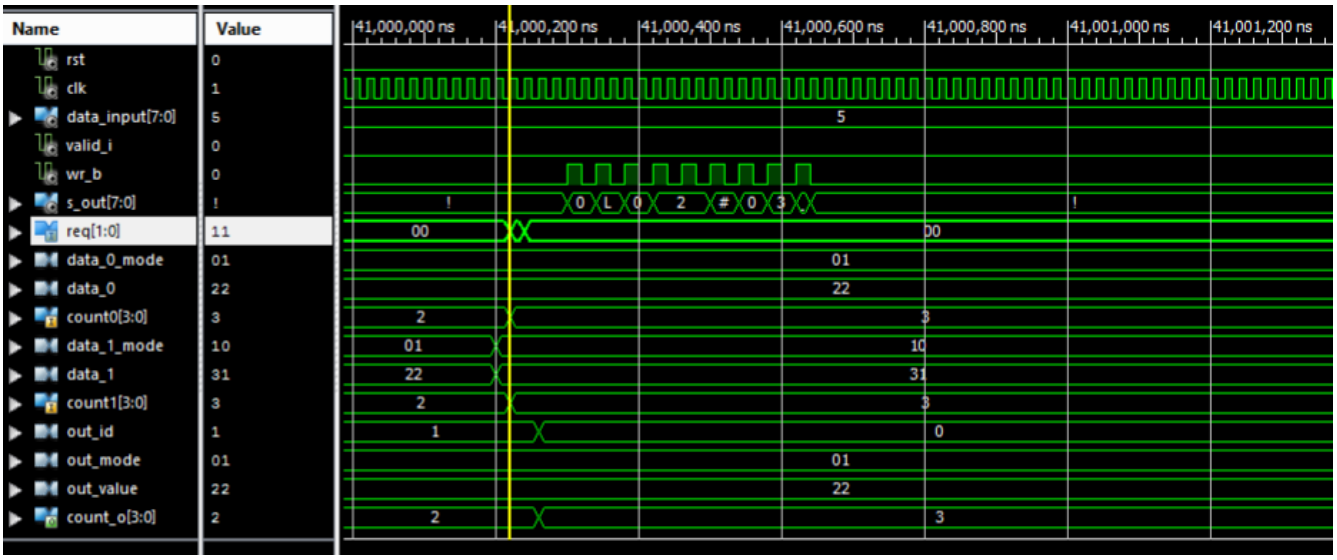
Μετά από λίγους κύκλους ο δεύτερος.



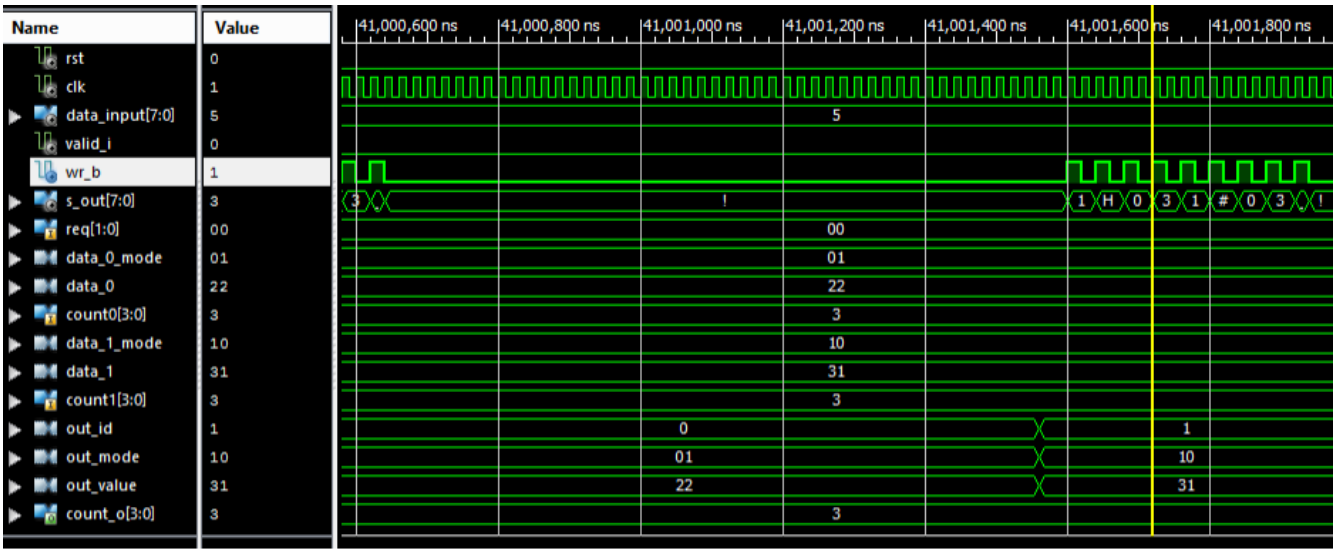
Μετά από 20ms πάλι έχουμε αναταγωνισμό και περνάει ο πρώτος (ίσα timestamps)



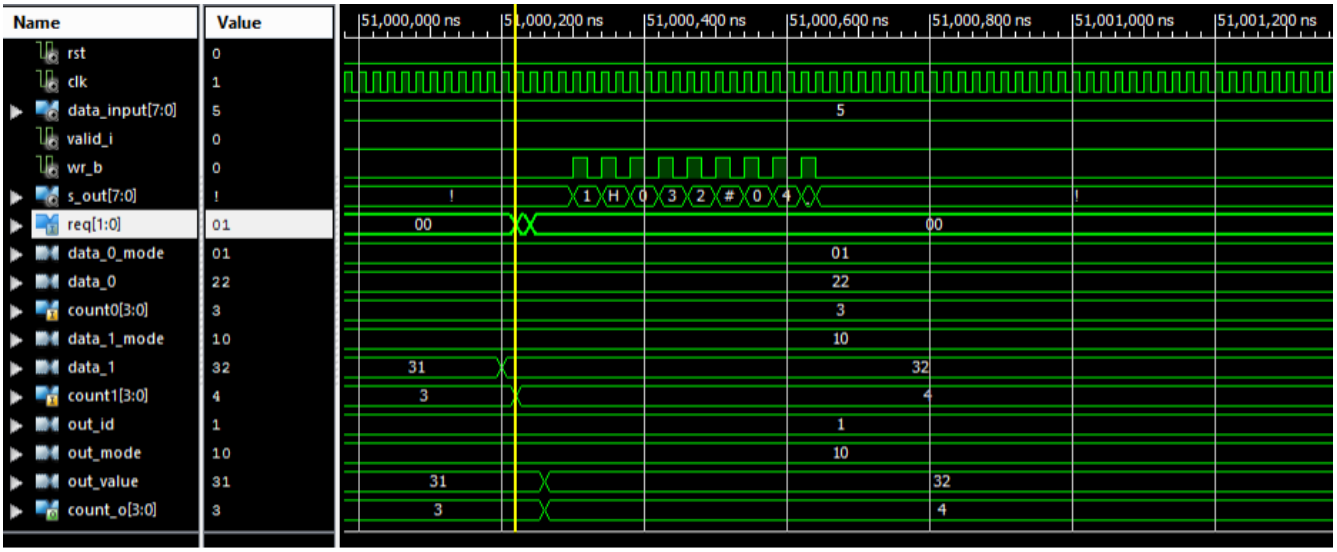
Μετά από λίγους κύκλους ο δεύτερος.



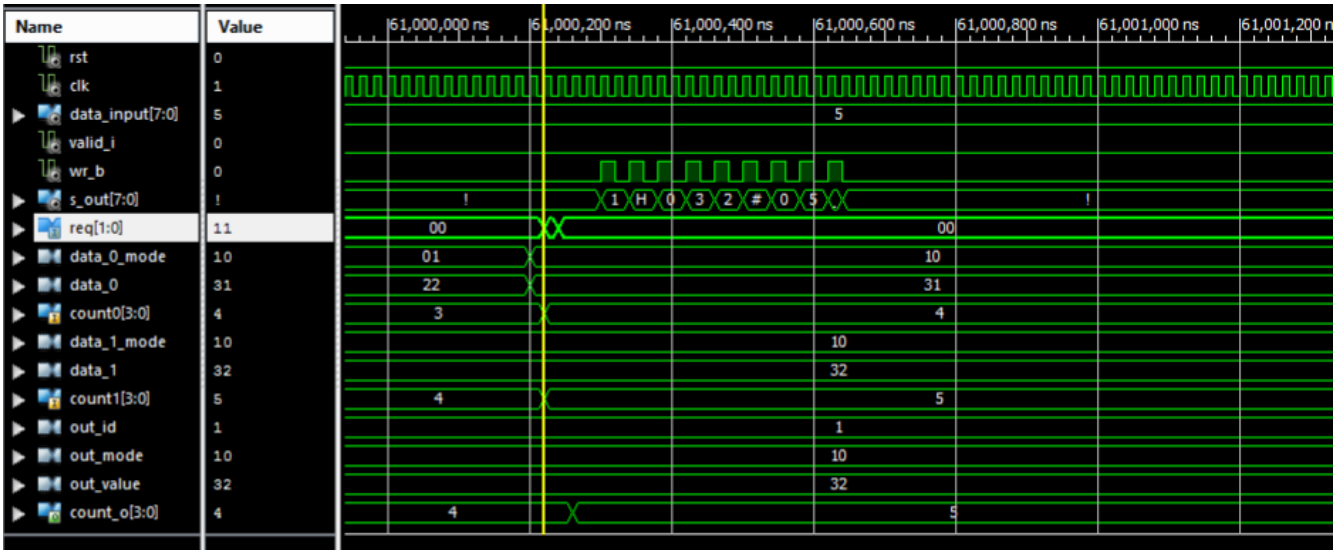
Μετά από 20ms πάλι έχουμε αναταγωνισμό και περνάει ο πρώτος (ίσα timestamps)



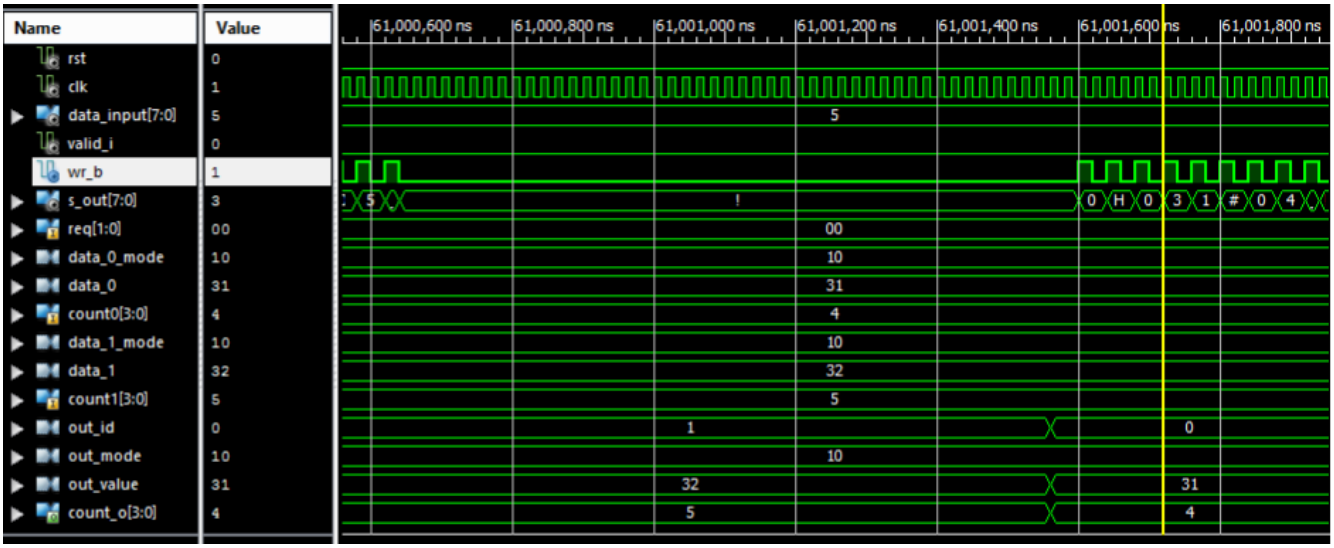
Μετά από λίγους κύκλους ο δεύτερος.



Μετά από 10ms ζητάει να περάσει μόνο ο δεύτερος (req=01).

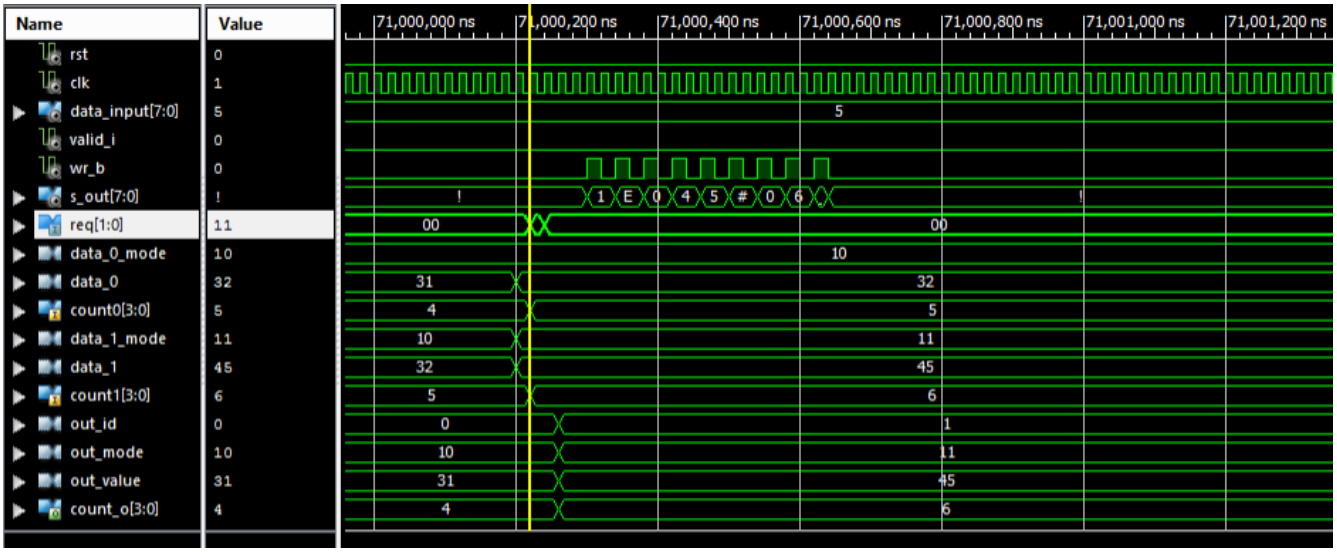


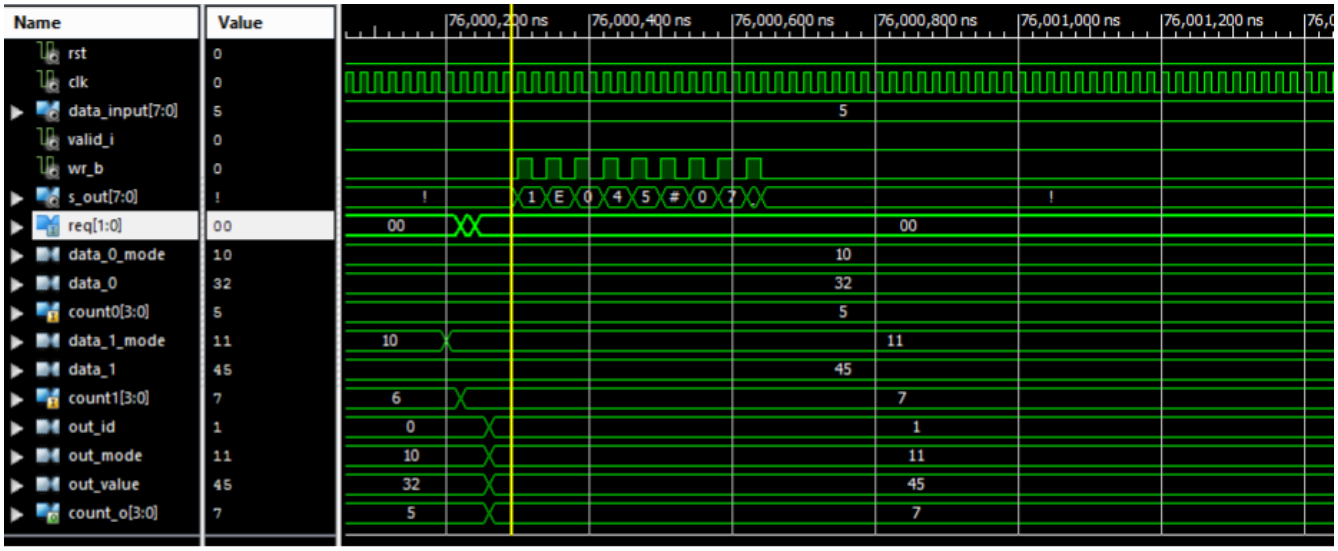
Μετά από 20ms έχουμε πάλι ανταγωνισμό και κερδίζει ο δεύτερος μιας και έχει μεγαλύτερο timestamp.



Μετά από λίγους κύκλους ο πρώτος.

Στην επόμενη μέτρηση ο δεύτερος μπήκε σε emergency, οι επόμενες δύο εικόνες δείνχουν ότι η αναμονή είναι 5ms, δεν δείχνουν ανταγωνισμο.


Name
Value
rst
clk
data_input[7:0]
valid_i
wr_b
s_out[7:0]
req[1:0]
data_0_mode
data_0
count0[3:0]
data_1_mode
data_1
count1[3:0]
out_id
out_mode
out_value
count_o[3:0]
71,000,000 ns
71,000,200 ns
71,000,400 ns
71,000,600 ns
71,000,800 ns
71,001,000 ns
71,001,200 ns

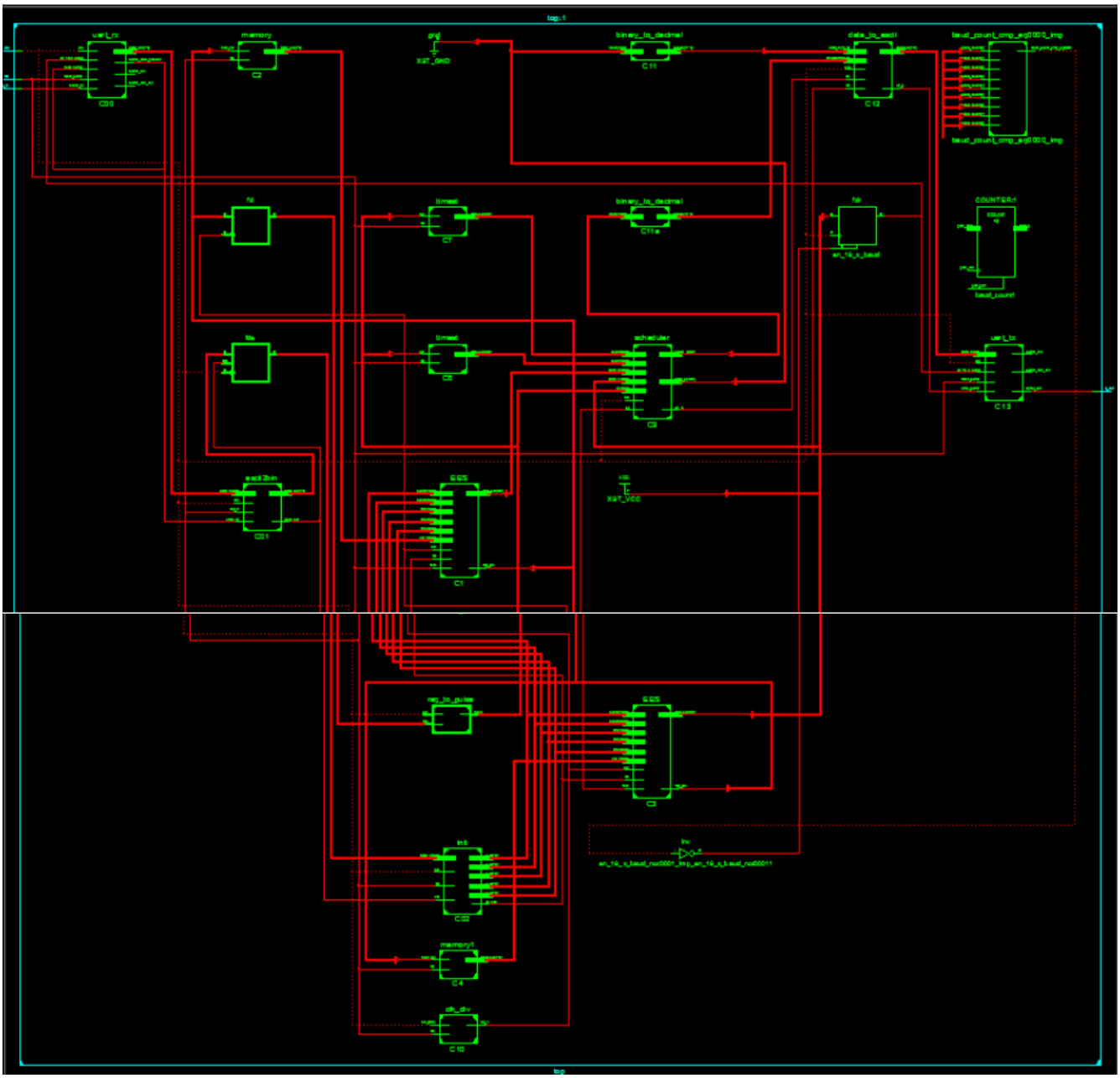

Name
Value
rst
clk
data_input[7:0]
valid_i
wr_b
s_out[7:0]
req[1:0]
data_0_mode
data_0
count0[3:0]
data_1_mode
data_1
count1[3:0]
out_id
out_mode
out_value
count_o[3:0]
76,000,200 ns
76,000,400 ns
76,000,600 ns
76,000,800 ns
76,001,000 ns
76,001,200 ns

Synthesize

```
Device utilization summary:
---------------------------

Selected Device : 3s200ft256-5

 Number of Slices:                      381  out of   1920    19%
 Number of Slice Flip Flops:            402  out of   3840    10%
 Number of 4 input LUTs:                687  out of   3840    17%
    Number used as logic:               651
    Number used as Shift registers:      36
 Number of IOs:                          4
 Number of bonded IOBs:                  4  out of    173     2%
 Number of GCLKs:                        3  out of      8    37%

Timing Summary:
---------------
Speed Grade: -4

   Minimum period: 8.597ns (Maximum Frequency: 116.320MHz)
   Minimum input arrival time before clock: 5.982ns
   Maximum output required time after clock: 7.165ns
   Maximum combinational path delay: No path found

=========================================================================
```

Η συχνότητα που θέλουμε 50MHz είναι μικρότερη απο αυτήν που προβλέπει ο synthesizer, άρα κατά πάσα πιθανότητα δεν θα έχει προβλήματα χρονισμού κατά το implementation.

Schematic



Το schematic μας έδωσε αυτό που περιμέναμε, όλα τα componet που περιγράφτηκαν στο top design. Συμπίπτει με το κύκλωμα που σχεδιάστηκε στην αρχή. fde είναι ο register πριν το init, fd οι registers πριν το req_to_pulse και fdr του baud_timer.

Implementation

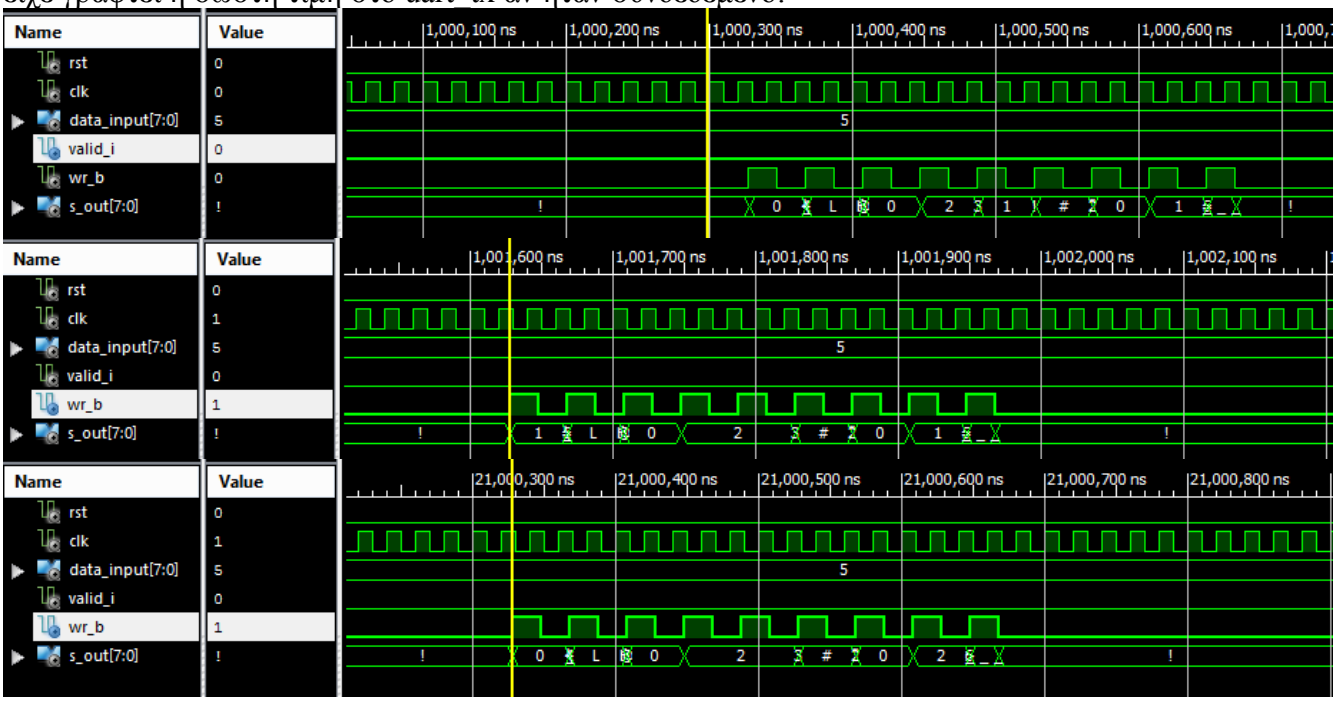
```
Device Utilization Summary:

   Number of BUFGMUXs                        3 out of 8      37%
   Number of External IOBs                   4 out of 173     2%
      Number of LOCed IOBs                   4 out of 4     100%

   Number of Slices                        417 out of 1920   21%
      Number of SLICEMs                     20 out of 960     2%
```

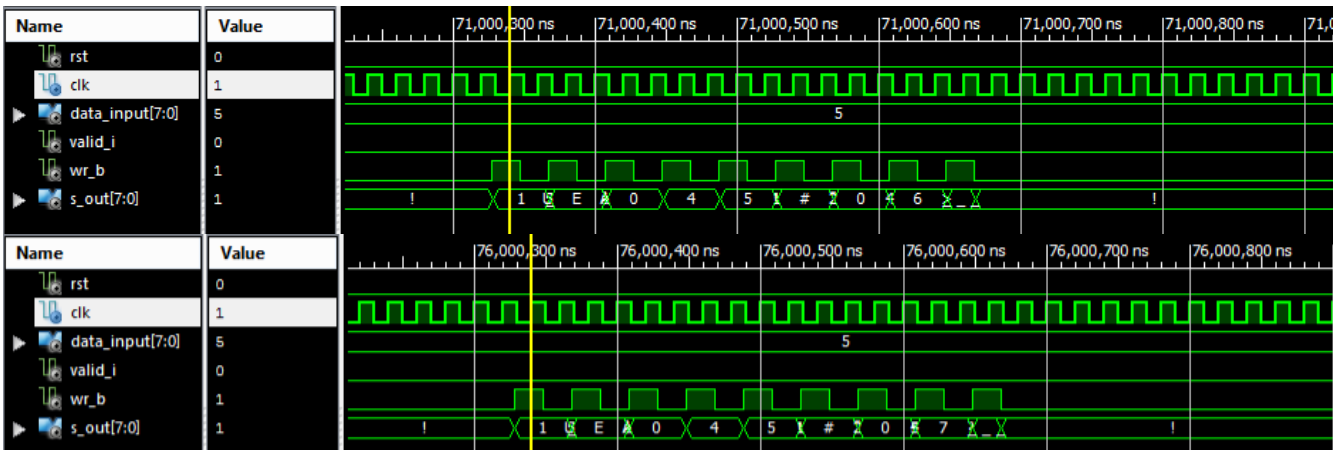
Constraint	Check	Worst Case Slack	Best Case Achievable	Timing Errors	Timing Score
NET "clk_BUFGP/IBUFG" PERIOD = 20 ns HIGH 50%	SETUP	12.383ns	7.617ns	0	0
	HOLD	0.719ns		0	0

Τα ποσοστά χώρου είναι τα αναμενόμενα και το ρολόι των 20ns που θέλαμε το έχουμε πιάσει.

post route simulation

Στις παρακάτω εικόνες φαίνεται ότι τα αποτελεσματα είναι ίδια με αυτά του behavioral, είναι τα αναμενόμενα στην κλίμακα των ms. Επίσης φαίνεται (ειδικά στις δύο τελευταίες στην κίτρινη γραμμή) ότι συνχρονίζονται το valid με την επιθυμητή τιμή και την άνωδο του ρολογιού, που σημαίνει ότι θα είχε γραφτει η σωστή τιμή στο uart_tx αν ήταν συνδεδεμένο.





Σημείωση:
Τα report χώρου και χρονισμού είναι για το top.vhd που έχει τις μονάδες uart και baud_timer, ενώ τα simulations είναι για το top_prev.vhd που δεν περιέχει τις μονάδες uart και baud_timer.