

# Εργαστηριακές Ασκήσεις

## CAD

Λουδάρος Ιωάννης (1067400)



Μπορείτε να δείτε την τελευταία έκδοση του Project [εδώ](#) ή σκανάροντας τον κωδικό QR που βρίσκεται στην επικεφαλίδα.

## Περιγραφή Αναφοράς

Παρακάτω παραθέτω τις απαντήσεις μου στις “Εργαστηριακές Ασκήσεις” του μαθήματος “Σχεδίαση Συστημάτων με χρήση Υπολογιστών (CAD)” καθώς και σχόλια τα οποία προέκυψαν κατά την εκπόνηση τους.

## Περιεχόμενα

1. Άσκηση 1.....	3
2. Άσκηση 2.....	4
Top-down Design : Carry Save Multiplier	4
Bottom-up Design : Register	5
3. Άσκηση 3.....	6
4. Άσκηση 4.....	7
Υπολογισμός Μέγιστης Συχνότητας	8
Εξομοίωση	8
Φυσική Υλοποίηση	9
5. Άσκηση 5.....	10
Παραμετροποιημένος Μετρητής	10
8-bit Μετρητής	10
Εξομοίωση Κυκλώματος	10
Υλοποίηση με Σύγχρονο Clear	11
Εισαγωγή Καθυστερήσης	11
6. Άσκηση 6.....	12
Απεικόνιση αριθμού	12

Πολλαπλασιασμός Αριθμών	12
Unsigned Multiplier	12
Signed Multiplier	12
7. Άσκηση 7 .....	13
Stopwatch	13
Μέτρηση Δεκάτων	13
8. Άσκηση 8 .....	14
Λειτουργία Προηγούμενου Πλήκτρου	14
Λειτουργία Calculator	14

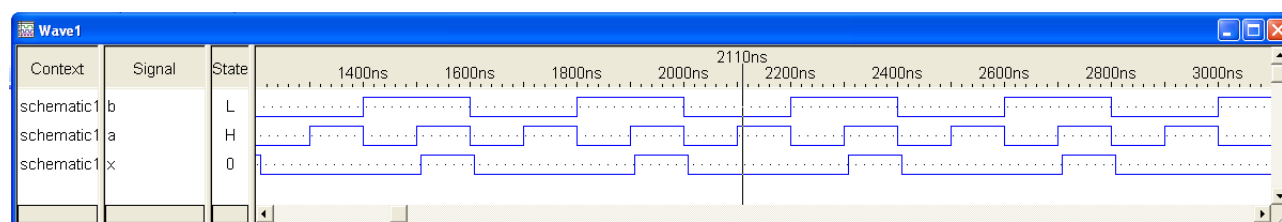
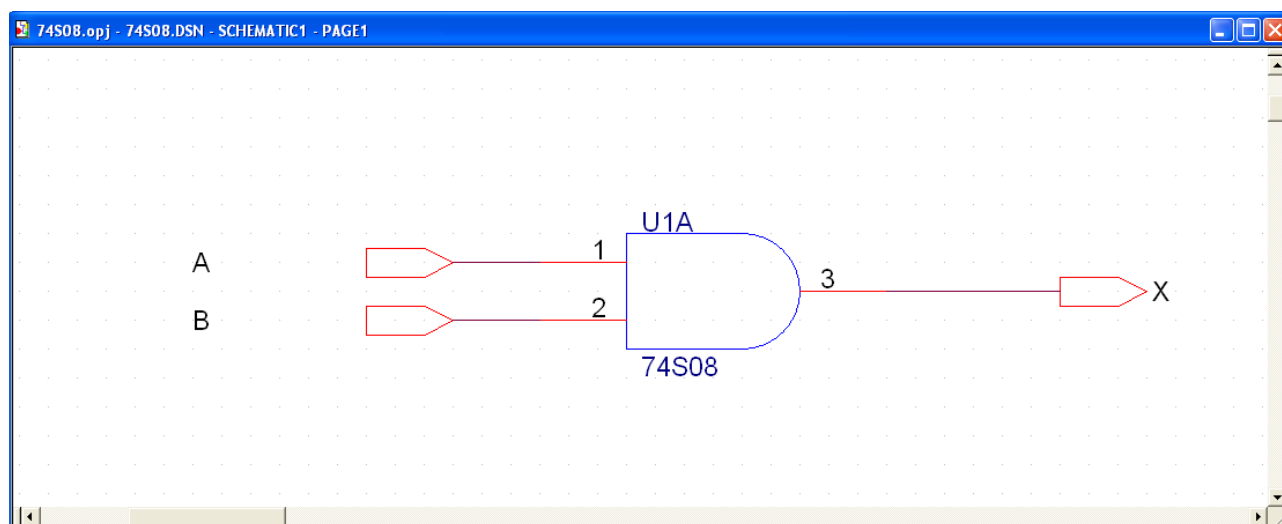
# Απαντήσεις

## 1. Άσκηση 1

Οι διαφορετικές υλοποιήσεις βρίσκονται σε ξεχωριστά project files στον κατάλογο "1" που συνοδεύει την παρούσα αναφορά. Οι καθυστερήσεις παρουσιάζονται παρακάτω και επιβεβαιώνονται από τα αντίστοιχα αρχεία εξομίωσης των ολοκληρωμένων.

	7408	74ALS08	74AS08	74S08	74LS08
Χρόνος καθυστέρησης μετάβασης στο λογικό 0	27 ns	9 ns	4 ns	8 ns	13 ns
Χρόνος καθυστέρησης μετάβασης στο λογικό 1	27 ns	9 ns	4 ns	8 ns	13 ns

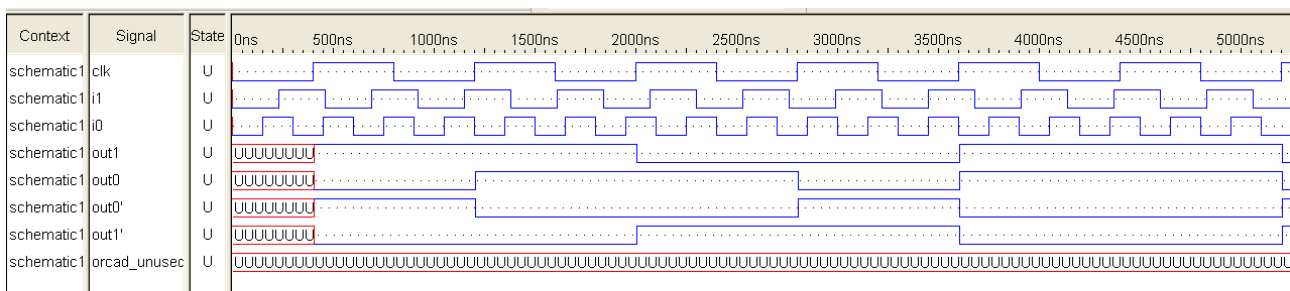
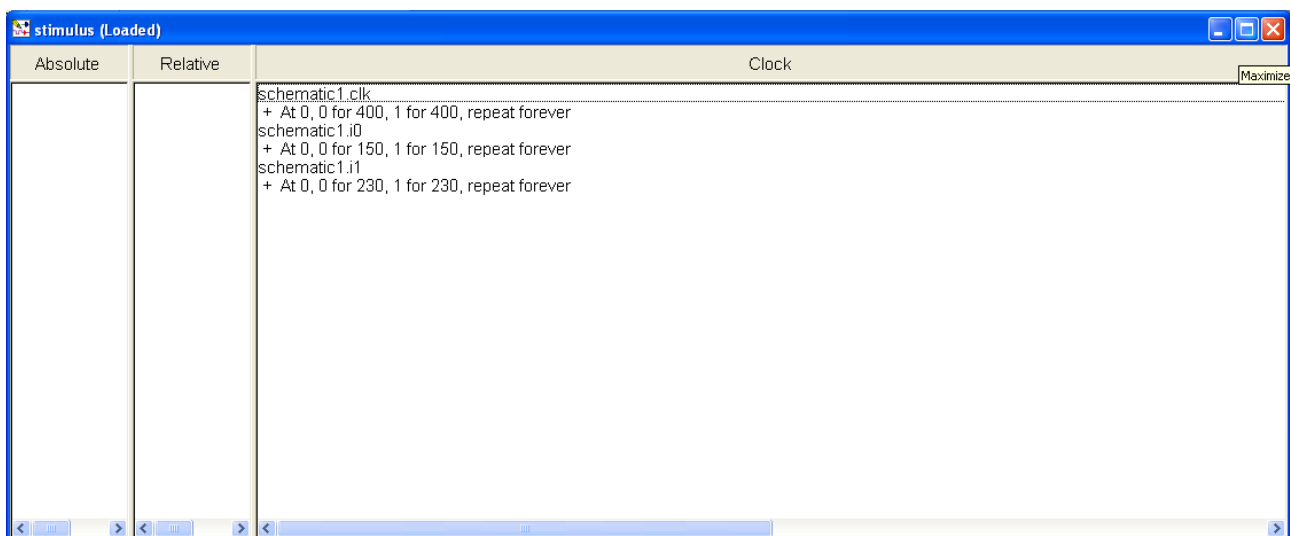
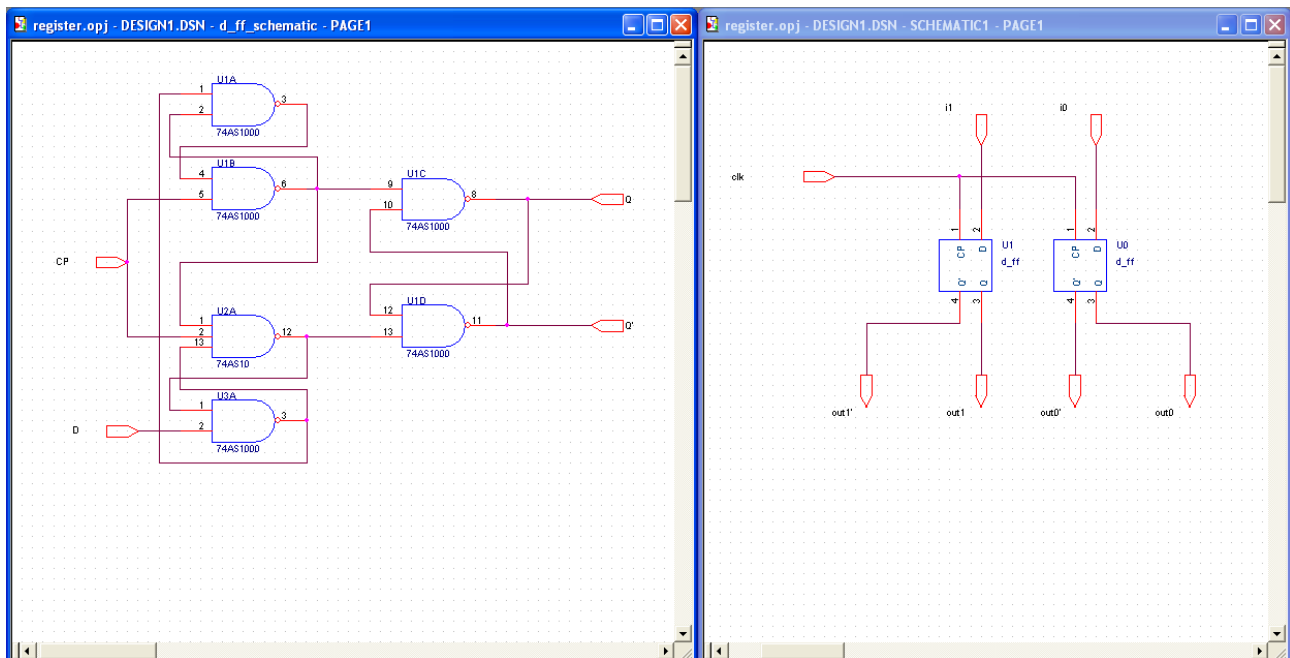
Ενδεικτικά παραθέτω screenshots από την υλοποίηση με το ολοκληρωμένο 74S08.





## Bottom-up Design : Register

Μπορείτε να βρείτε το project file του σχεδιασμού μας στον κατάλογο “2/Register” που συνοδεύει την παρούσα αναφορά. Η υλοποίηση μας, το stimulus και οι κυματομορφές που προέκυψαν από αυτό, εμφανίζονται παρακάτω:







## Υπολογισμός Μέγιστης Συχνότητας

Για να υπολογίσω την μέγιστη συχνότητα κάνω την υπόθεση ότι η θύρα μπορεί να διαβάσει τα αποτελέσματα του κυκλώματος ακαριαία.

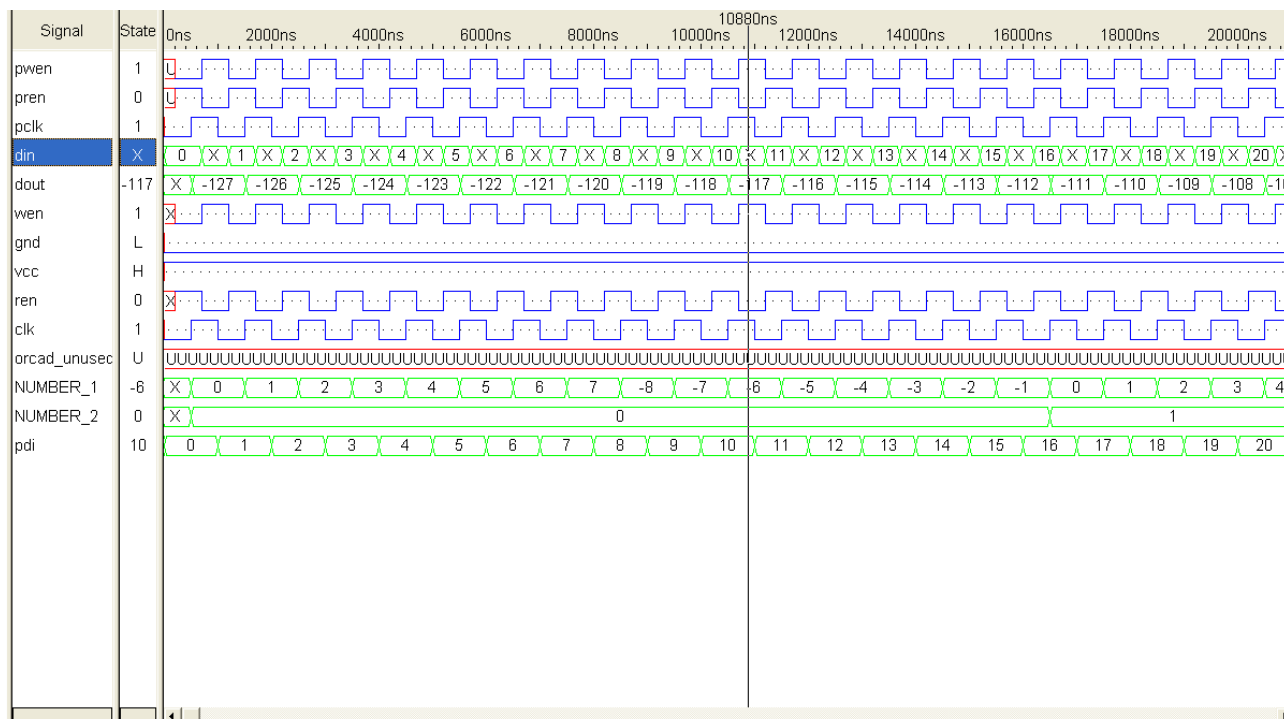
Η συλλογιστική είναι απλή. Παίρνοντας πληροφορίες από την προσομοίωση βλέπουμε ποια είναι η καθυστέρηση που έχει η διαδρομή από την είσοδο μέχρι την έξοδο ώστε να βρούμε την ελάχιστη περίοδο.

Module	Πλήθος	Καθυστέρηση
74ACT244	3	4 ns
74ACT175	1	2 ns
74ACT283	2	16 ns
Σύνολο	6	46 ns

Άρα χρειαζόμαστε 46 ns για να γίνει ο υπολογισμός που θέλουμε. Το κύκλωμα λοιπόν έχει θεωρητική μέγιστη συχνότητα:

$$\frac{1}{46 \cdot 10^{-9}} \approx 0.0217 \cdot 10^9 \text{Hz ή αλλιώς } 21.7 \text{ MHz.}$$

## Εξομοίωση

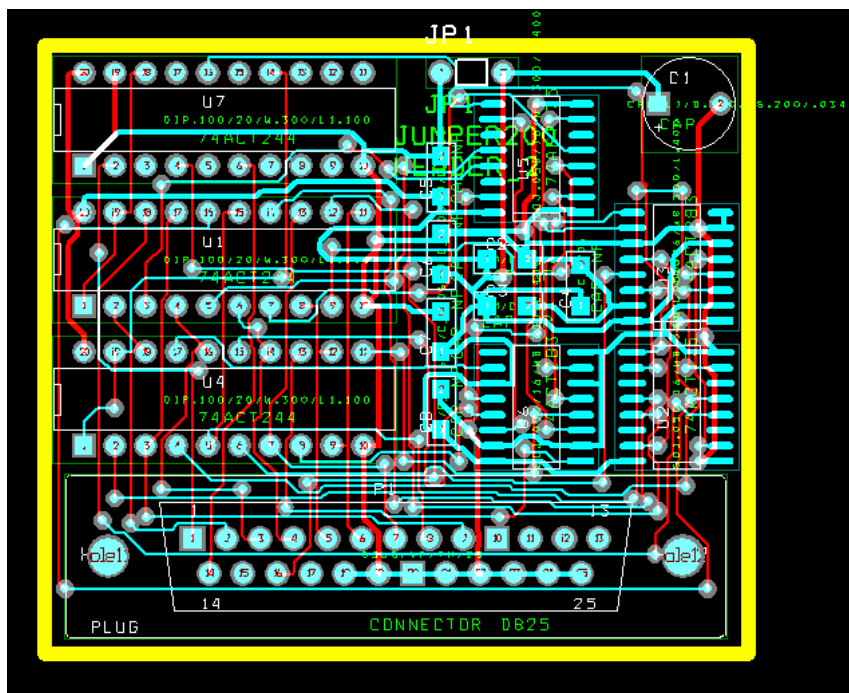


Οι κυματομορφές που αποδεικνύουν την ορθή λειτουργία του κυκλώματος.



## Φυσική Υλοποίηση

Παρακάτω παραθέτω τις υλοποιήσεις που προέκυψαν. Μπορείτε να βρείτε τα εν λόγω αρχεία μέσα στους καταλόγους “4/Layout” και “4/Layout2”.

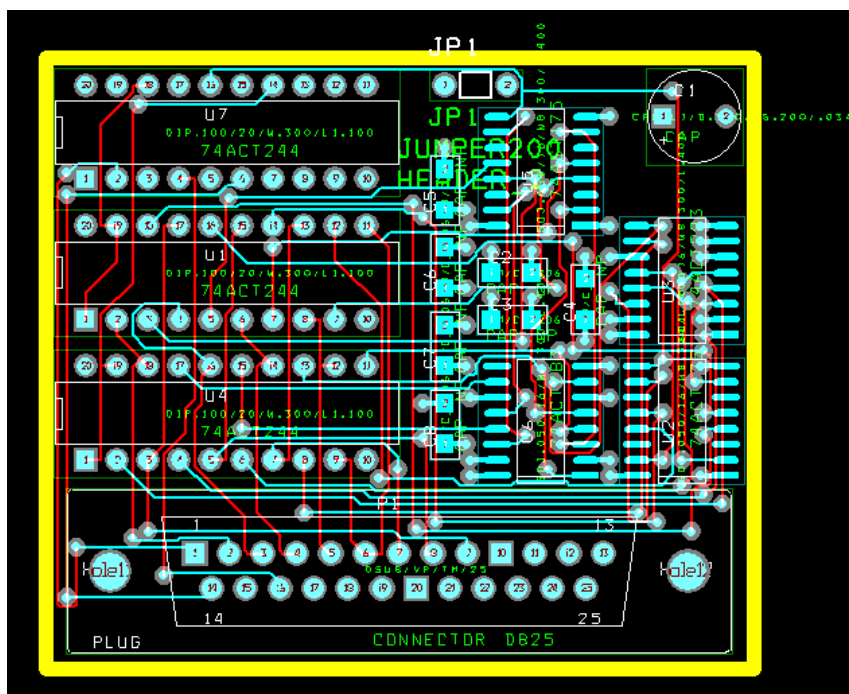


Υλοποίηση με 2 επίπεδα.

Οι διαφορές μεταξύ των δύο layout είναι προφανείς με γυμνό μάτι.

Έχουμε καταφέρει να φτιάξουμε μια καθαρότερη πλακέτα. Αυτό μπορεί να οδηγήσει σε ένα πιο αξιόπιστο κύκλωμα, αφού με αυτόν τον τρόπο μειώνουμε το ενδεχόμενο να προκληθούν λάθη μέσω του ηλεκτρομαγνητικού θορύβου.

Οι διαφορές στις τεχνικές routing προκύπτουν από την ανάγκη των through-hole συσκευών, να υπάρχουν μεγάλες επιχαλκωμένες οπές για την σύνδεση τους. Αυτές τις οπές, μπορούμε να δούμε ότι τις χρησιμοποιούν τα layers GND και POWER ώστε να ενωθούν κατευθείαν με το ολοκληρωμένο.



Υλοποίηση σε 4 επίπεδα.

Αντίθετα, για την ένωση με τα surface-mount ολοκληρωμένα, χρησιμοποιείται, πέρα από την “vías” για να ενωθούν τα αντίστοιχα layers, επιπλέον καλωδίωση στα επίπεδα 1 και 2.

Αν το εργαλείο επιχειρούσε να συνδέσει απευθείας τα επίπεδα GND και POWER με τα ολοκληρωμένα, θα καταλήγαμε σε βραχυκυκλώματα, και αυτό γιατί έχουμε δώσει εντολή στο εργαλείο μας να χρησιμοποιεί γραμμές μεγέθους 508 για την τροφοδοσία και για την γη. Αυτές οι γραμμές είναι πολύ μεγάλες για τα λεπτεπίλεπτα ποδαράκια για το soldering των surface mount ολοκληρωμένων.

## 5. Άσκηση 5

### Παραμετροποιημένος Μετρητής

Σχεδιάστηκε ένας παραμετροποιημένος μετρητής με default τα 4 bit.

Μπορείτε να βρείτε τον κώδικα που παράχθηκε πατώντας το παρακάτω κουμπί.

counter.v

### 8-bit Μετρητής

Ο ζητούμενος μετρητής δημιουργήθηκε χρησιμοποιώντας ένα high και ένα low order nibble των 4ων bit. Η λογική που χρειάστηκε να προστεθεί είναι για την είσοδο επίτρεψης του high order nibble. Συγκεκριμένα, η είσοδος επίτρεψης οδηγείται από το λογικό “και” όλων των bit εξόδου του low order nibble μαζί με το enable όλου του κυκλώματος.

counter\_8bit.v

### Εξομοίωση Κυκλώματος

```
# Loading project Counter
vsim work.counter_tb
# vsim work.counter_tb
# Loading work.counter_tb
# Loading work.counter_8bit
# Loading work.counter
# SymInitialize: Invalid parameter.
#
#
# Load canceled
run-all
# Load works correctly
# Clear works correctly
# Count works correctly
# Enable works correctly
# ** Note: $finish() : C:/users/iloudaros/Documents/Projects/My Projects/CEID/CAD-Exercises-and-Project/Exercises/5/counter_tb.v(79)
# Time: 5261 ns Iteration: 0 Instance: ./counter_tb
# 1
# Break at C:/users/iloudaros/Documents/Projects/My Projects/CEID/CAD-Exercises-and-Project/Exercises/5/counter_tb.v line 79
VSIM 3>
```

Το testbench μας κάνει μια σειρά ελέγχων που αφορούν την λειτουργία του load, του clear και του enable.

Εκτυπώνει στο τερματικό μας μια σειρά μηνυμάτων για να μας ενημερώσει για την σωστή λειτουργία του κυκλώματος μας και μας παράγει τις κατάλληλες κυματομορφές ώστε να επιβεβαιώσουμε ότι το κύκλωμα λειτουργεί σωστά.

Μπορείτε να βρείτε το testbench πατώντας το παρακάτω κουμπί.

counter\_tb.v

## Υλοποίηση με Σύγχρονο Clear

Η μόνη αλλαγή που χρειάζεται για να κάνουμε το clear σύγχρονο, είναι να το αφαιρέσουμε από το sensitivity list του always.

Μπορείτε να δείτε τον μετρητή και το αντίστοιχο testbench πατώντας τα παρακάτω κουμπιά.

[counter\\_sync\\_clear.v](#)

[counter\\_sync\\_clear\\_tb.v](#)

## Εισαγωγή Καθυστερήσης

Η μέγιστη συχνότητα λειτουργίας του μετρητή διαμορφώνεται στο 1/20, αφού η λογική του enable που έχουμε προσθέσει δεν προσθέτει καθυστέρηση σύμφωνα με την εκφώνηση.

Μπορείτε να δείτε τον μετρητή πατώντας το παρακάτω κουμπί.

[counter\\_delay.v](#)

## 6. Άσκηση 6

### Απεικόνιση αριθμού

Δημιουργήσαμε ένα module το οποίο απεικονίζει προσημασμένους αριθμούς χρησιμοποιώντας το 7-segment του XSA. Δέχεται έναν προσημασμένο αριθμό των 5 bit και παράγει μια απεικόνιση των 8 bit ώστε να φωτιστούν τα κατάλληλα λαμπάκια του 7-segment.

Μπορείτε να δείτε την περιγραφή του module καθώς και το αρχείο διασύνδεσης πατώντας τα παρακάτω κουμπιά.

[btd.v](#)[btd.ucf](#)

Η μέγιστη συχνότητα λειτουργίας είναι περίπου 78 MHz.  
(To time analyser επιστρέφει μέγιστη καθυστέρηση 12.816 ns)

### Πολλαπλασιασμός Αριθμών

Για αυτά τα υποερωτήματα χρησιμοποιούμε το προηγούμενο μας module ώστε να αναπαραστήσουμε το αποτέλεσμα του πολλαπλασιασμού των δύο μας εντέλων.

---

#### Unsigned Multiplier

Μπορείτε να δείτε την περιγραφή του module καθώς και το αρχείο διασύνδεσης πατώντας τα παρακάτω κουμπιά.

[unsigned\\_multiplier.v](#)[unsigned\\_multiplier.ucf](#)

Η μέγιστη συχνότητα λειτουργίας είναι περίπου 77.9 MHz.  
(To time analyser επιστρέφει μέγιστη καθυστέρηση 12.832 ns)

---

#### Signed Multiplier

Μπορείτε να δείτε την περιγραφή του module καθώς και το αρχείο διασύνδεσης πατώντας τα παρακάτω κουμπιά.

[signed\\_multiplier.v](#)[signed\\_multiplier.ucf](#)

Η μέγιστη συχνότητα λειτουργίας είναι περίπου 76.4 MHz.  
(To time analyser επιστρέφει μέγιστη καθυστέρηση 13.082 ns)

## 7. Άσκηση 7

### Stopwatch

Προσθέσαμε την λειτουργία του stopwatch εισάγοντας ένα καινούργιο module. Η λειτουργία αυτού του module είναι απλή: Αφήνει να περάσει ένα bus από μέσα του, δειγματοληπώντας την τιμή του σε κάθε θετική ακμή του ρολογιού. Σε περίπτωση όμως που του δωθεί το σήμα pause, οδηγεί αυτό το bus, με την τιμή που αυτό είχε κατά την τελευταία δειγματοληψία.

Το σήμα pause ελέγχεται από τον δεύτερο διακόπτη.

Μπορείτε να δείτε όλα τα αρχεία του σχεδιασμού μας ή να μεταβείτε κατευθείαν στο καινούργιο module χρησιμοποιώντας τα παρακάτω κουμπιά.

Stopwatch

holder.v

Το module μας συνδέθηκε μεταξύ του second counter και των bin\_2\_7.

### Μέτρηση Δεκάτων

Περνάμε το ρολόι μας μέσα από τους πρώτους τρεις υπάρχοντες mod25 counters. Έτσι έχουμε ένα ρολόι  $100 \cdot 10^6 \div 25^3 = 64 \cdot 10^2$ . Για να φτάσουμε στα επιθυμητά 10 Hz λοιπόν, αρκεί να προσθέσουμε έναν μετρητή των 6 bit (mod64) και έναν τον 4 bit (mod10).

Σύμφωνα με τα παραπάνω, περιγράψαμε τα module cnt6b και count10 και τα προσθέσαμε εντός του OneHertz module, το οποίο πλέον έχει μια επιπλέον έξοδο, την en\_next\_tenth. Επίσης, προσαρμόσαμε το module secondcounter, ώστε να μετράει και δέκατα του δευτερολέπτου τα οποία εξάγει μέσω της καινούργιας πόρτας st. Πρόσθετα, χρειάστηκε να μεγαλώσουμε τον holder μας ώστε να κρατάει και τα δέκατα του δευτερολέπτου. Τέλος, ενημερώσαμε το ucf αρχείο ώστε να συνδέσουμε την υλοποίηση μας με τα λεντακια του XST.

Μπορείτε να δείτε τα προσαρμοσμένα modules και ucf μέσω των παρακάτω κουμπιών:

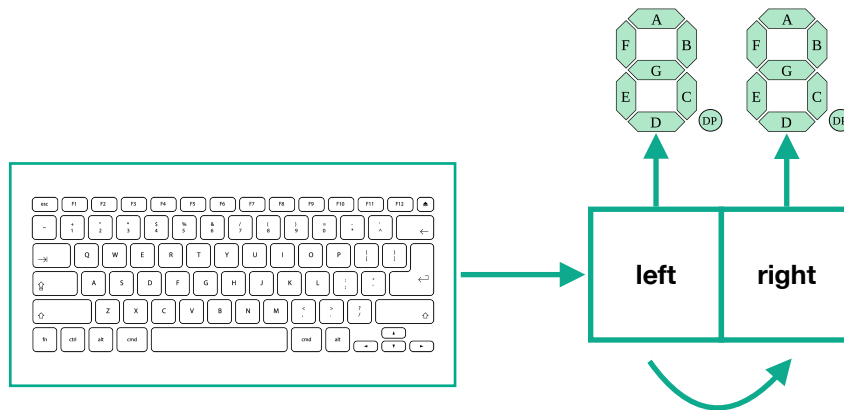
Precise Timer

tops.ucf

## 8. Άσκηση 8

### Λειτουργία Προηγούμενου Πλήκτρου

Το FSM μας τώρα αντί να εξάγει απλά το scancode το τροφοδοτεί σε ένα log από scancodes. Το log αυτό είναι των 2 θέσεων, κάθε φορά που βρίσκεται ένας αριθμός, εισάγεται στην αριστερή θέση και αυτός που βρισκόταν εκεί, μετακινείται στην δεξιά. Σχηματικά αυτό φαίνεται παρακάτω:



Οι μόνες αλλαγές που χρειάστηκε να κάνουμε είναι :

1. Να προσθέσουμε στο kbd\_protocol ένα σήμα ready που μας ενημερώνει για το πότε ανιχνεύτηκε πάτημα κουμπιού
2. Να προσθέσουμε ένα module το οποίο λειτουργεί ως log των σκαναρισμένων scancodes. Δέχεται για εισόδους, το σήμα ready και το scancode που ανίχνευσε το kbd\_protocol. Έχει για εξόδους το right και το left που ενημερώνονται όπως φαίνεται στο παραπάνω σχήμα.
3. Οι έξοδοι του log μας, ενώνονται η κάθε μια σε ένα scan\_2\_7seg που οδηγούν τα 7 segment της XST. (Έχουμε ενημερώσει το ucf αντίστοιχα).

keyboard\_scanner\_with\_log

### Λειτουργία Calculator

Χρειάστηκε να σχεδιάσουμε μια στοιχειώδη alu για αυτό το ερώτημα. Τα δύο μας έντελα, τροφοδοτούνται μόνιμα στην alu, αφού πρώτα περάσουν μέσα από δύο scan\_2\_binary modules. Η alu, χρησιμοποιώντας για opcode το τελευταίο scan, εκτελεί την πράξη, το αποτέλεσμα της οποίας, περνάει μέσα από ένα bin\_2\_7seg και τελικά τυπώνεται στο 7seg της XSA. Άξιο σημείωσης είναι ότι αφού δεν μας προσφέρεται υλοποίηση για τον operator της διαίρεσης, σχεδιάσαμε έναν μικρό υποσχεδιασμό που την υπολογίζει.

Επιπλέον το kbd\_protocol module μας έχει πλέον την δυνατότητα να αναγνωρίζει πότε σκανάρεται κάποιος τελεστής. Σε αυτή την περίπτωση το σήμα is\_number γίνεται 0. Το log δεν ενημερώνεται και η έξοδος του bin\_2\_7seg module περνάει στο 7\_seg. Πρόσβαση σε όλα τα παραπάνω μπορείτε να έχετε από το παρακάτω κουμπί:

Calculator