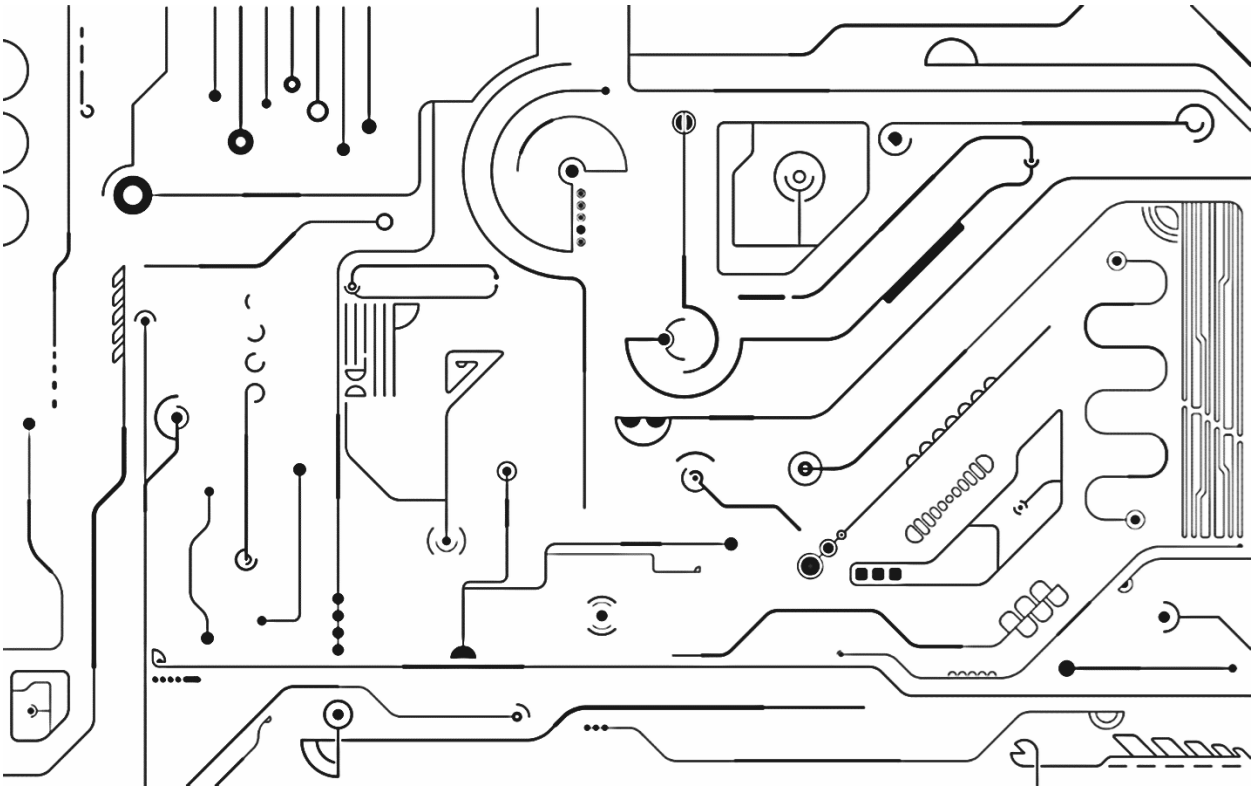
ΑΝΑΦΟΡΑ 3ης ΕΡΓΑΣΤΗΡΙΑΚΗΣ ΑΣΚΗΣΗΣ
LAB REPORT 03

Εργαστήριο Μικροεπεξεργαστών & Υλικού

Πέμπτη 10 Νοεμβρίου 2016

9:00-11:00



ΕΠΑΝΑΠΡΟΓΡΑΜΜΑΤΙΖΟΜΕΝΑ ΚΥΚΛΩΜΑΤΑ
PAL/GAL & PLDShell (HDL)

Μιχάλης Γαλάνης, Νίκος Πήλιουρας

ΟΜΑΔΑ Β' (LAB10130614)

Σκοπός του Εργαστηρίου

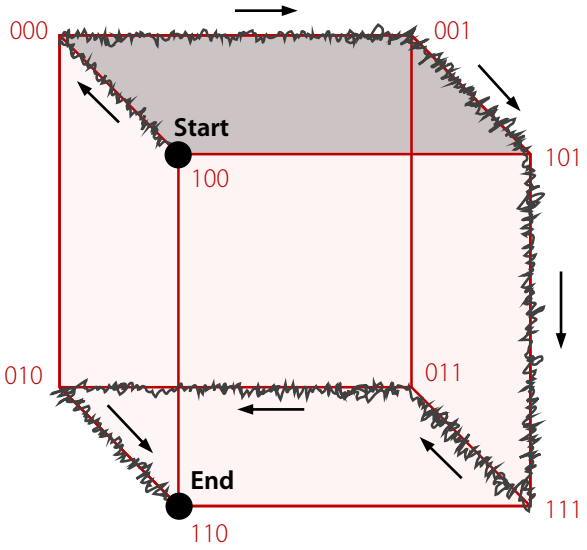
Σκοπός αυτής της εργαστηριακής άσκησης ήταν η εξοικείωση με τη γλώσσα **PLDShell** η οποία επιτρέπει τον προγραμματισμό επαναπρογραμματιζόμενων ολοκληρωμένων κυκλωμάτων **(Hardware Description Language - HDL)** όπως είναι το **GAL AMD 22V10** που χρησιμοποιήθηκε καθώς και την προσομοίωση λογικών κυκλωμάτων που υλοποιήθηκαν.

Προαπαιτούμενα

Τα προαπαιτούμενα των προηγούμενων εργαστηριακών ασκήσεων ταυτίζονται και με την τρέχουσα όμως επιπλέον απαιτούνταν η μελέτη της **γλώσσας PLDShell** και των χαρακτηριστικών στοιχείων του ολοκληρωμένου κυκλώματος **GAL AMD 22V10 (ονομασία – διάταξη - δυνατότητα επαναπρογραμματισμού).**

Διεξαγωγή του Εργαστηρίου

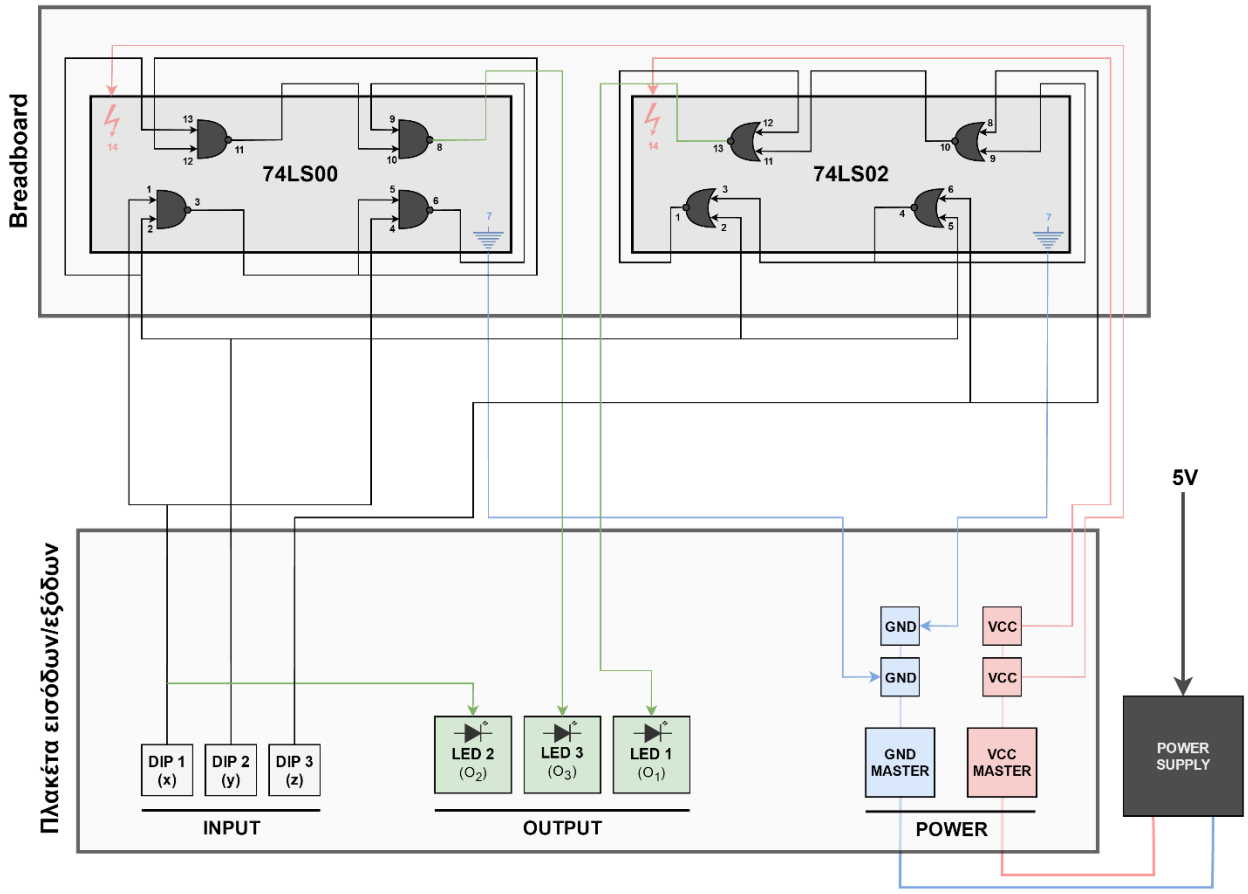
Πρακτικά, αυτή η εργαστηριακή άσκηση αποτέλεσε συνέχεια της προηγούμενης. **Παραθέτουμε περιληπτικά το προηγούμενο μέρος**: η προηγούμενη άσκηση ζητούσε να κατασκευάσουμε ένα κύκλωμα με τρεις εισόδους **x**,**y**,**z**, και τρεις εξόδους $\mathbf{O_1}$, $\mathbf{O_2}$,$\mathbf{O_3}$. Για κ=4 παραθέτουμε τη διαδρομή στον κύβο για τον κώδικα Gray.



X	Y	Z	O_1	O_2	O_3
0	0	0	**1**	0	0
0	0	1	0	0	0
0	1	0	0	0	**1**
0	1	1	**1**	0	**1**
1	0	0	**1**	**1**	**1**
1	0	1	0	**1**	**1**
1	1	0	0	**1**	0
1	1	1	**1**	**1**	0

Η διαφορά των δύο εργαστηριακών ασκήσεων έγκειται στην υλοποίηση του λογικού κυκλώματος λόγω διαφορετικής διαταξης και συνδεσμολογίας.

Στο προηγούμενο εργαστήριο χρησιμοποιήσαμε συνολικά 8 λογικές πύλες και δύο ολοκληρωμένα κυκλώματα TTL (1 x 74LS00 , 1 x 74LS02). Η τελική συνδεσμολογία διαφαίνεται στο παρακάτω σχήμα:

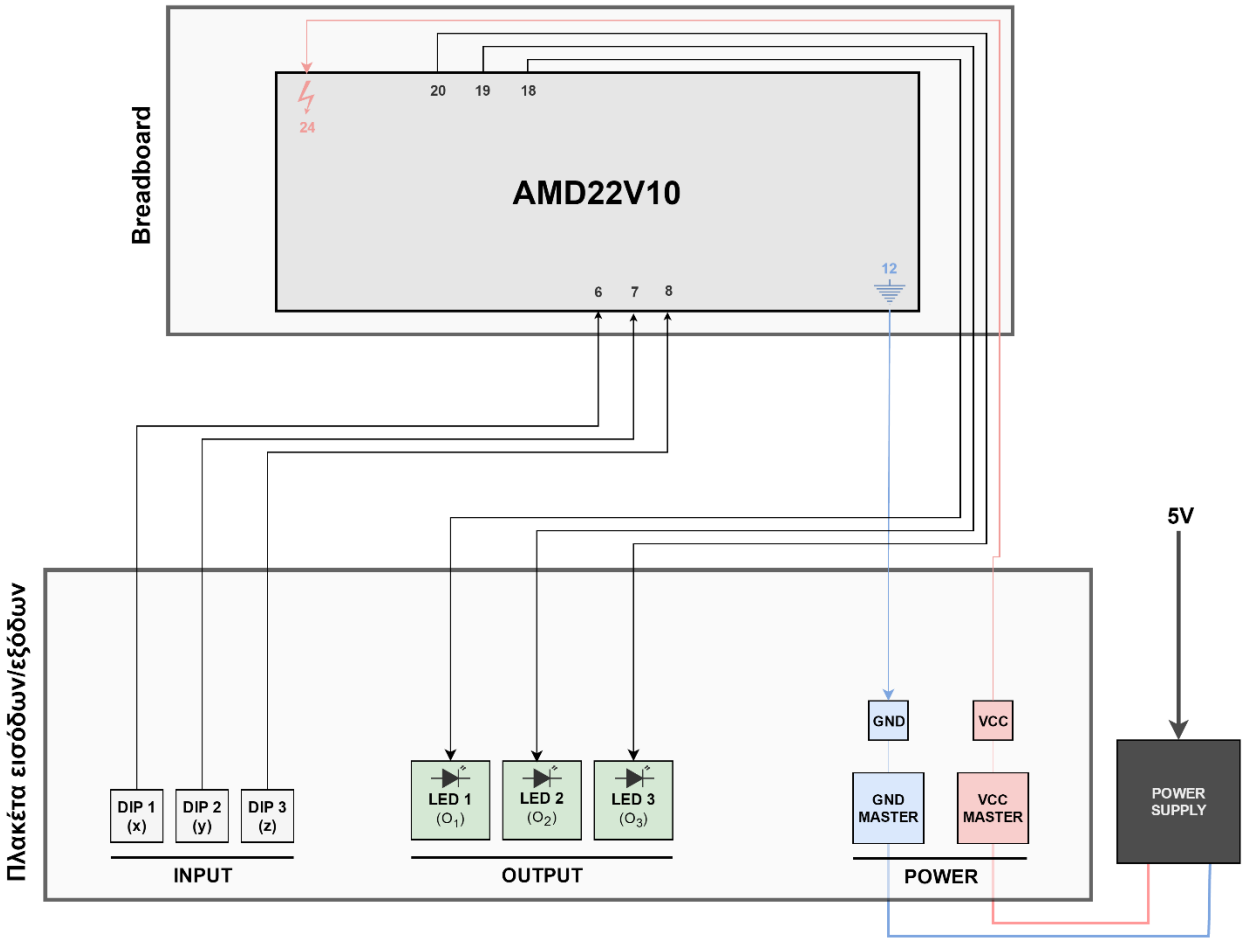


Εν αντιθέσει, αυτή τη φορά χρησιμοποιήσαμε το **GAL AMD 22v10** το οποίο έχει τη δυνατότητα να προγραμματίζεται, **δηλαδή να μεταβάλλει την εσωτερική του διάταξη επιλέγοντας κάθε φορά κατάλληλες πύλες για να παραγάγει το ζητούμενο αποτέλεσμα.**

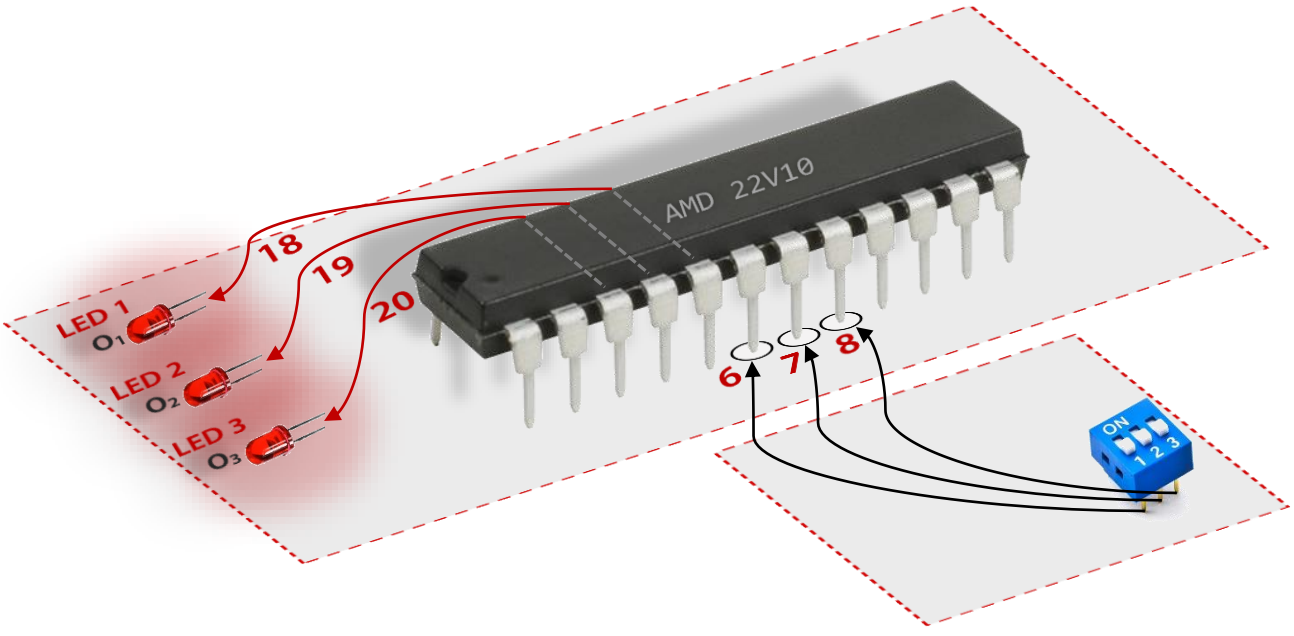
Ο προγραμματισμός του επιτυγχάνεται απο εξωτερικό παράγοντα με τον εξής τρόπο: ο χρήστης (εμείς) εισάγει ένα αρχείο κώδικα υπο τη μορφή **.pds** στο PLDShell. Το πρόγραμμα αυτό μεταφράζει τον κώδικα (compile) και προσομοιώνει το λογικό κύκλωμα με κυματομορφές ορατές στον χρήστη. Στη διαδικασία του compilation, το PLDShell δημιουργεί ενα αρχείο **.jed** το οποίο αναγνωρίζεται απο **ειδική συσκευή** η οποία μεταφέρει την πληροφορία στο AMD 22v10.

Αφού ακολούθησε η διαδικασία αυτή με βοήθεια του καθηγητή, καταφέραμε να υλοποιήσουμε το κυκλωμά μας συνδέοντας απλά και μόνο εισόδους, εξόδους και τροφοδοσία.

Παραθέτουμε παρακάτω τη γενική σχηματική αναπαράσταση του κυκλώματος με τη χρήση του AMD 22v10. Σημειώνεται οτι οι είσοδοι που χρησιμοποιήθηκαν ήταν τα pins **6, 7** και **8** για το **x, y** και **z** αντίστοιχα. Στην έξοδο παρομοίως είχαμε τα pins **18, 19** και **20** για τα $\mathbf{O_1}$, $\mathbf{O_2}$ και $\mathbf{O_3}$ αντίστοιχα.



Παρατηρούμε ότι η συνδεσμολογία με τη χρήση του AMD 22v10 καθίσταται πολύ ευκολότερη καθώς η επιλογή των λογικών πυλών είναι αυτόματη μέσω του ολοκληρωμένου αυτού κυκλώματος.



Κώδικας PLDShell

Ο κώδικας που χρειάστηκε το PLDShell ώστε να πραγματοποιηθεί αυτή η άσκηση είναι ο παρακάτω (παρουσιάζονται σχόλια που εξηγούν τα διάφορα τμήματα του):

```
1   CHIP LAB3 22V10 }
2   ;pin inputs
3   ;pin num var
4   PIN 6 X
5   PIN 7 Y
6   PIN 8 Z
7   ;pin outputs
8   PIN 18 O1
9   PIN 19 O2
10  PIN 20 O3
11  EQUATIONS
12  O1 = /(Y :+: Z)
13  O2 = X
14  O3 = X :+: Y
15  SIMULATION
16  VECTOR A := [X Y Z]
17  FOR I := 0 TO 7 DO
18  BEGIN
19  SETF A := I
20  END
```

Καθορίζουμε αρχικά οτι πρόκειται για προγραμματισμό του ολοκληρωμένου κυκλώματος **22V10** με όνομα σχεδίασης "LAB3".

Στο σημείο αυτό δηλώνουμε τις εισόδους στο AMD 22V10. Στη συγκεκριμένη περίπτωση, αντιστοιχούμε το **X** στο **pin 6**, το **Y** στο **pin 7** και το **Z** στο **pin 8**.

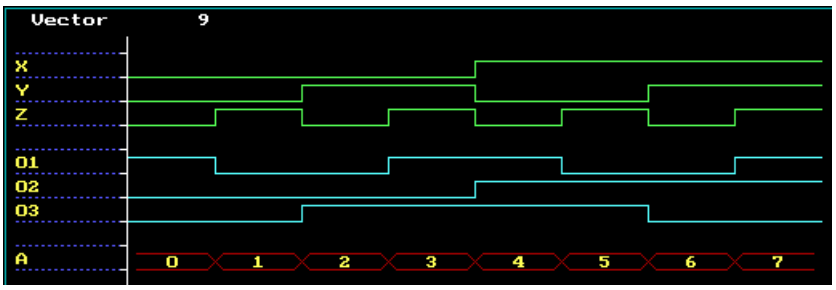
Αντίστοιχα, πληροφορούμε το πρόγραμμα ότι τα **pins 18,19** και **20** πρόκειται για έξοδοι με ονόματα $\mathbf{O_1}$, $\mathbf{O_2}$ και $\mathbf{O_3}$ αντίστοιχα.

Με βοήθεια Άλγεβρας Boole καθορίζουμε τις λογικές συναρτήσεις, των οποίων το αποτέλεσμα εξέρχεται απο τις αντίστοιχες εξόδους $\mathbf{O_1}$, $\mathbf{O_2}$ και $\mathbf{O_3}$.

Η εντολή **vector** ομαδοποιεί τους **3 αριθμούς ενός bit** σε **έναν αριθμό 3-bit.** Για κάθε αριθμό 3-bit απο το **0** εώς το **7 (000-111)** το PLDShell εκτελεί προσωμοίωση του λογικού κυκλώματος η οποία φαίνεται σε εμας ως κυματομορφή και κατόπιν τερματίζεται ο προγραμματισμός.

Κυματομορφή

Το ακόλουθο σχήμα αποτελεί την κυματομορφή του λογικού κυκλώματος που προσομοίωσε το PLDShell μεσω του κώδικα που χρησιμοποιήθηκε παραπάνω. Επιβεβαιώσαμε με αυτόν τον τρόπο την ορθότητα του.



Συμπεράσματα

Στο εργαστήριο αυτό επεκτείναμε τις γνώσεις μας σχετικά με τα ολοκληρωμένα κυκλώματα και μελετήσαμε το AMD 22V10. Κατανοήσαμε τα χαρακτηριστικά του, και κυρίως βασιστήκαμε στην ιδιότητα του να μεταβάλλει δυναμικά τις εξόδους του. Κατασκευάσαμε το ίδιο κύκλωμα με το προηγούμενο εργαστήριο για να συγκρίνουμε την πολυπλοκότητα των δύο τρόπων υλοποίησης τους και καταλήξαμε στο συμπέρασμα ότι το AMD 22V10 βοηθάει κατά πολύ όσον αφορά τη συνδεσμολογία καθώς καταφέραμε να υλοποιήσουμε το κύκλωμά μας με **10 μόνο καλώδια** (3 για την είσοδο, 3 για την έξοδο και 2 για την τροφοδοσία (VCC-GND)) **ενάντι 23** (3 για την είσοδο, 3 για την έξοδο, 15 για χρήση λογικών πυλών και 4 για την τροφοδοσία).