ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ & ΥΛΙΚΟΥ
ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ ΓΙΑ ΤΟ ΜΑΘΗΜΑ:
ΠΡΟΧΩΡΗΜΕΝΗ ΛΟΓΙΚΗ ΣΧΕΔΙΑΣΗ – ΗΡΥ203
http://www.mhl.tuc.gr
ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2019-2020

Αναφορά για τις Εργαστηριακές Ασκήσεις 1-5

Αριθμός Ομάδας: …**LAB20343360**..…….....
Ονοματεπώνυμο 1: …**ΠΕΤΡΟΥ ΔΗΜΗΤΡΙΟΣ.....2018030070**..…
Ονοματεπώνυμο 2: …**ΦΡΑΓΓΙΑΣ ΓΕΩΡΓΙΟΣ.....2018030086**.....

Σκοπός των ασκήσεων

Οι εργαστηριακές ασκήσεις, μελετώντας η κάθε μία ένα ξεχωριστό κομμάτι της ψηφιακής λογικής, στόχευαν στην εξοικειώση με την γλώσσα VHDL και τα εργάλεια ανάπτυξης της καθώς και στην σχεδίαση και ανάπτυξη λογικών κυκλωμάτων με αυτήν. Η πολυπλοκότητα των ασκήσεων κλιμακώθηκε, ξεκινώντας από την υλοποίηση απλών συνδυαστικών κυκλωμάτων μέχρι και την σχεδίαση μηχανών πεπερασμένων καταστάσεων. Στόχο επίσης αποτέλεσε και η σωστή κατανόηση και κατ'επέκταση και χρήση του παραλληλισμού, όπως αυτός υπάρχει στην VHDL προκειμένου να παρομοιάσει τις φυσικές σχεδιάσεις υλικού. Για όλες τις ασκήσεις ήταν γνώμονας η εφαρμογή ιεραρχικής σχεδίασης και η αναγνώριση των πλεονεκτημάτων που προσφέρει.

Περιγραφή τεχνολογίας που χρησιμοποιήθηκε

Στις εργαστηριακές ασκήσεις της Προχωρημένης Λογικής Σχεδίασης χρησιμοποιήθηκε το εργαλείο της Xilinx Vivado. Το Vivado δέχεται στην προκειμένη περίπτωση κώδικα στην γλώσσα περιγραφής υλικού και σχεδίασης VHDL για την σύνθεση κυκλωμάτων και την προσομοίωση των αντίστοιχων testbench. Τα κυκλώματα που παράγονται αποτελούνται από αντίστοιχες λογικές πύλες TTL. Αυτό έχει ως απώτερο σκοπό την δημιουργία ενός δυαδικού αρχείου .bitfile, το οποίο ρυθμίζει ανάλογα μια αναδιατασσόμενη συσκευή FPGA. Η συσκευή FPGA είναι το αποτέλεσμα της σχεδιαστικής ροής, που ακολουθείται από το εργαλείο της Xilinx, σε μορφή hardware.

Περιγραφή της μεθοδολογίας επίλυσης των ασκήσεων

Με κύριο άξονα την ιεραρχική σχεδίαση, τις αρχές της αποσύνθεσης και την σχεδιαστική ροή του waterfall model (δεν εφαρμόστηκε στην ολότητα του) η μεθοδολογία επίλυσης των ασκήσεων κινήθηκε όσο τον δυνατόν πιο κοντά στα πρότυπα που παρουσιάστηκαν στο μάθημα.

Στην $1^{η}$ και $2^{η}$ εργαστηριακή άσκηση σκοπός ήταν η δημιουργία συνδυαστικών κυκλωμάτων. Η σχεδιαστική διαδικασία ξεκινούσε από την μελέτη λογικών εξισώσεων και block diagrams, ώστε να είναι κατανοητή η λειτουργικότητα της κάθε μονάδας (bottom-up) αλλά και η διασύνδεση των μονάδων στο ολοκληρωμένο κύκλωμα (top-down). Η μελέτη αυτή ήταν εύκολη, διότι περιλαμβανόταν στις εκφωνήσεις των ασκήσεων.

Έπειτα, η υλοποίηση ακολουθούσε το μοντέλο bottom-up δηλαδή ιεραρχική σύνδεση. Αρχικά με υλοποίηση μονάδων πρώτου επιπέδου, διενεργώντας testbench για το καθένα ώστε τα σφάλματα να εντοπίζονται στην πηγή τους, και συνθέτοντας με χρήση αυτών μονάδες μεγαλύτερου επιπέδου(π.χ. full adder με χρήση half adder).

Στην επίλυση αυτών των ασκήσεων βοήθησαν και οι μέθοδοι δειγματοληψίας των testbench. Για παράδειγμα, στο κύκλωμα 1 της 1ης άσκησης το οποίο είχε 8 εισόδους ένα πλήρες testbench θα χρειαζόταν θεωρητικά 2^8=256 περιπτώσεις. Όμως, παρατηρήθηκε ότι κάθε μία από τις συναρτήσεις που ζητούνταν εξαρτιόταν από μόνο μία είσοδο $C_{0...5}$. Οπότε, χρειάστηκε μόνο να δοκιμαστούν οι περιπτώσεις, όπου όλες οι είσοδοι C ήταν 0 ή όλες 1. Έτσι απαραίτητες ήταν μόνο 8 καταστάσεις.

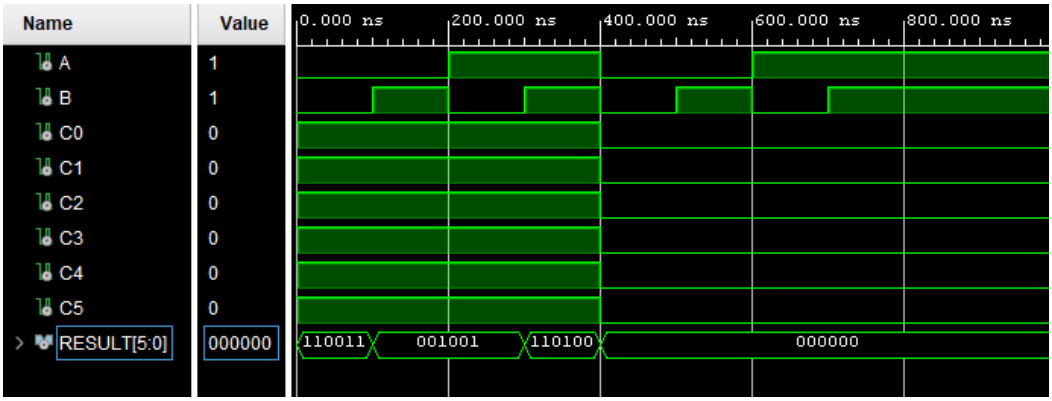
Στην 3η, 4η και 5η άσκηση με στόχο την υλοποίηση μηχανών πεπερασμένων καταστάσεων και ενός μετρητή (ακολουθιακών σύγχρονων κυκλωμάτων) ακολουθήθηκαν τα σχετικά βήματα σχεδίασης. Αφού καθορίστηκαν οι απαιτήσεις ως προς τις καταστάσεις, την διαδοχή τους, τις εισόδους, τις εξόδους και τον τύπο των FSM, η σχεδίαση συνέχισε στο πιο πρακτικό της κομμάτι. Αναγνωρίστηκαν οι ανάγκες σε ports και έγινε η αποσύνθεση του κυκλώματος σε επιμέρους modules. Δημιουργήθηκαν τα entities και το κάθε ένα απέκτησε το κατάλληλο architecture. Προκείμενου να ανιχνευθούν σφάλματα και ανεπιθύμητες συμπεριφορές νωρίς, πραγματοποιήθηκε simulation για κάθε module. Αφού επιβεβαιωνόταν ότι η απόκριση κάθε module ήταν η αναμενόμενη δημιουργήθηκαν τα top-level entities, όπου ήταν απαραίτητο, τα οποία συνέθεταν και υλοποιούσαν την τελική σχεδίαση. Σε περίπτωση που το wrapper module διαπιστωνόταν ότι δεν λειτουργούσε εντός των προδιαγραφών, εφαρμοζόταν το κατάλληλο debugging ή και αρκετές φορές ριζική επανασχεδίαση και υλοποιήση ορισμένων κρίσιμων υπομονάδων του.

Ειδικότερα, ο συγχρονισμός των FSM με ένα σήμα ρολογιού επιτεύχθηκε με όλη τη δομη case, που υλοποιεί τις καταστάσεις και τις μεταβάσεις τους, να περιμένει την θετική ακμή του ρολογιού. Εντός του fragment κώδικα που αντιστοιχεί στην κάθε κατάσταση, με εντολές συνθήκης που ελέγχουν τις παρεχόμενες εισόδους, υλοποιήθηκαν κατάλληλα οι μεταβάσεις από την εκάστοτε κατάσταση στις υπόλοιπες με την χρήση ειδικά ορισμένων σημάτων.

Ανά τις ασκήσεις, όπως αναφέρθηκε, εφαρμόστηκε σε ιδιαιτέρως πολλά σημεία προσομοίωση. Αυτό, δεν είναι κάτι περίπλοκο, παρά η απόδοση τιμών στις εισόδους του κάθε module και η παρατήρηση της συμπεριφοράς του στις θύρες εξόδου. Σε ένα testbench, ορίζονταν τοπικά σήματα τα οποία οδηγούνταν στις θύρες της μονάδας που βρισκόταν υπό testing. Αφού είχε γίνει το κατάλληλο port map (η σύνδεση των τοπικών σημάτων στις θύρες) διαδοχικά γινόταν, μέσω ενός process, ανάθεση τιμών στις εισόδους για συγκεκριμένα χρονικά παράθυρα με βάση τις εκάστοτε απαιτήσεις. Σημειώνεται, ότι στις ασκήσεις που αντικείμενο τους ήταν η υλοποίηση ακολουθιακών κυκλωμάτων υπήρχε εντός του κάθε testbench κατάλληλο process για την παραγωγή σήματος ρολογιού.

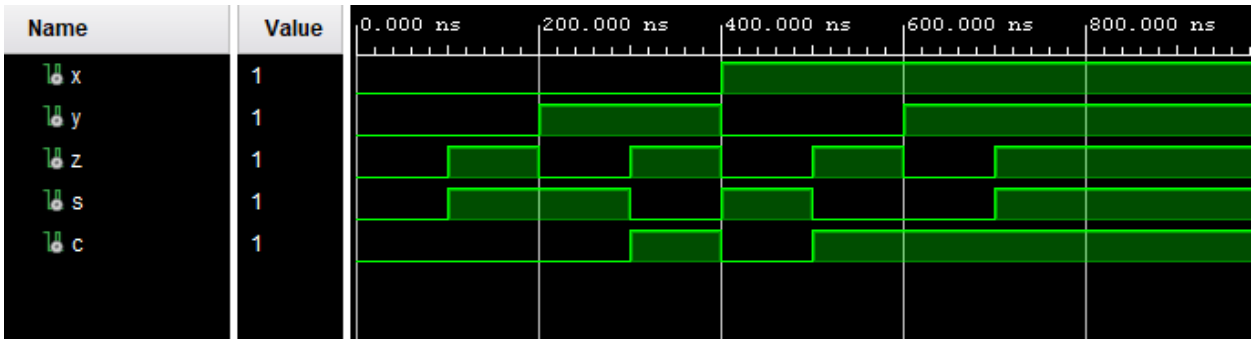
Αναφορά αποτελεσμάτων

Μετά από την ολοκληρωμένη σχεδίαση των modules και επιτυχημένο simulation που επιβεβαίωνε την λειτουργία τους για την κάθε άσκηση, γινόταν και simulation για την τελική σχεδίαση. Παρακάτω παρατίθενται τα αποτελέσματα που παράχθηκαν μέσω των testbench.



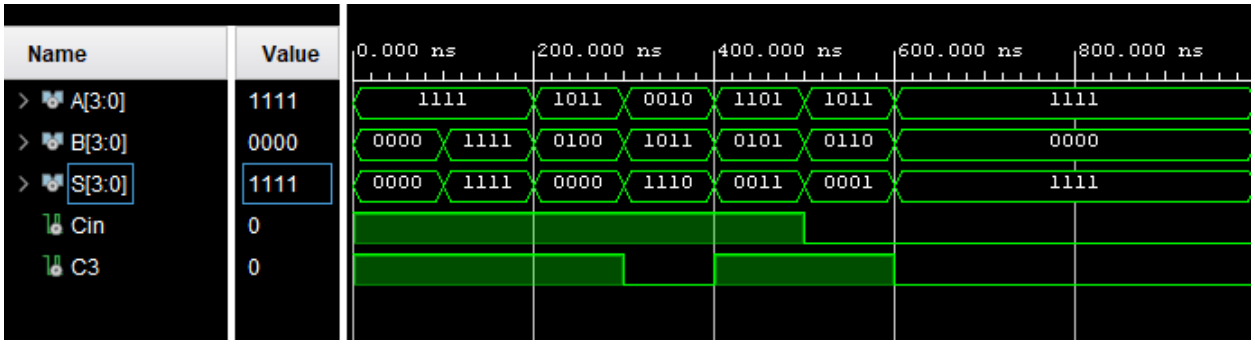
1. Ασκ1, Κυματομορφές εξισώσεων

Στην προσομοίωση των εξισώσεων της πρώτης άσκησης, βλέπουμε τα αποτελέσματά τους μέσα στο bus RESULT. Όσο τα σήματα ελέγχου C0-C5 είναι ανενεργά, δεν παράγεται καμία έξοδος ανεξαρτήτως των εισόδων A και B.



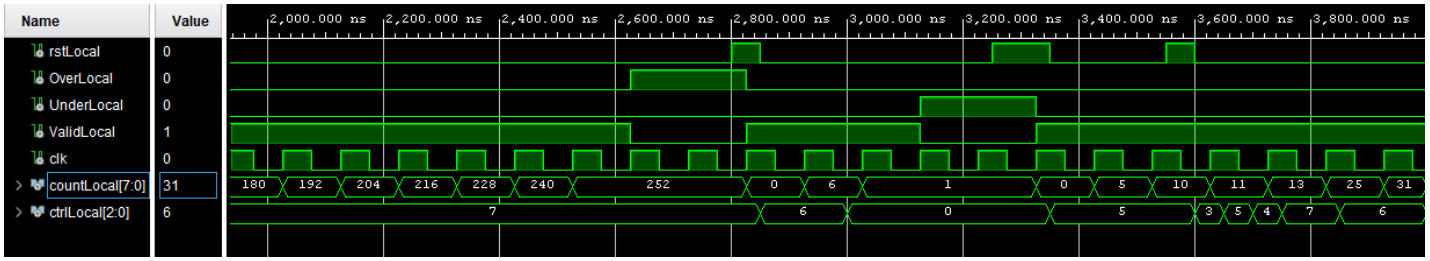
2. Ασκ1, Κυματομορφές πλήρους αθροιστή

Δοκιμάζοντας τον πλήρη αθροιστή, τα αποτελέσματα είναι τα αναμενόμενα. Παρατηρώντας τις ακραίες περιπτώσεις βλέπουμε ότι για όλες τις εισόδους «1» (αθροιστέους & carry-in) το άθροισμα είναι «1» και το carry-out «1» επαληθεύοντας τις αρχές της δυαδικής άθροισης και της άλγεβρας Boole. Όλες οι υπόλοιπες περιπτώσεις φέρονται ανάλογα



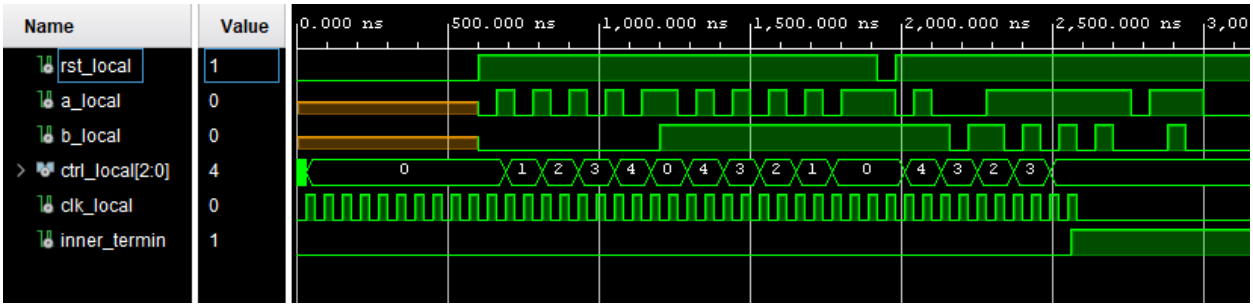
3. Ασκ2, Αθροιστής με πρόβλεψη κρατουμένου

Η διαδικασία προσομοίωσης για τον αθροιστή με πρόβλεψη κρατουμένου δείχνει ότι η σχεδίαση συναντά τις αρχικές προδιαγραφές. Η άθροιση του 15 με το 0 και carry-in 1 βλέπουμε ότι αφήνει κρατούμενο εξόδου 1 και άθροισμα 0. Αντίστοιχα, σε περιπτώσεις που δεν αναμένεται η πρόσθεση να προκαλέσει overflow παρατηρείται το σωστό άθροισμα στην έξοδο για τους δύο προσθετέους.



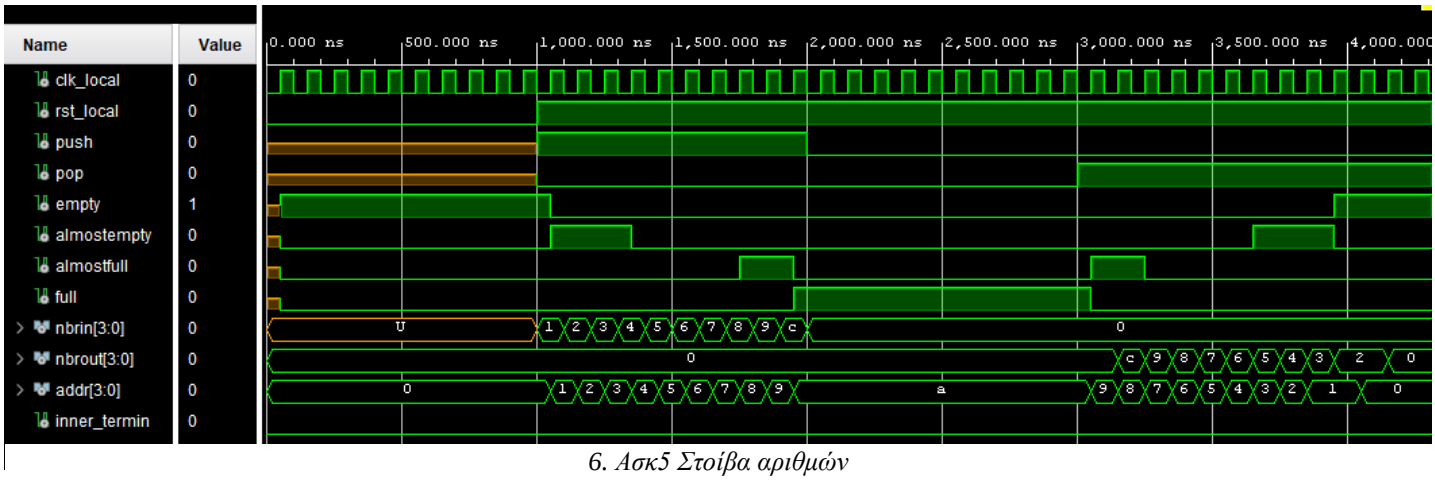
4. Ασκ3, Περίεργος Μετρητής

Έχοντας προχωρήσει την μέτρηση σε αρκετά μεγάλους αριθμούς, βλέπουμε ότι προσπαθώντας να κάνουμε βήμα που ξεπερνά το όριο του 254 (υπερχείλιση) ο μετρητής σταματάει και το σήμα Valid γίνεται «0». Αντίστοιχα επιβεβαιώνεται και η λειτουργία του ελέγχου (πρόβλεψης) για Underflow. Τέλος φαίνεται ότι ο μετρητής έχει την δυνατότητα να δέχεται διαφορετικό βήμα μέτρησης σε κάθε κύκλο ρολογιού όπως είχε προδιαγραφεί.



5. Ασκ4 Μηχανή πεπερασμένων καταστάσεων

Η μηχανή πεπερασμένων καταστάσεων που υλοποιήθηκε ανάλογα με τις εισόδους A και B πραγματοποιεί τις κατάλληλες μεταβάσεις μεταξύ των καθορισμένων states. Παρατηρούμε και την επιτυχή επέμβαση του σήματος RST στο κύκλωμα καθώς και την παύση της λειτουργίας όλης της μηχανής μετά τον τερματισμό του ρολογιού. Η δεξιόστροφη και αριστερόστροφη διαδοχή (με βάση το state diagram) των καταστάσεων φέρεται να λειτουργεί πρότυπα αφού η μηχανή επανεκκινείται στην «0» αφού φτάσει στην «4» και αντιστρόφως.



6. Ασκ5 Στοίβα αριθμών

Στέλνοντας για 10 κύκλους RST η στοίβα παραμένει αδρανής. Η διαδικασία του push ξεκινάει και παρατηρούνται οι ενδείξεις για την χωρητικότητα της στοίβας να μεταβάλλονται καταλλήλως ξεκινώντας από την Empty να είναι «1», ύστερα να την διαδέχεται η Almost Empty κ.ο.κ.. Αργότερα, μετά από 10 κύκλους αδράνειας προκειμένου να κατασταλλάξει η στοίβα, ξεκινάει το Pop και οι αριθμοί αφαιρούνται από την LIFO με την σειρά που πρέπει. Ωστόσο παρουσιάστηκε πρόβλημα και η στοίβα καθυστέρει έναν κύκλο, χάνοντας το Pop της μοναδιαίας διεύθυνσης.

Συμπεράσματα

1η Άσκηση: Η πρώτη επαφή με την γλώσσα περιγραφής υλικού έδειξε ότι διευκολύνει αρκετά τη σχεδιαστική διαδικασία. Για .Η απόδοση των εργαλείων προσομοίωσης και ανάπτυξης για μικρά μη-σύνθετα κυκλώματα είναι ιδιαίτερα γρήγορη.

2η Άσκηση: Η υλοποιήση του αθροιστή με πρόβλεψη κρατουμένου είναι κλιμακωτή, τα module των χαμηλότερων επιπέδων αποσφαλματώνονται και ύστερα συνεχίζει στα top-level entities

3η Άσκηση: Ήταν η πρώτη άσκηση που εισήγαγε τις έννοιες της ακολουθιακής λογικής στην VHDL. Η γλώσσα έχει όλες τις απαραίτητες δυνατότητες προκειμένου να μπορέσει να υλοποιηθεί ένας μετρητής. Οι εντολές συνθήκης παίζουν το βασικό ρόλο για τον αντικατοπτρισμό της διαδοχής των καταστάσεων στον κώδικα.

4η Άσκηση: Η συγγραφή κώδικα που να περιγράφει μια FSM βοήθησε σημαντικά στην κατανόηση της ιδιότητας της VHDL να είναι concurrent. Απαιτούνται ειδικά σήματα προκειμένου να περιγραφούν οι καταστάσεις.

5η Άσκηση: Μία FPGA μπορεί να συνδεθεί με εξωτερικά περιφερειακά όπως μια μνήμη ή κάποιες εξωτερικές θύρες επικοινωνίας. Η κατάλληλη σχεδίαση μπορεί να χειριστεί εύκολα θέσεις-διευθύνσεις μνήμης ώστε να υλοποιήσει μια στοίβα κάποιου είδους.

Στην σχεδίαση και υλοποίηση κυκλωμάτων με γλώσσα περιγραφής υλικού οι θεωρητικές σχεδιαστικές ροές, όπως το waterfall model και η ιεραρχική σχεδίαση φαίνονται χρήσιμες και επιβεβαιώνεται η πρακτικότητα τους. Η διαδικασία σχεδίασης ενός κυκλώματος με επιμέρους προσομοίωση σε κάθε στάδιο της, αποδεικνύεται ότι μπορεί να εκμηδενίσει και να αποτρέψει τα σφάλματα. Προσομοίωση ωστόσο δεν σημαίνει ότι ανατίθενται όλες οι δυνατές τιμές στις εισόδους, κάτι τέτοιο θα ήταν χαοτικό σε πολύπλοκα κυκλώματα. Επιλέγονται διακριτές περιπτώσεις στις οποίες η συμπεριφορά της σχεδιάσης κρίνεται ότι θα είναι καθοριστική. Είναι αξιοσημείωτο το πόσο υπερτερεί η τεχνολογία των FPGA όσο αφορά στην ευκολία της υλοποίησης σε επίπεδο τόσο σχεδίασης όσο και hardware. Μόνο λίγες γραμμές περιγραφικού κώδικα αρκούν για να μας υλοποιήσουν μία FSM ή έναν μετρητή χωρίς πολλαπλά IC και άπειρα σύρματα. Διαπιστώνουμε πως η ευελιξία που προσφέρει η VHDL και κατ επέκταση οι FPGAs επισκιάζουν σε μεγάλο ποσοστό τους κλασσικούς τρόπους υλοποιήσης κυκλωμάτων, χωρίς ωστόσο να εκλείπουν σφάλματα και ανεπιθύμητες καταστάσεις.