# 1η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ

# “ΣΧΕΔΙΑΣΜΟΣ ΣΥΣΤΗΜΑΤΩΝ VLSI”

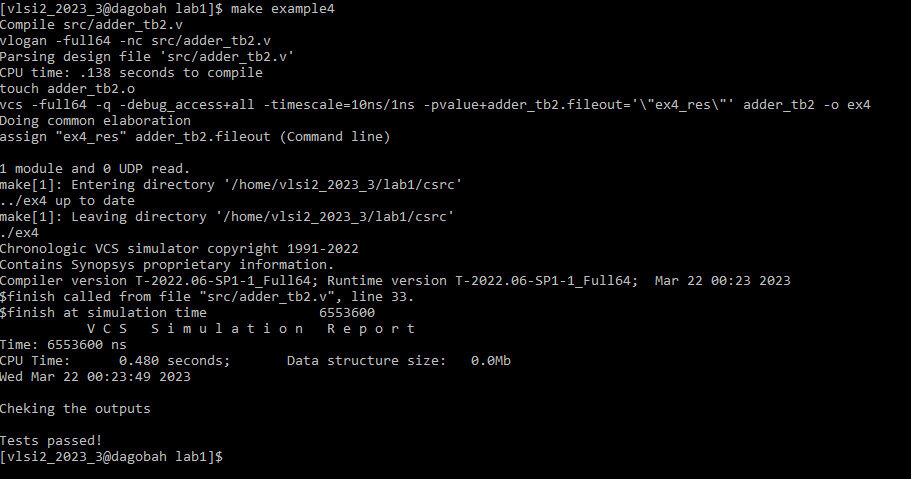
Ζητούμενα:

1. Για το πρώτο ζητούμενο τροποποιήσαμε τον κώδικα του παράδειγμα 4α, στον οποίο δημιουργείται ένα testbench για τον έλεγχο της λειτουργίας του 8-bit rca για τυχαίες εισόδους με τη χρήση αρχείου εξόδου το οποίο στη συνέχεια εξετάζεται για σφάλματα από διαφορετικό πρόγραμμα. Στην δική μας έκδοση του κώδικα ο οποίος παρατίθεται στο αρχείο adder\_tb2.v χρησιμοποιούμε 2 for loops (εμφωλευμένες η μία στην άλλη) για να αναθέσουμε κάθε πιθανή τιμή στις μεταβλητές a, b (8-bit είσοδοι του rca, άρα τιμές από 0 έως 255) ανά 10 χρονικές στιγμές και να γράψουν το αποτέλεσμα σε ένα αρχείο εξόδου, το οποίο θα ελεγχθεί στη συνέχεια όπως και στο προαναφερθέν παράδειγμα. Ο έλεγχος του αρχείου γίνεται μέσω του makefile μέσω των παρακάτω εντολών οι οποίες διαβάζουν γραμμή-γραμμή το αρχείο και εμφανίζουν σφάλμα σε περίπτωση που έχει παραχθεί κάποιο λάθος αποτέλεσμα από την εξομοίωση του κυκλώματος.

@echo -e "\nCheking the outputs\n"

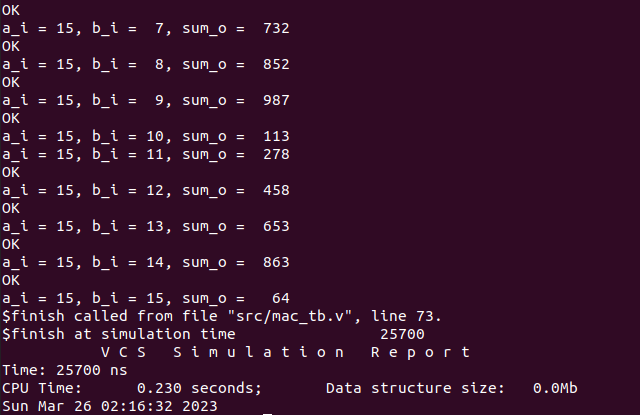
@awk '{if (($$1 + $$2)!=$$3){print "error in: "$$0;er=1 }}END{if (er!=1){print "Tests passed!"}}' ex4\_res

To αρχείο που παράγεται παρατίθεται και αυτό στο zip της αναφοράς με όνομα ex4\_res.

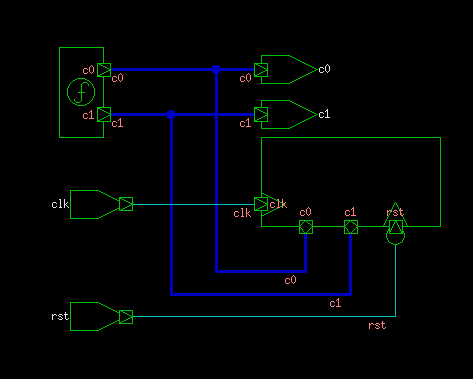
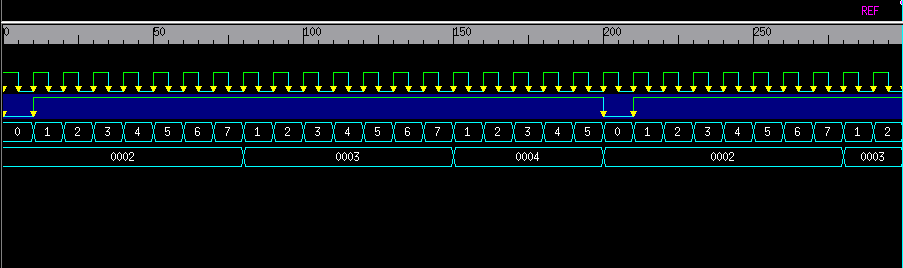


Εικόνα : Εκτέλεση script και εμφάνιση πιθανών σφαλμάτων

1. Για το δεύτερο ζητούμενο ακολουθήσαμε παρόμοια στρατηγική με το πρώτο ζητούμενο και χρησιμοποιώντας 2 for loops εμφωλευμένες η μία στην άλλη, δώσαμε κάθε πιθανή τιμή στις μεταβλητές a\_i και b\_i, οι οποίες σε αυτό το παράδειγμα ωστόσο, εκτυπώνονται μαζί με το αποτέλεσμα που παράγεται από το κύκλωμα στο terminal καθώς και η λέξη “OK” σε περίπτωση που η πράξη είναι σωστή. Ο συγκεκριμένος τρόπος ελέγχου του κυκλώματος δεν είναι απολύτως σωστός, καθώς για την υλοποίηση του ίδιου του κυκλώματος χρησιμοποιούμε τις ίδιες συναρτήσεις που χρησιμοποιούμε και για τον έλεγχο συνεπώς δεν μπορούμε να ελέγξουμε στην πραγματικότητα αν υπάρχει κάποιο σφάλμα στο κύκλωμα καθώς οι συναρτήσεις θα δίνουν το ίδιο αποτέλεσμα, πλην της περίπτωσης overflow που όπως θα δούμε και στο screenshot από την εκτέλεση παρακάτω δεν εκτυπώνεται η λέξη “OK”. O κώδικας για το testbench του mac παρατίθεται στο αρχείο mac\_tb.v

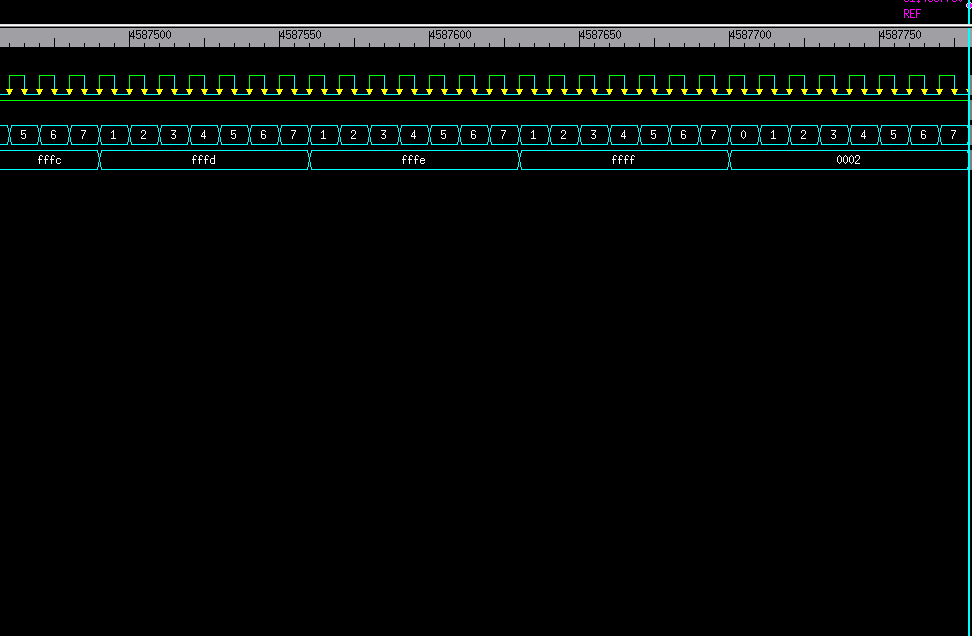


Εικόνα : Εκτέλεση script και εμφάνιση αποτελεσμάτων στο terminal

1. Για το τρίτο ζητούμενο χρειάστηκε να περιγράψουμε ολόκληρο το κύκλωμα σε Verilog (αρχείο double\_counter.v). Αρχικά κατασκευάσαμε ένα module με εισόδους το ρολόι (clk) και το σήμα επαναφοράς αρνητικής λογικής (rst) και εξόδους τα σήματα c0 και c1 (των 3 και 16 bit αντίστοιχα) που είναι και οι τιμές των καταχωρητών του counter. Αφού αρχικοποιήσουμε τα σήματα στις τιμές 0 και 2 αντίστοιχα στο initial block συνεχίζουμε με ένα always block το οποίο έχει στην λίστα ευαισθησίας τα σήματα (clk [θετική ακμή] και rst [αρνητική ακμή]), μέσα στο οποίο περιγράφεται η λειτουργία του κυκλώματος με μια σειρά από if statements. Πρώτα απ’ όλα σε περίπτωση που το σήμα rst έχει την τιμή 0 γίνεται ασύγχρονη επαναφορά του κυκλώματος στις τιμές 0 και 2 στους καταχωρητές c0 και c1 αντίστοιχα σε διαφορετική περίπτωση εάν ο c0 έχει φτάσει την τιμή 7 (άνω όριο του 1ου μετρητή) δίνεται c0 η τιμή 1 ώστε να ξαναξεκινήσει το μέτρημα και ελέγχεται αν ο καταχωρητής c1 έχει φτάσει στο άνω όριό του (65535), όπου και σε αυτή την περίπτωση θέτει τους 2 καταχωρητές στις τιμές 0 και 2 αντίστοιχα με το σήμα rst. Σε περίπτωση που ο c1 δεν έχει φτάσει το άνω όριό του, τότε αυξάνεται κατά 1 και σε περίπτωση που ο c0 δεν έχει φτάσει το άνω όριό του αυξάνεται και αυτός κατά 1. Για την εξομοίωση του κυκλώματος χρησιμοποιήσαμε την δυνατότητα του εργαλείου να δημιουργήσουμε ένα αρχείο tcl στο οποίο δίνουμε τιμές στα σήματα και τρέχουμε το simulation για ορισμένο χρονικό διάστημα και βλέπουμε στο γραφικό περιβάλλον τις αντίστοιχες κυματομορφές (αρχείο double\_counter.tcl)  
   

Εικόνα : Έλεγχος Κυματομορφής για λειτουργία του Counter και την ορθή λειτουργία του σήματος rst

Εικόνα : Παραχθέν Σχηματικό



Εικόνα : Έλεγχος Κυματομορφής για λειτουργία του Counter όταν φτάσει τα άνω όρια των καταχωρητών