ΆΣΚΗΣΗ 5

ΑΝΔΡΟΜΙΔΑΣ ΝΙΚΟΛΑΟΣ 1084641

ΤΣΙΛΙΓΙΑΝΝΗΣ ΚΩΝΣΤΑΝΤΙΝΟΣ 1084642

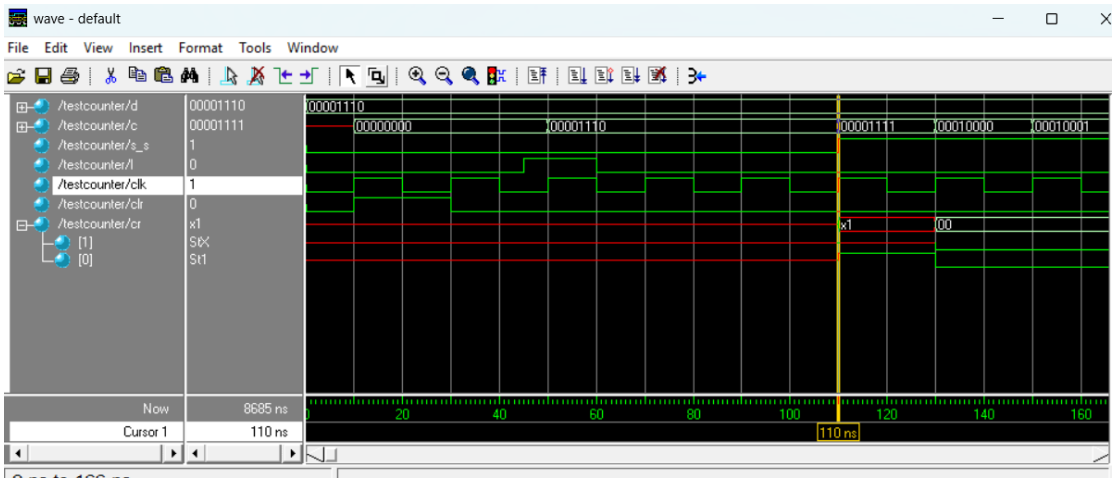
1) Η αλλαγή που κάνουμε στον 7-bit μετρητή είναι να αλλάξουμε τις :
   input [6:0] data;
   output [6:0] count;
   reg [6:0] count;
   ΣΕ
   input [3:0] data;
   output [3:0] count;
   reg [3:0] count;

   Έτσι μετατρέπουμε τον μετρητή μας σε 4bit

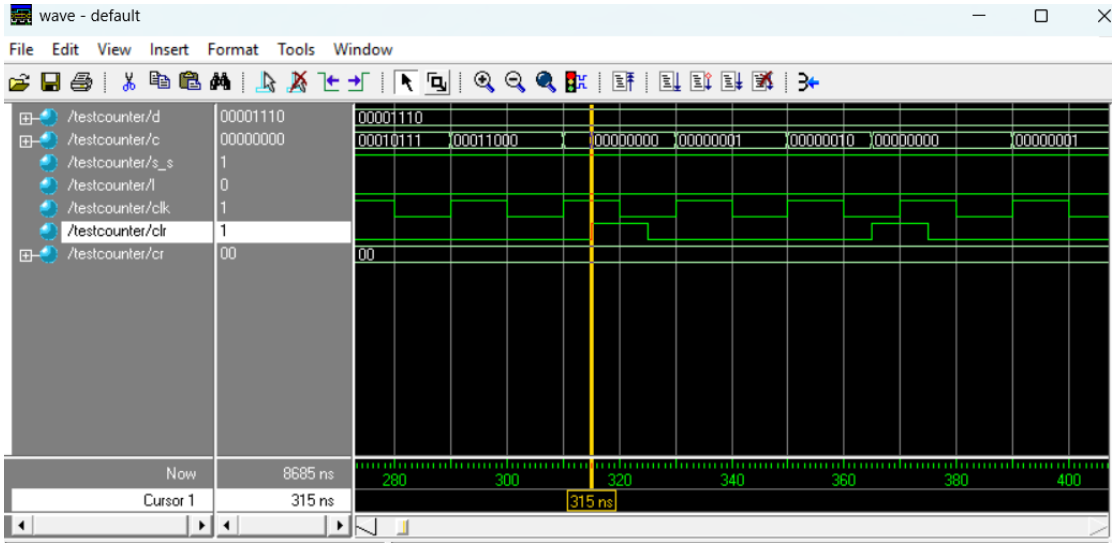
2) Ο κώδικας αυτού του ερωτήματος υλοποιεί έναν 8-bit δυαδικό μετρητή αποτελούμενο από δύο συνδεδεμένους 4-bit μετρητές. Κάθε 4-bit μετρητής έχει δυνατότητα ασύγχρονου μηδενισμού, φόρτωσης εξωτερικής τιμής και αύξησης της μέτρησης μέσω σήματος ενεργοποίησης. Ο πρώτος μετρητής αυξάνεται με βάση το εξωτερικό σήμα start/stop και, όταν φτάσει την τιμή 14, ενεργοποιεί τον δεύτερο μετρητή μέσω του σήματος carry, ώστε να πραγματοποιηθεί η αλυσιδωτή μέτρηση. Η σχεδίαση επιτρέπει την καταγραφή και παρακολούθηση της συνολικής τιμής 8-bit, καθώς και των σημάτων υπερχείλισης από κάθε στάδιο. Στο δεύτερο module (counter8_bit_2_4bit) γίνεται instantiation δύο αντιγράφων του βασικού μετρητή counter, με κοινό ρολόι και σήματα ελέγχου, ενώ το carry του πρώτου μετρητή χρησιμοποιείται ως start/stop του δεύτερου, υλοποιώντας έτσι τη λογική της αλυσιδωτής καταμέτρησης.

3)

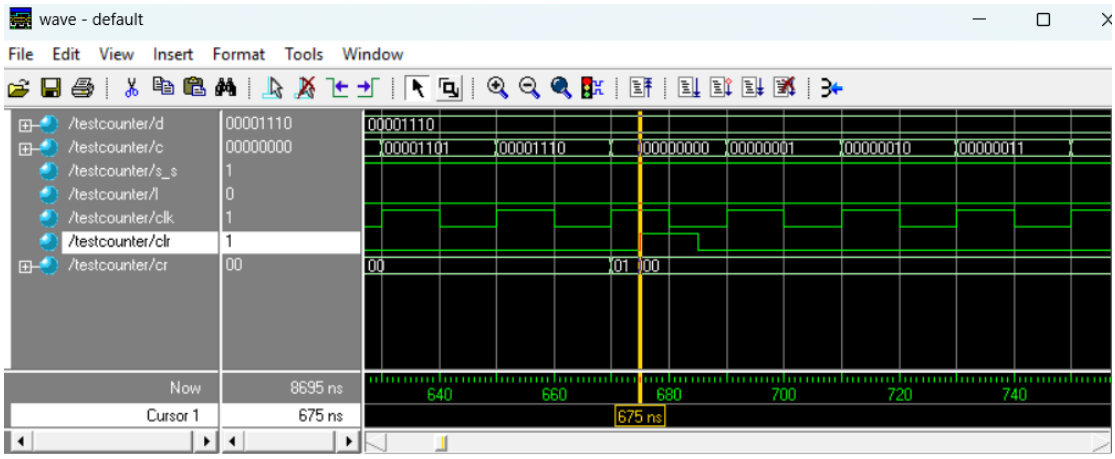
4) Ακολουθούν κάποια στιγμιότυπα επαλήθευσης της ορθής λειτουργίας



Εδώ φαίνεται αρχικά στα 45 ns ότι ενεργοποιείται το σήμα load και φορτώνεται η τιμή που έχουμε ορίσει και έπειτα στα 110 ns φαίνεται ότι ξεκινά η μέτρηση με την ενεργοποίηση του σήματος start_stop. Στην επόμενη θετική ακμή ενεργοποιείται και ο 2ος μετρητής καθώς το κρατούμενο cr[0] είναι 1 και ξεκινά η μέτρηση στον 2ο 4-bit μετρητή.



Εδώ βλέπουμε την λειτουργία του ασύγχρονου καθαρισμού, δηλαδή στα 315 ns με 325 και με θετική ακμή ρολογιού(330 ns) και στα 365 ns και με θετική ακμή ρολογιού (370 ns) γίνεται κανονικά ο καθαρισμός και στις 2 περιπτώσεις αφού πρόκειται για ασύγχρονο καθαρισμό και όποτε ενεργοποιείται ο καθαρισμός τα μηδενίζει.

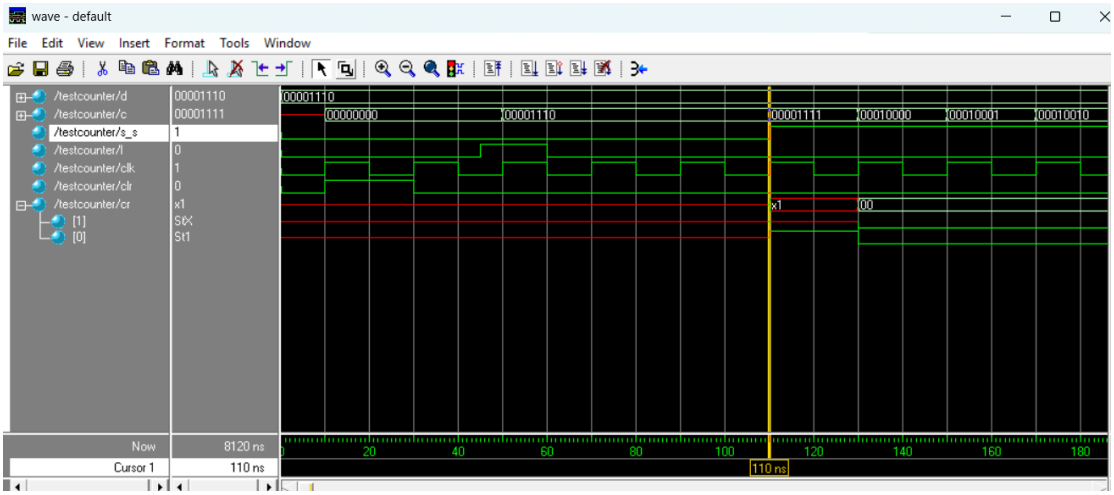


Εδώ βλέπουμε τη σωστή λειτουργία του καθαρισμού που μηδενίζει το κρατούμενο και αποτρέπει έτσι τη λανθασμένη λειτουργία της μέτρησης, δηλαδή έπειτα από τον καθαρισμό να πήγαινε πχ 00000000 σε 00010001 κάτι που θα ήταν λάθος.

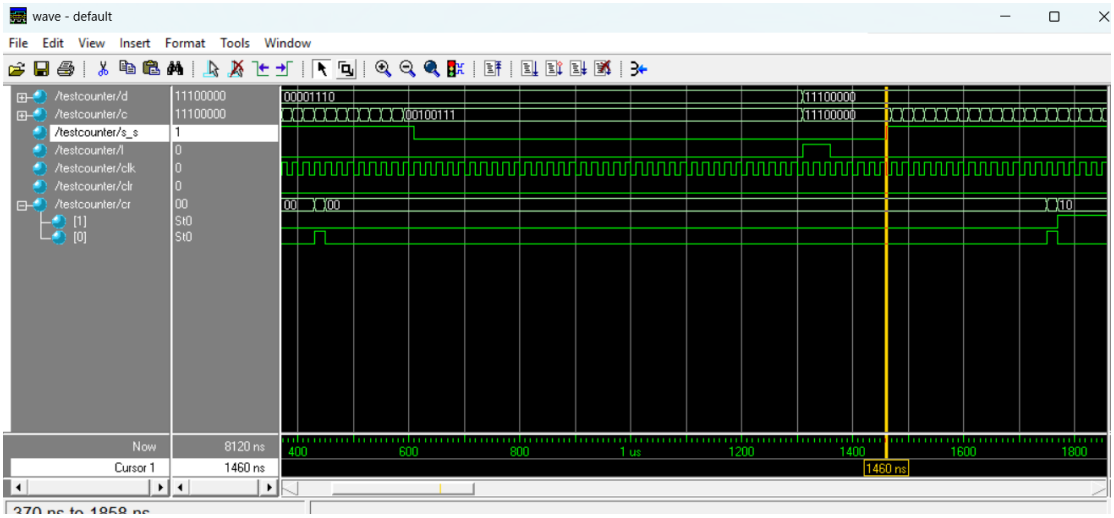
5) α) Ο κώδικας αυτού του ερωτήματος υλοποιεί έναν 8-bit δυαδικό μετρητή αποτελούμενο από δύο συνδεδεμένους 4-bit μετρητές. Κάθε 4-bit μετρητής έχει δυνατότητα σύγχρονου μηδενισμού, φόρτωσης εξωτερικής τιμής και αύξησης της μέτρησης μέσω σήματος ενεργοποίησης. Ο πρώτος μετρητής αυξάνεται με βάση το εξωτερικό σήμα start/stop και, όταν φτάσει την τιμή 14, ενεργοποιεί τον δεύτερο μετρητή μέσω του σήματος carry, ώστε να πραγματοποιηθεί η αλυσιδωτή μέτρηση. Η σχεδίαση επιτρέπει την καταγραφή και παρακολούθηση της συνολικής τιμής 8-bit, καθώς και των σημάτων υπερχείλισης από κάθε στάδιο. Στο δεύτερο module (counter8_bit_2_4bit) γίνεται instantiation δύο αντιγράφων του βασικού μετρητή counter, με κοινό ρολόι και σήματα ελέγχου, ενώ το carry του πρώτου μετρητή χρησιμοποιείται ως start/stop του δεύτερου, υλοποιώντας έτσι τη λογική της αλυσιδωτής καταμέτρησης.

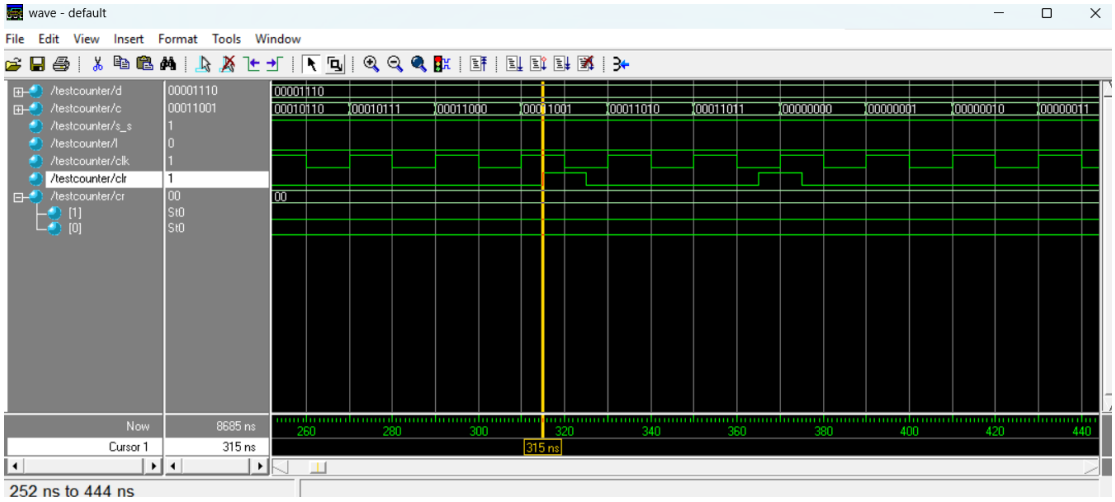
β) Το testbench για τη μονάδα counter8_bit_2_4bit χρησιμοποιείται για την προσομοίωση και επαλήθευση της λειτουργίας του 8-bit μετρητή. Δημιουργήσαμε σήματα ελέγχου (ρολόι, reset, load, start/stop) καθώς και την είσοδο δεδομένων, ενώ η έξοδος παρακολουθείται μέσω των σημάτων count και carry. Το ρολόι το δημιουργήσαμε με εναλλαγή κάθε 10 ns, ενώ η ακολουθία της προσομοίωσης περιλαμβάνει αρχικοποίηση, ενεργοποίηση reset, φόρτωση τιμής, ενεργοποίηση μέτρησης, και επαναφόρτωση νέων τιμών σε διάφορες χρονικές στιγμές. Μέσω αυτών των βημάτων ελέγχεται πλήρως η συμπεριφορά της σχεδίασης σε διαφορετικές συνθήκες λειτουργίας, επιβεβαιώνοντας τη σωστή cascade μέτρηση και τη λειτουργία των σημάτων φόρτωσης, reset και start/stop.

γ) Ακολουθούν κάποια στιγμιότυπα επαλήθευσης της ορθής λειτουργίας

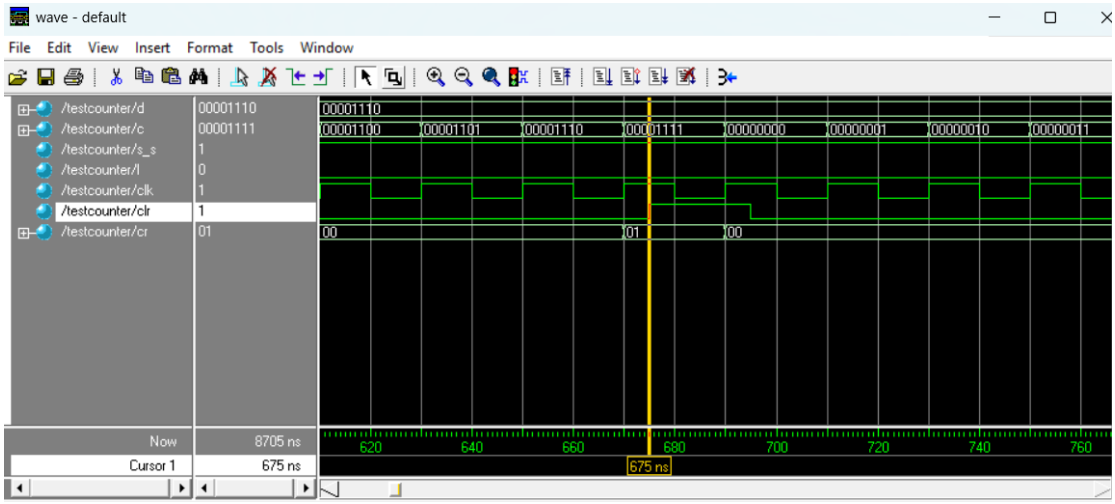


Εδώ φαίνεται αρχικά στα 45 ns ότι ενεργοποιείται το σήμα load και φορτώνεται η τιμή που έχουμε ορίσει και έπειτα στα 110 ns φαίνεται ότι ξεκινά η μέτρηση με την ενεργοποίηση του σήματος start_stop. Στην επόμενη θετική ακμή ενεργοποιείται και ο 2ος μετρητής καθώς το κρατούμενο cr[0] είναι 1 και ξεκινά η μέτρηση στον 2ο 4-bit μετρητή.





Εδώ βλέπουμε την λειτουργία του σύγχρονου καθαρισμού, δηλαδή στα 315 ns me 325 επειδή η θετική ακμή ρολογιού(330 ns) δεν «βλέπει» το σήμα καθαρισμού στο 1, τότε δεν πραγματοποιείται ο καθαρισμός. Ενώ στα 365 ns η θετική ακμή ρολογιού (370 ns) «βλέπει» το 1 και γίνεται κανονικά ο καθαρισμός.



Εδώ βλέπουμε τη σωστή λειτουργία του καθαρισμού που μηδενίζει το κρατούμενο και αποτρέπει έτσι τη λανθασμένη λειτουργία της μέτρησης, δηλαδή έπειτα από τον καθαρισμό να πήγαινε πχ 00000000 σε 00010001 κάτι που θα ήταν λάθος.

6)