Παναγιώτου Δήμητρα

ΑΕΜ 2595

ECE333

LAB 2(UART)

Στόχος της δεύτερης εργαστηριακής άσκησης είναι η υλοποίηση ενός σειριακού συστήματος επικοινωνίας αποστολέα – παραλήπτη. Το σύστημα αποτελείται από ένα transmitter και ένα receiver τα οποία μέσω ενός καναλιού το οποίο είναι διαθέσιμο όταν το sample_ENABLE είναι high.

Αναλυτικότερα:

PartA

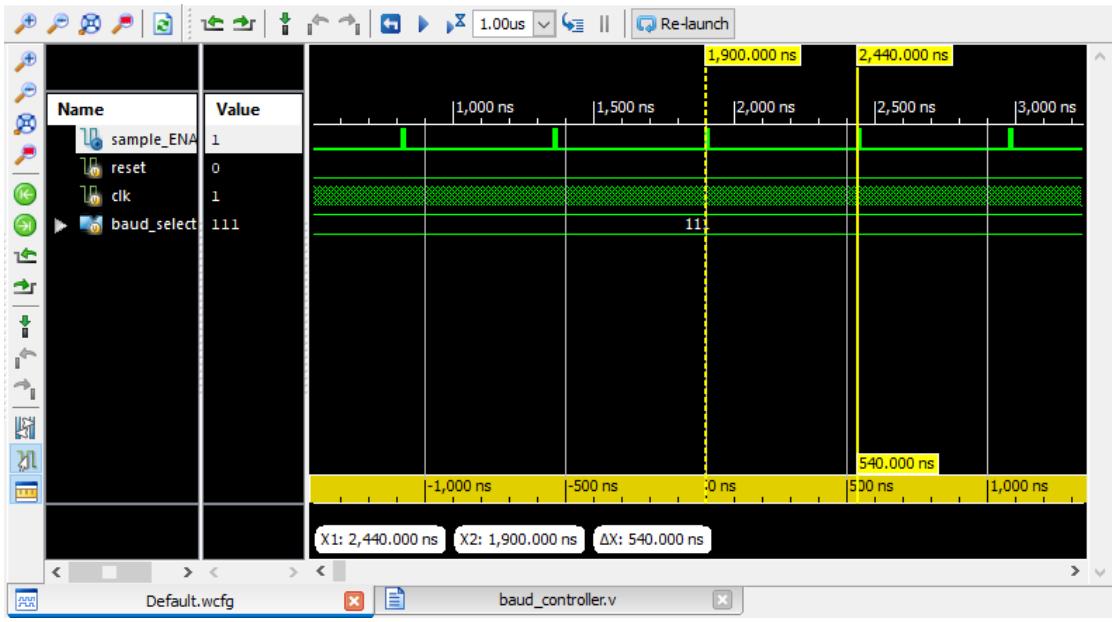
Στο πρώτο μέρος ζητήθηκε η υλοποίηση ενός baud_controller κυκλώματος , σκοπός του οποίου είναι να παρέχει στον transmitter και στον receiver την πληροφορία για το σε ποια συχνότητα θα μεταδίδει ο ένας και θα λαμβάνει ο άλλος. Πιο απλά ο baud_controller παράγει ένα σήμα sample_ENABLE το οποίο είναι θετικά ενεργό και μένει ενεργό για ένα κύκλο με συχνότητα 16× Baud Rate.

Για τις παρακάτω τιμές baud_rate παρουσιάζονται οι μέγιστες τιμές του counter για συχνότητα ρολογιού 100MHz.

Baud_sel	Baud_Rate	Calculation	MAX COUNTER VALUE	ERROR
000	300bit/sec	1/(16*300)=1/4800=208.333ns	20833	208.33333-208.333=0.0003ns
001	1200bit/sec	1/(16*1200)=1/19200=52.083ns	5208	52.0832-52.080=0.003ns
010	4800bit/sec	1/(16*4800)=1/76800=13.020ns	1302	13.0204-13.02=0.0004ns
011	9600bit/sec	1/(16*9600)=1/153600=6.510ns	651	6.51032-6.51=0.00032ns
100	19200bit/sec	1/(16*19200)=1/307200=3.255ns	326	3.255-3.26=0.005ns
101	38400bit/sec	1/(16*38400)=1/614400=1.627	163	1.6273-1.63=0.0027ns
110	57600bit/sec	1/(16*57600)=1/921600=1.085ns	108	1.0854-1.08=0.0054ns
111	115200bit/sec	1/(16*115200)=1/1843200=0.542ns	54	0.542-0.54=0.002ns

Υλοποίηση: Το κύκλωμα δέχεται σαν είσοδο έναν καταχωρητή baud_select 3 bit. Σε κάθε τιμή του baud_select αντιστοιχεί ένα bandwith στο οποίο θα συμφωνήσουν transmitter-receiver για την επικοινωνία τους. Χρησιμοποιώντας την πληροφορία για την ταχύτητα μετάδοσης των δεδομένων , το κύκλωμα δημιουργεί ένα σήμα sampling_enable το οποίο έχει διάρκεια έναν κύκλο ρολογιού. Το σήμα αυτό είναι και η έξοδος του κυκλώματος. Πιο συγκεκριμένα η υλοποίηση περιλαμβάνει ένα always block που ενεργοποιείται κάθε φορά που αλλάζει το baud_select .Στο ίδιο block περιέχεται ένα case statement για την επιλογή του μέγιστου αριθμού κύκλων ρολογιού ώστε επιτευχθεί η προσυμφωνημένη ταχύτητα μετάδοσης. Επιπλέον χρησιμοποιείται ένα always block όπου αλλάζουν οι τιμές του μετρητή. Τέλος υπάρχει ένα ακόμα always block που διαχειρίζεται το σήμα sampling_enable σύμφωνα με την τιμή του μετρητή.

Επαλήθευση λειτουργίας:



partB

Στο μέρος Β υλοποιήθηκε ο αποστολέας (transmitter.v) , ένα κύκλωμα το οποίο αποτελείται από 6 always blocks τα οποία αναλυτικά υλοποιούν τις παρακάτω λειτουργίες:

ΥΛΟΠΟΙΗΣΗ:

- Το πρώτο always block σε συνδυασμό με το δεύτερο always block υλοποιεί μία δομή ελέγχου , η οποία αποσκοπεί να μεταβιβάσει το κύκλωμα στο κατάλληλο nextState (και έπειτα με κατάλληλη ανάθεση στο currentState) ανάλογα με τα σήματα που είναι ενεργοποιημένα. Το block αυτό ενεργοποιείται σε κάθε αλλαγή τιμής των σημάτων Tx_EN , Tx_WR , start_transmission , transmission_ended και reset και περιλαμβάνει εσωτερικά if-else statements για τον έλεγχο των σημάτων.
- Το τρίτο always block ενεργοποιείται σε κάθε αλλαγή του currentState και έχει ως στόχο να ειδοποιήσει το κύκλωμα ότι έχει υπάρχουν έτοιμα διαθέσιμα δεδομένα για μεταβίβαση.

- Το τέταρτο always block χρησιμοποιείται για την ανάθεση των δεδομένων σε ένα buffer , ο οποίος σε επόμενο στάδιο θα χρησιμοποιηθεί για την αποστολή των δεδομένων. Το συγκεκριμένο block ενεργοποιείται οποτεδήποτε έχουμε θετική ακμή του σήματος Tx_WR (ενημερώνει πότε υπάρχουν διαθέσιμα δεδομένα και έχει διάρκεια έναν κύκλο ρολογιού).
- Το πέμπτο always block παράγει τον παλμό transmit_ENABLE με τον οποίο πραγματοποιείται η αποστολή των δεδομένων. Το block αυτό ενεργοποιείται σε κάθε θετική ακμή του σήματος Tx_sample_ENABLE που παράγεται από το baud_controller σύμφωνα με την προσυμφωνημένη ταχύτητα μετάδοσης.
- Το έκτο always block και τελευταίο για την υλοποίηση του uart_transmitter ενεργοποιείται σε κάθε θετική ακμή του transmit_ENABLE και έχει ως σκοπό να περάσει τα δεδομένα στην έξοδο. Στο τέλος της μετάδοσης θέτει το σήμα transmission_ended ίσο με 1 που σηματοδοτεί την λήξη της μετάδοσης καθώς και την κατάλληλη αλλαγή του state.

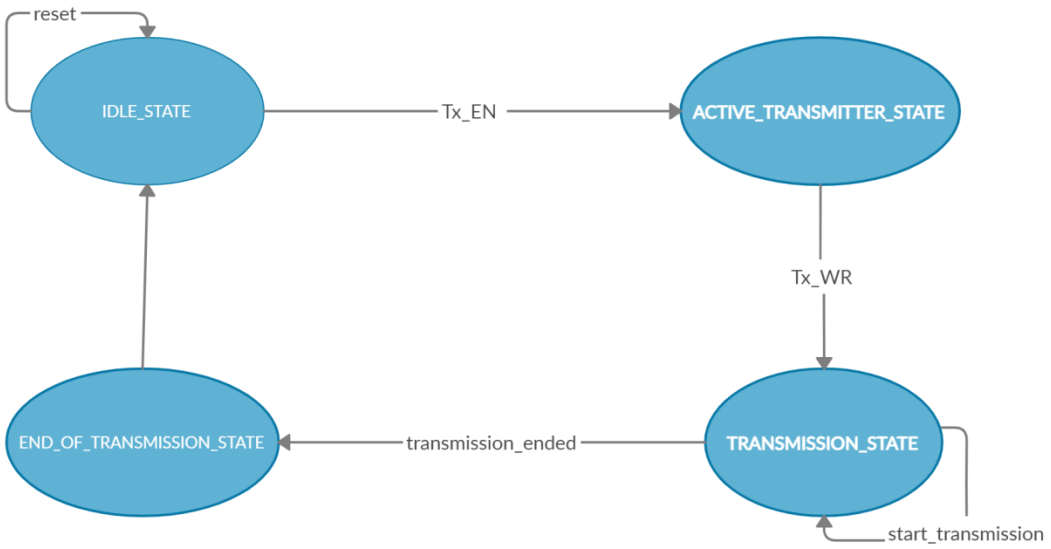
*Στη υλοποίηση του transmitter χρησιμοποιήθηκε και ο baud_controller του μέρους Α

Στην υλοποίηση του transmitter χρησιμοποιήθηκε ένα ακόμα module , το resetSychronizer με σκοπό να συγχρονίσει το σήμα reset με το clk.Στόχος του είναι να δημιουργεί παλμό reset σε οποιαδήποτε θετική ακμή του clk κρίνεται απαραίτητο.Υλοποίηση:** Η υλοποίηση του resetSychronizer περιλαμβάνει δύο always statements έτσι ώστε σε κάθε κύκλο ρολογιού να γίνεται ανάθεση του reset σε μια προσωρινή μεταβλητή και έπειτα στον επόμενο κύκλο να ανατίθεται η προσωρινή μεταβλητή στη νέα τιμή του reset. (Η προσωρινή μεταβλητή κρίνεται απαραίτητη για να αποφευχθούν φαινόμενα μεταστάθειας.)

<u>FSM διάταξη για τον transmitter:</u>

Για την ορθή λειτουργία του transmitter κρίνεται απαραίτητη η δημιουργία μίας μονάδας κατάστασεων. Οι καταστάσεις τις οποίες περιλαμβάνει είναι οι παρακάτω:

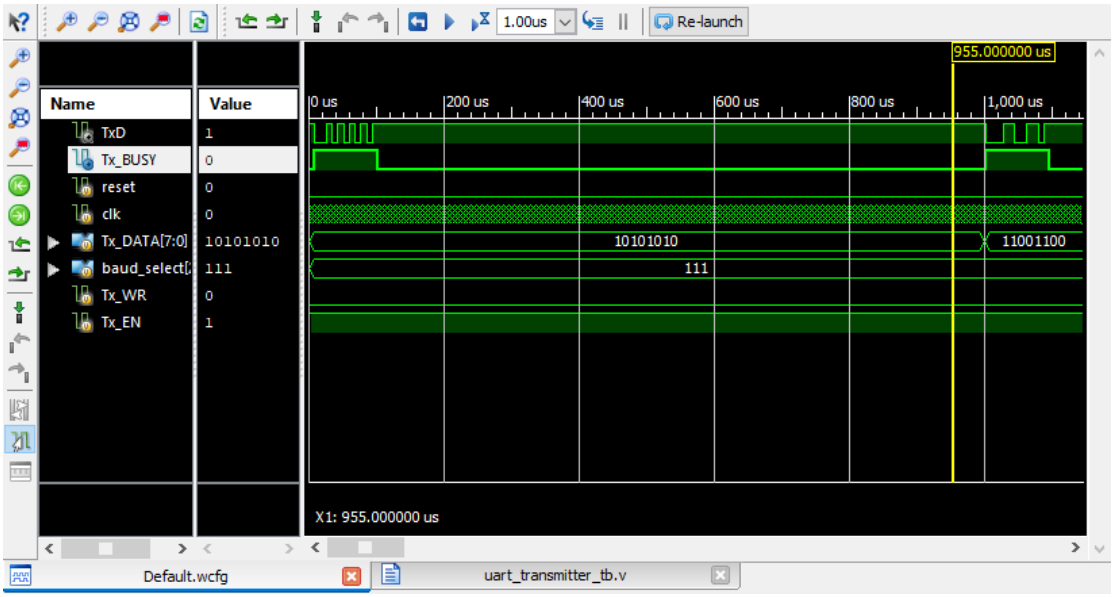
- **IDLE_STATE**: Ο transmitter μεταβαίνει σε αυτή την κατάσταση όσο λαμβάνει reset=1.
- **ACTIVE_TRANSMITTER_STATE**: Το κύκλωμα είναι έτοιμο να αποστείλει δεδομένα εφόσον είναι ενεργό και το σήμα Tx_EN διατηρεί την τιμή 1.
- **TRANSMISSION_STATE**: Το κύκλωμα είναι έτοιμο να αποστείλει δεδομένα εφόσον είναι ενεργό και έχει λάβει θετική ακμή του σήματος Tx_WR όπου ειδοποιεί το σύστημα ότι υπάρχουν έτοιμα δεδομένα προς αποστολή(Tx_DATA). Το κύκλωμα παραμένει σε αυτή την κατάσταση όσο το σήμα start_transmisision (το οποίο αρχικοποιείται στην τιμή 1 στην αρχή αυτού του σταδίου) συνεχίζει να διατηρείται ψηλά.
- **END_OF_TRANSMISSION_STATE**: Η μετάβαση στο END_OF_TRANSMISSION_STATE γίνεται στην θετική ακμή του σήματος transmission_ended και το σύστημα διατηρείται στη κατάσταση αυτή μέχρι τον επόμενο παλμό του transmit_ENABLE. Αν και έφοσον δεν υπάρξει νέος παλμός Tx_WR το σύστημα μεταβαίνει στην κατάσταση IDLE_STATE όπου και επαναρχικοποιείται για την επόμενη είσοδο.



ΕΠΑΛΗΘΕΥΣΗ ΛΕΙΤΟΥΡΓΙΑΣ:

Στο testbench δίνονται τρία πακέτα δεδομένων προς αποστολή:

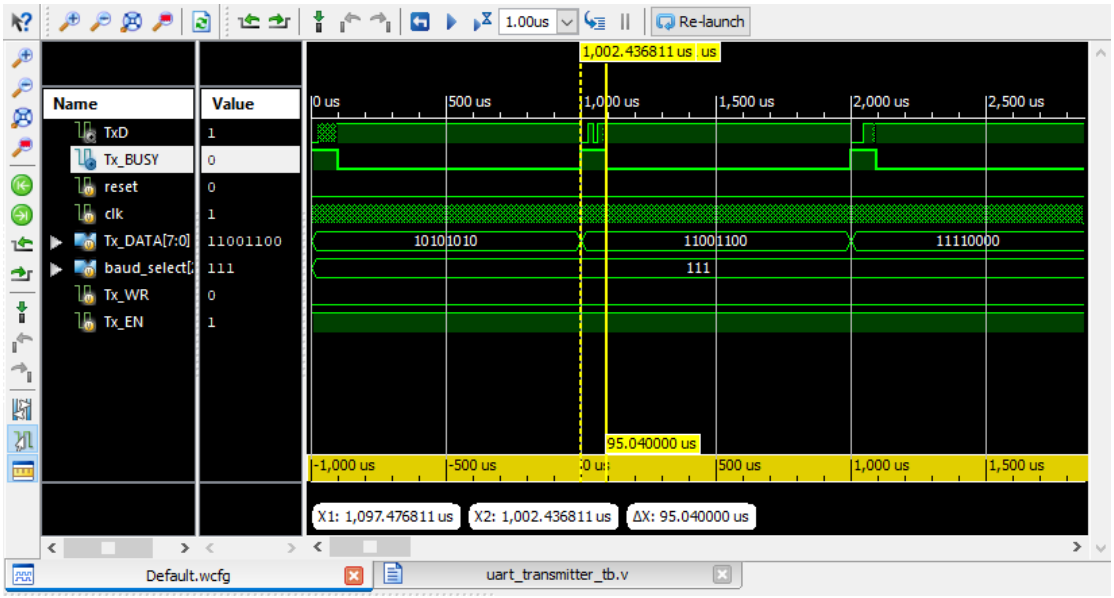
1. 10101010
2. 11001100
3. 11110000



Behavioral simulation partB

Η διάρκεια μετάδοσης ισούται με για baud_select=3’b111

540ns * 16(cycles per transmit_ENABLE) *11(bit per transmission) = 95.040ns



Post and route simulation partB

partC

Στο μέρος C υλοποιήθηκε μία δομή λήψης δεδομένων (receiver.v) , ένα κύκλωμα το οποίο αποτελείται από 6 always blocks τα οποία αναλυτικά υλοποιούν τις παρακάτω λειτουργίες:

- Το πρώτο always block σε συνδυασμό με το δεύτερο always block υλοποιεί μία μονάδα ελέγχου , η οποία έχει στόχο να μεταβιβάσει το κύκλωμα στο κατάλληλο nextState (και έπειτα με κατάλληλη ανάθεση στο currentState) ανάλογα με τα σήματα που είναι ενεργοποιημένα. Το block αυτό ενεργοποιείται σε κάθε αλλαγή τιμής των σημάτων Rx_EN , RxD , start_bit , Rx_DONE , check_done , και reset και περιλαμβάνει εσωτερικά if-else statements για τον έλεγχο των σημάτων.
- Το τρίτο always block διαχειρίζεται την διαδικασία δημιουργίας παλμού για την λήψη του δείγματος. Σε περίπτωση , λοιπόν , που η μονάδα είναι ενεργή (Rx_EN=1) δημιουργείται ένας παλμός take_sample κάθε 8 θετικές ακμές του Rx_sample_ENABLE , ένα σήμα που προκύπτει από το baud_controller module ανάλογα με την προσυμφωνημένη ταχύτητα μετάδοσης. Ο παλμός για την λήψη του δείγματος δημιουργείται στην μέση του παλμού μετάδοσης για μεγαλύτερη ακρίβεια στην τιμή που λαμβάνεται.
- Το τέταρτο always block πραγματοποιεί την δειγματοληψία και ενεργοποιείται σε κάθε θετική ακμή του take_sample . Η διαδικασία της δειγματοληψίας υλοποιείται με ένα μετρητή δεδομένων ο οποίος αρχικοποιείται στην τιμή 0 και αυξάνει από την στιγμή που εντοπιστεί low RxD . Παράλληλα γίνονται οι κατάλληλες αναθέσεις στα start_bit , stop_bit , parity_bit και read_data ανάλογα με την τιμή του μετρητή. Όταν ο μετρητής φτάσει στην τιμή ‘b1010(δηλαδή ‘d10) και ανατεθεί η σωστή τιμή στο stop_bit το κύκλωμα σηκώνει τον παλμό Rx_DONE για να ενημερώσει το σύστημα ότι η διαδικασία λήψης έχει λάβει τέλος.
- Το πέμπτο always block υλοποιεί μια δομή ελέγχου για την εγκυρότητα των δεδομένων που μόλις λάβαμε και ενεργοποιείται με κάθε αλλαγή στην τιμή του Rx_DONE ή του reset. Εσωτερικά

πραγματοποιεί ελέγχους στις τιμές των start_bit και stop_bit για να εντοπίσει πιθανά frame errors που πραγματοποιήθηκαν κατά την λήψη. Σε επόμενο βήμα ελέγχεται και το parity_bit για να ελεγχθεί πιο σχολαστικά το read_data για την ακρίβεια των δεδομένων του. Με την ολοκλήρωση των ελέγχων και εφόσον δεν έχει εντοπιστεί κάποιο λάθος το κύκλωμα αυτό ειδοποιεί το σύστημα ότι τα δεδομένα του read_data είναι έγκυρα , αναθέτοντας στο Rx_VALID την τιμή 1.

- Το έκτο always block ενεργοποιείται σε κάθε θετική ακμή του Rx_VALID και η λειτουργία που πραγματοποιεί είναι η ανάθεση του read_data στο Rx_DATA.
  Σε περίπτωση που δεν δημιουργηθεί παλμός Rx_VALID τα δεδομένα δεν αλλάζουν και διατηρούν την προηγούμενη τιμή τους.

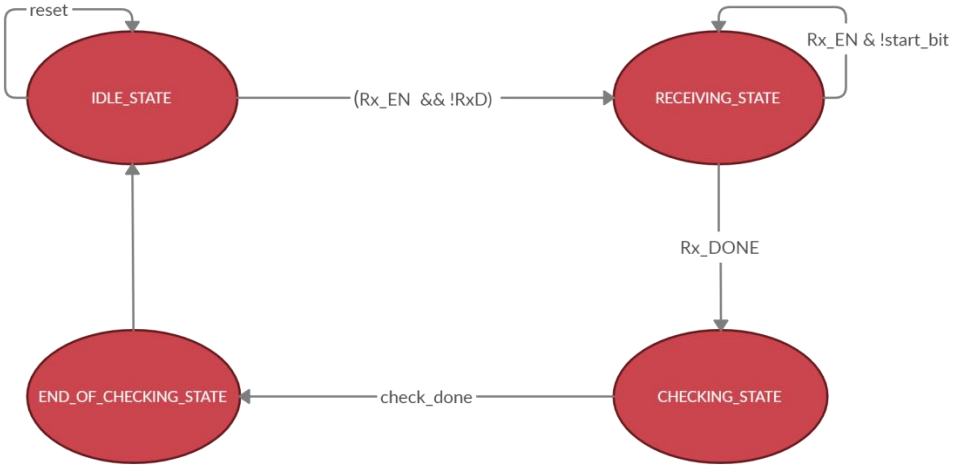
*Στη υλοποίηση του receiver χρησιμοποιήθηκε και ο baud_controller του μέρους Α

Στην υλοποίηση του receiver χρησιμοποιήθηκε ένα ακόμα module , το resetSychronizer με σκοπό να συγχρονίσει το σήμα reset με το clk.Στόχος του είναι να δημιουργεί παλμό reset σε οποιαδήποτε θετική ακμή του clk κρίνεται απαραίτητο.Υλοποίηση:** Η υλοποίηση του resetSychronizer περιλαμβάνει δύο always statements έτσι ώστε σε κάθε κύκλο ρολογιού να γίνεται ανάθεση του reset σε μια προσωρινή μεταβλητή και έπειτα στον επόμενο κύκλο να ανατίθεται η προσωρινή μεταβλητή στη νέα τιμή του reset. (Η προσωρινή μεταβλητή κρίνεται απαραίτητη για να αποφευχθούν φαινόμενα μεταστάθειας.)

<u>FSM διάταξη για τον receiver:</u>

Για την ορθή λειτουργία του receiver κρίνεται απαραίτητη η δημιουργία μίας μονάδας κατάστασεων. Οι καταστάσεις τις οποίες περιλαμβάνει είναι οι παρακάτω:

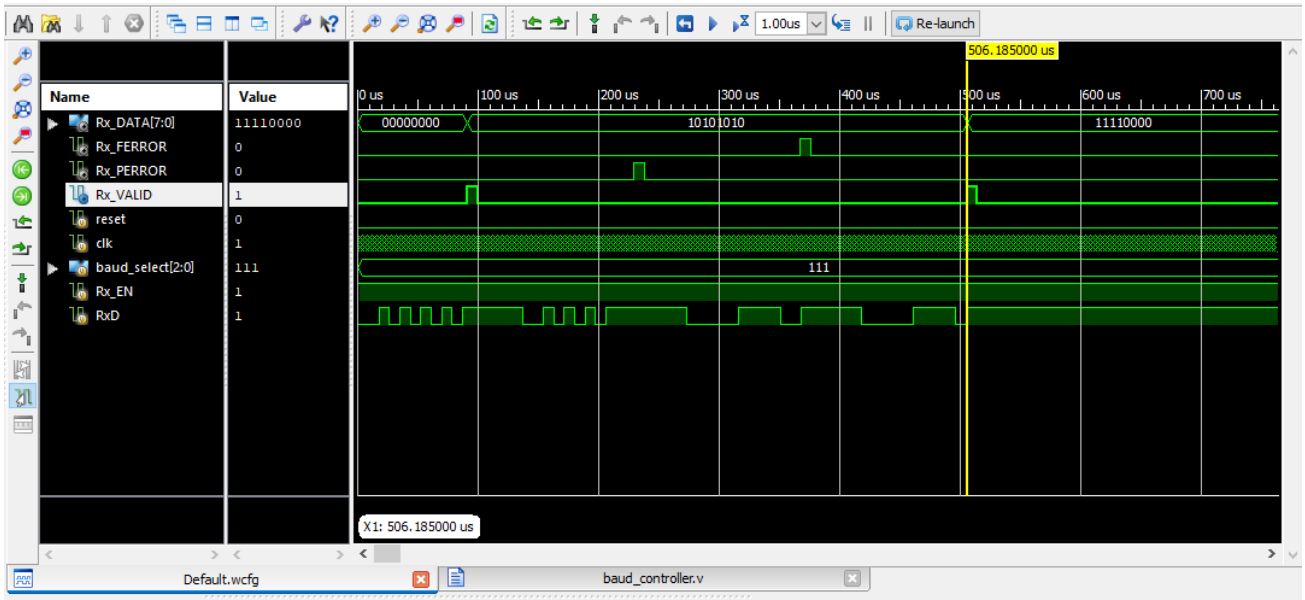
- **IDLE_STATE**: Ο receiver μεταβαίνει σε αυτή την κατάσταση όσο λαμβάνει reset=1.
- **RECEIVING_STATE**: Το κύκλωμα είναι έτοιμο να λάβει δεδομένα εφόσον είναι ενεργό και παράλληλα η είσοδος RxD είναι 0. Το κύκλωμα παραμένει σε αυτή την κατάσταση όσο το start_bit συνεχίζει να διατηρεί την τιμή 0.
- **CHECKING_STATE**: Για την μετάβαση του κυκλώματος σε αυτό το στάδιο απαιτείται η λήψη θετικής ακμής του σήματος Rx_DONE που βεβαιώνει ότι έχει ολοκληρωθεί η λήψη των δεδομένων . Ταυτόχρονα αποτελεί το έναυσμα για να αρχίσουν οι διαδικασίες ελέγχου των δεδομένων.
- **END_OF_CHECKING_STATE**: Πρακτικά η μετάβαση από το CHECKING_STATE στο END_OF_CHECKING_STATE γίνεται ακαριαία γιατί οι έλεγχοι που πραγματοποιούνται γίνονται σχεδόν αμέσως και έτσι το σήμα check_done γίνεται 1 σχεδόν αμέσως μετά την λήψη του Rx_DONE.Στην επόμενη θετική ακμή του take_sample το σύστημα μεταβαίνει στην κατάσταση IDLE_STATE μέχρι την επόμενη RxD=0 είσοδο.



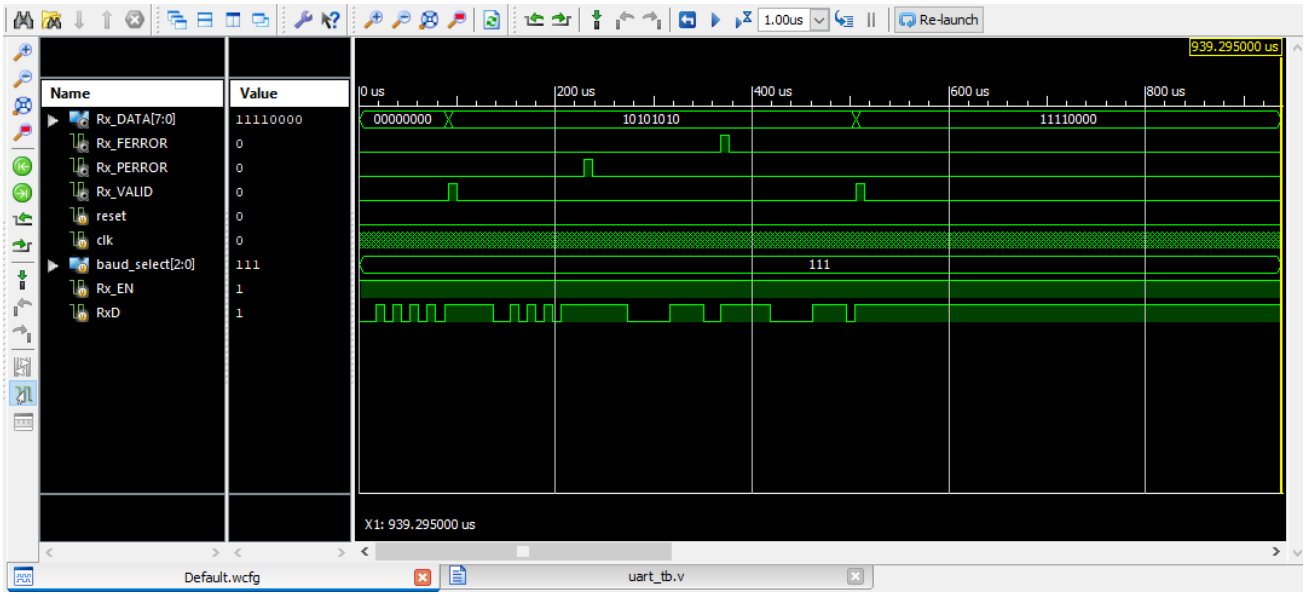
ΕΠΑΛΗΘΕΥΣΗ ΛΕΙΤΟΥΡΓΙΑΣ:

Στο testbench δίνονται τέσσερα πακέτα δεδομένων:

4. 01010101 με έγκυρο parity_bit και πλήρες frame
5. 01010101 με λανθασμένο parity_bit και πλήρες frame
6. 00001111 με έγκυρο parity_bit και λανθασμένο frame
7. 00001111 με έγκυρο parity_bit και πλήρες frame



Behavioral Simulation partC



Post and Route Simulation partC

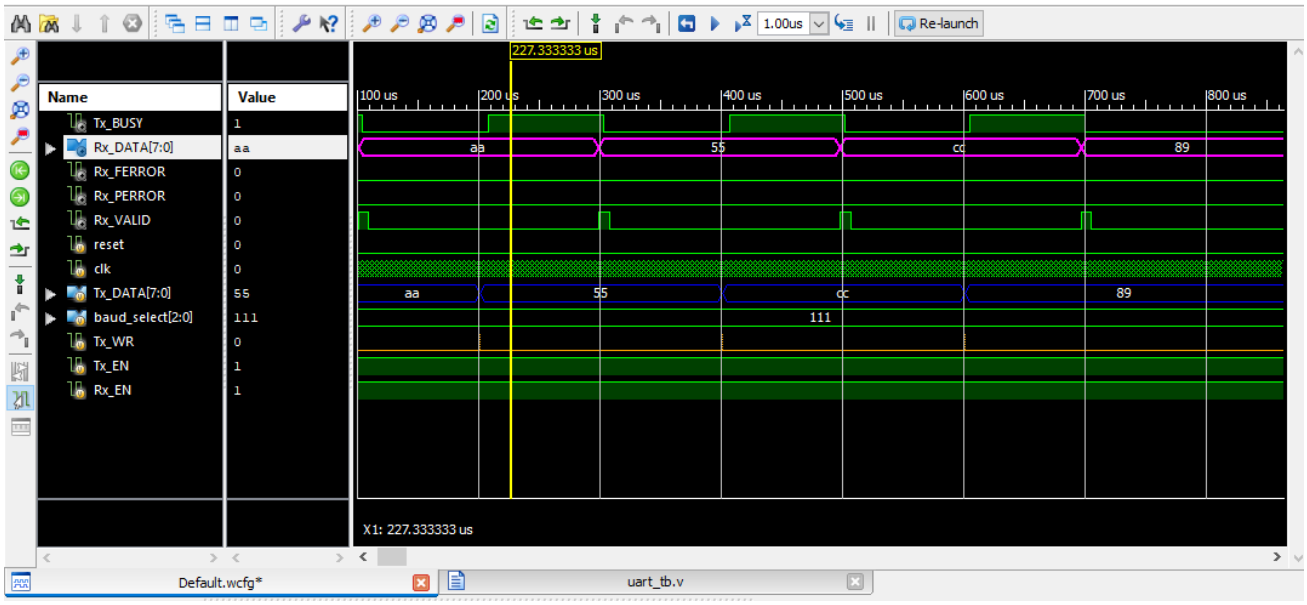
partD

Στο τελευταίο μέρος της εργασίας πραγματοποιήθηκε η σύνδεση των module transmitter και receiver που υλοποιήθηκαν στο δεύτερο και στο τρίτο μέρος της εργασίας. Η έξοδος TxD του transmitter χρησιμοποιήθηκε σαν είσοδος (RxD) στον receiver έτσι ώστε τα δεδομένα που αποστάλθηκαν από τον transmitter να παραληφθούν από τον receiver.

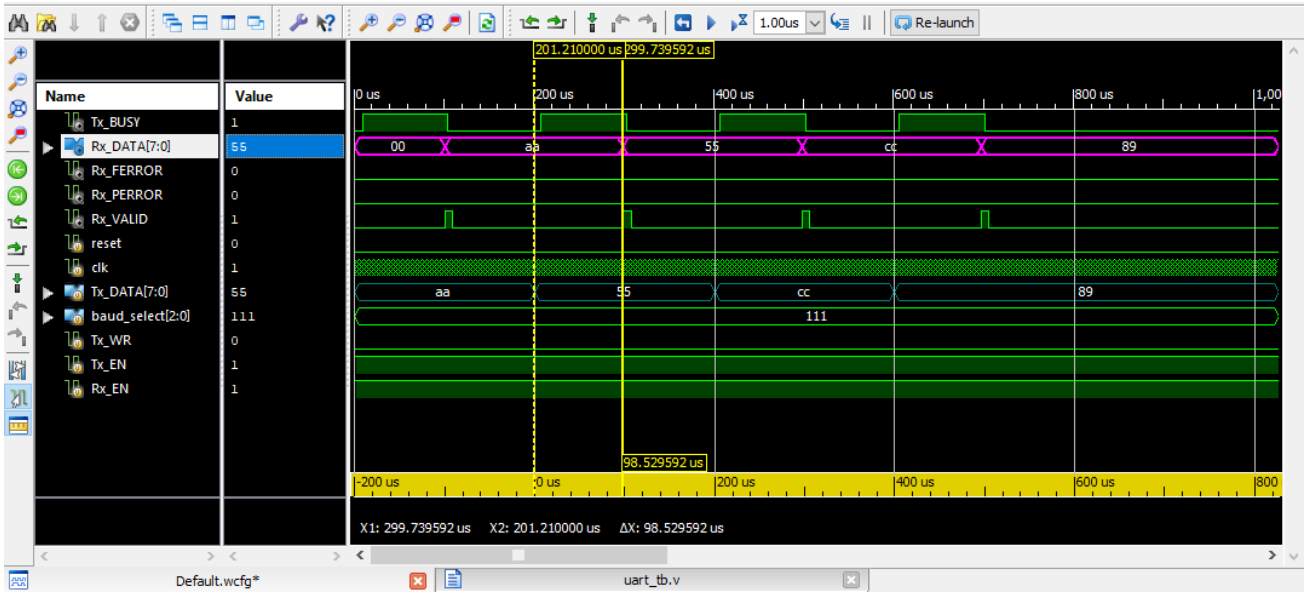
ΕΠΑΛΗΘΕΥΣΗ ΛΕΙΤΟΥΡΓΙΑΣ:

Στο testbench δίνονται τρία πακέτα δεδομένων προς αποστολή:

1. 10101010 (‘haa)
2. 01010101 (‘h55)
3. 11001100 (‘hcc)
4. 10001001 (‘h89)



Behavioral Simulation partD



Post and Route Simulation partD