2η Εργαστηριακή Άσκηση-ce430

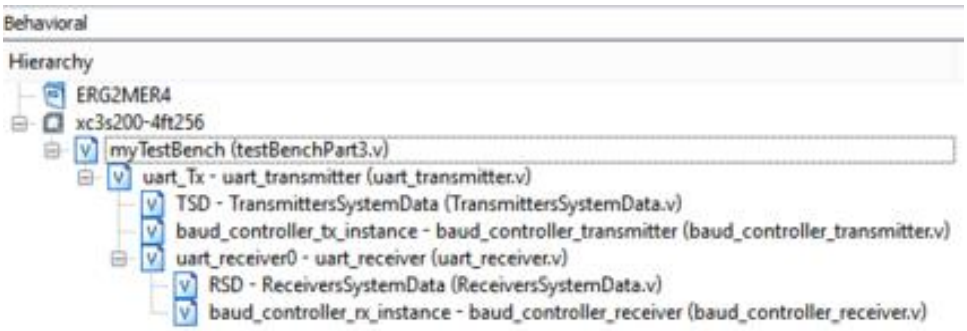
Αξελός Χρήστος | AEM 1814 | caxelos@inf.uth.gr|16/11/16

ΣΤΟΧΟΙ

- Υλοποίηση του πρωτοκόλλου επικοινωνίας UART, transmitter/receiver
- Χωρίζεται η διαδικασί σε 4 μέρη, 1ο η υλοποίηση του baud_rat, 2ο η υλοποίηση του Transmitter, 3ο η υλοποίηση του receiver και 4ο η συνδεση των δύο για αποστολή 1-bit δεδομένων
- Δεν υλοποίησα το κομμάτι που το output γίνεται input στο 7-led-segment της fpga

ΠΕΡΙΓΡΑΦΗ

- Η ιεραρχεία των modules είναι η παρακάτω:



- Τα modules TransmittersSystemData και ReceiversSystemData καλούνται από το uart_transmitter και uart_receiver αντίστοιχα, ενεργοποιούν/απενεργοποιούν την αποστολή/λήψη δεδομένων
- Έκανα διαφορετικά αρχεία για το baud rate του αποστολέα και του παραλήπτη, γιατί πρέπει να χουν διαφορετικό fsm για ένα ίδιο baud_select
- Για την προσομοίωση, έδωσα ως inputs στο uart_receiver τα clock, reset εκτος απτο bit αποστολής απτον παραλήπτη, χωρις αυτό να επηρεάζει την περίπτωση αποστολέας και παραλήπτης να τρέχουν σε διαφορετικά ρολόγια

ΜΕΡΟΣ 1: Υλοποίηση Baud Controller

ΥΛΟΠΟΙΗΣΗ BAUD CONTROLLER ΓΙΑ ΤΟΝ ΑΠΟΣΤΟΛΕΑ

- Για την ενεργοποίηση του σήματος αποστολής, χρησιμοποιώ έναν 18-bit μετρητή. Η max τιμή του μετρητή για κάθε baud select, τα σφάλματα και το πως προέκυψαν αναφέρονται στο αρχείο baud_controller_transmitter.v

- Αρχικά διαιρώ την συχνότητα 50MHz με το baud rate, το οποίο είναι σε bits/sec. Μετά στρογγυλοποιώ το αποτέλεσμα στον κοντινότερο ακέραιο και σημειώνω το σφάλμα. Το σφάλμα γίνεται μεγιστο(0.36) για τα 3 μικρότερα baud rate, ενώ όσο το baud rate αυξάνεται, το σφάλμα μειώνεται(ελαχιστο 0.02). Χρονικά το σφάλμα είναι σφάλμα*περίοδος. Για παράδειγμα, όταν το baud rate είναι baud_rate = 300bits/sec, για ρολόι των 50MHz, γίνεται αποστολή ανά 18'b101000101100001011 κύκλους

- Η υλοποίηση είναι σχετικά απλή, όπου σε ένα always block αυξάνω τον counter κατά ένα. Για σήμα reset στο κύκλωμα χρησιμοποιώ το σήμα Tx_EN, αφού το baud controller λειτουργεί όσο γίνεται αποστολή

ΥΛΟΠΟΙΗΣΗ BAUD CONTROLLER ΓΙΑ ΤΟΝ ΠΑΡΑΛΗΠΤΗ

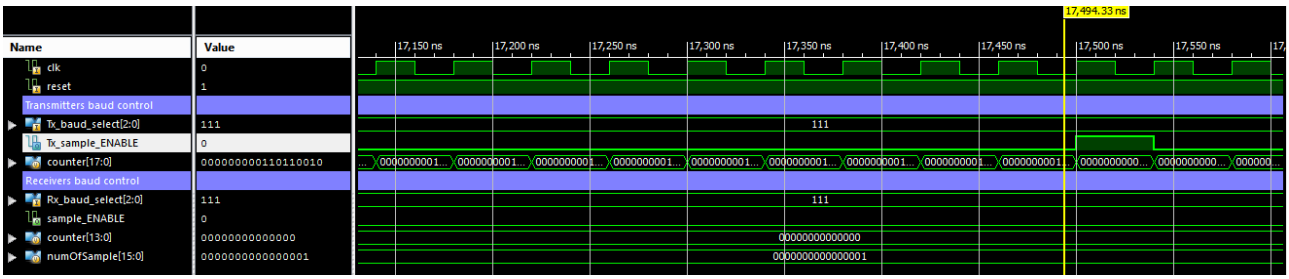
- Μοιάζει με τον αντίστοιχο του αποστολέα, αλλά έχει διαφορετικές max τιμές των counter

- Χρησιμοποιώ έναν 16-bit μετρητή αντί για 18-bit, διότι ο παραλήπτης τρέχει σε ταχύτητα 16 φορές πιο γρήγορη απτόν αποστολέα. Δηλαδή για baud rate = 300bits/sec, για ρολόι 50MHz, μπορώ να κάνω δειγματοληψία ανα 14'b1010001011 κύκλους

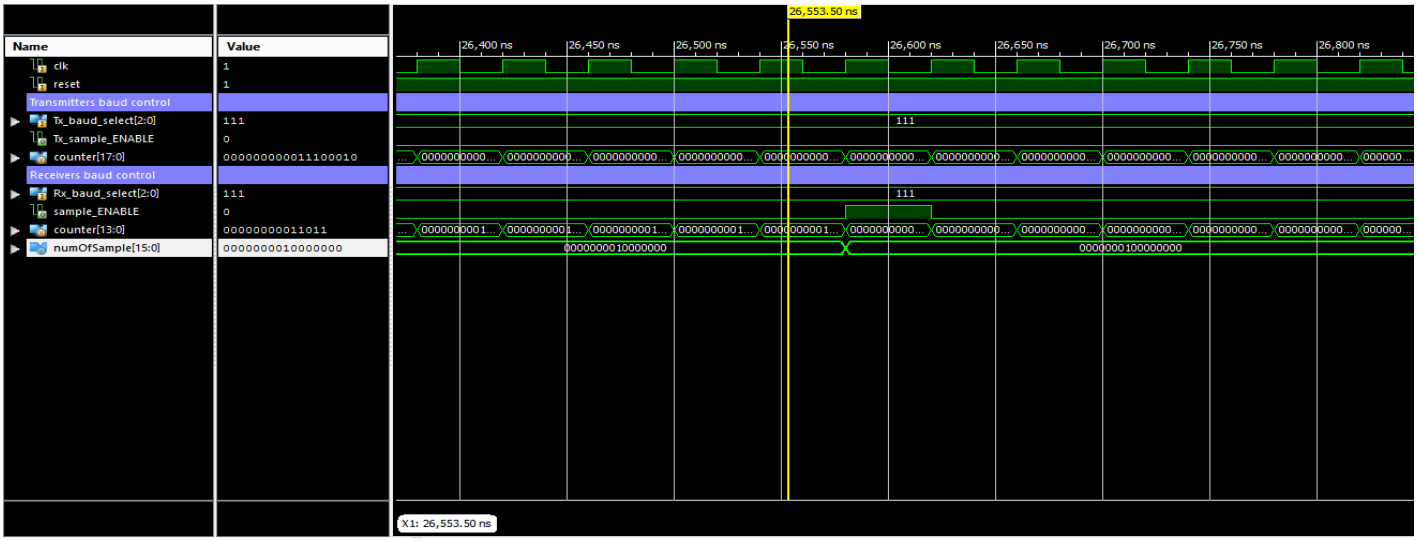
- Επειδή στον παραλήπτη κάνω δειγματοληψία 3 φορές στα bit 7/16, 8/16, και 9/16 για να προκύψει το τελικό bit, χρησιμοποιώ τον counter reg [15:0] numOfSample με one hot κωδικοποίση και αρχική τιμή 16'b0000_0000_0000_0001, τον οποίο κάνω shift κατα ένα ανά 14'b1010001011 κύκλους(για rate 300bits/sec , clk=50mhz), όταν δηλαδή μπορεί να γίνει δειγματοληψία. Μόλις ο numOfSample γίνει 16'b0000_0001_0000_0000, κάνω το σήμα Rx_ENABLE = 1 κάνω δειγματοληψία. Λεπτομέρειες για τον υπολογισμό του τελικού bit στέλνει ο αποστολέας εξηγούνται στο Μέρος 3.

- Η παραπάνω υλοποίηση με one hot με βοήθησε σαυτήν την περίπτωση για να δημιουργήσω το σήμα ελέγχου της δειγματοληψίας numOfSample[8]

Στιγμιότυπο ενεργοποίησης αποστολέα



Στιγμιότυπο ενεργοποίησης δειγματοληψίας



ΜΕΡΟΣ 2: Υλοποίηση uart transmitter

ΥΛΟΠΟΙΗΣΗ

- Κύριο module της υλοποίησης είναι το module uart_transmitter.v, το οποίο συνδέεται εξωτερικά με το uart_receiver.v, όπου στέλνει το bit επικοινωνίας, ενώ εσωτερικά καλεί τα modules TransmittersSystemData, οπου λαμβάνει byte πληροφορίας, το baud_controller για τον χρονισμό της αποστολής κάθε bit στον receiver

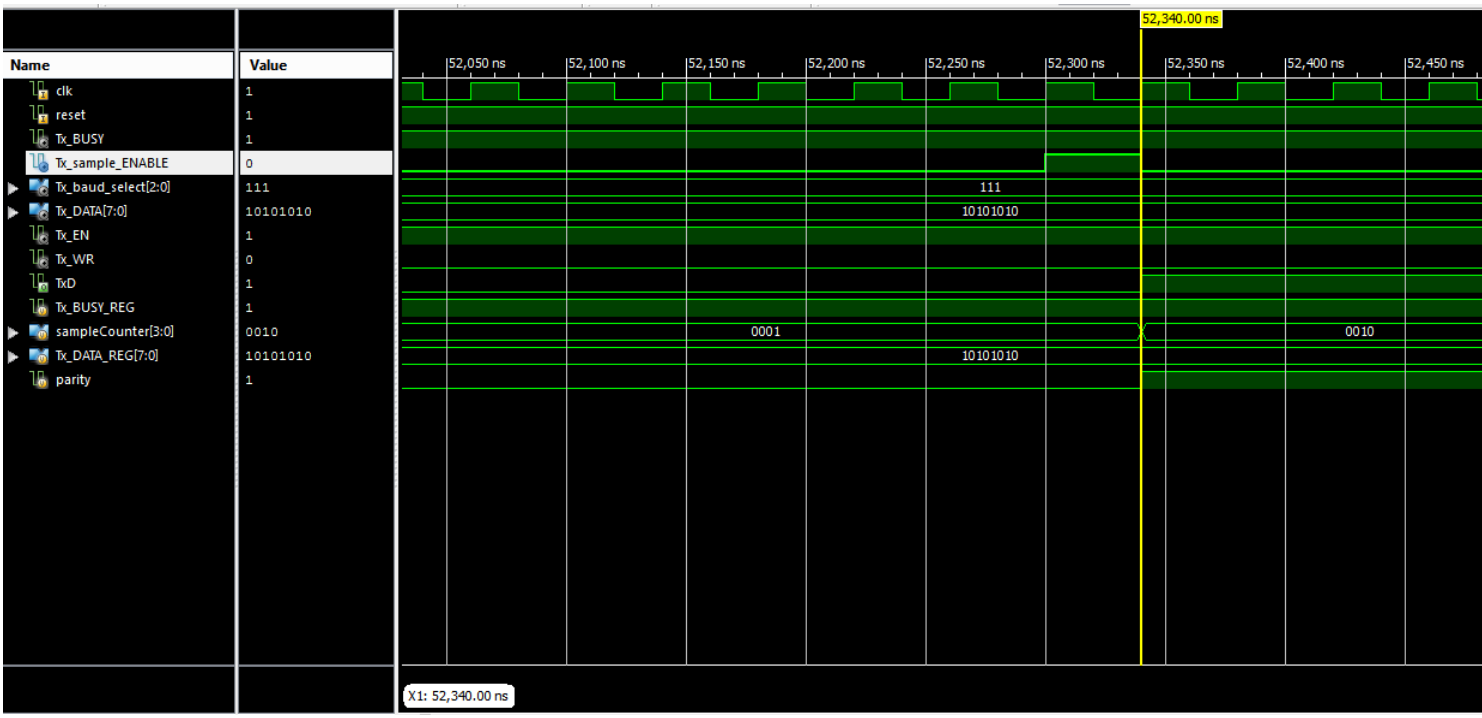
- Για την έναρξη της επικοινωνίας, στέλνω το σήμα Tx_BUSY από τον transmitter στο σύστημα του, για να του δείξει αν στέλνει δεδομένα στον receiver ή όχι

- Εφόσο ο transmitter δεν στέλνει στον receiver(Tx_BUSY=0) και πρέπει να σταλθούν δεδομένα, τότε ενεργοποιεί το σήμα Tx_EN σε ένα και παράλληλα στέλνει ένα byte πληροφορίας για να σταλθεί σειριακά στον receiver μέσω του σήματος Tx_WR=1 για έναν κύκλο ρολογιού

- Ο receiver αναγνωρίζει το Tx_EN=1 και στέλνει το startBit στον receiver, ενώ ειδοποιεί το σύστημα να μην στείλει ακομά το επόμενο byte(Tx_BUSY=1) έως ότου στέλει το stop bit στον receiver και απενεργοποιήσει το Tx_BUSY

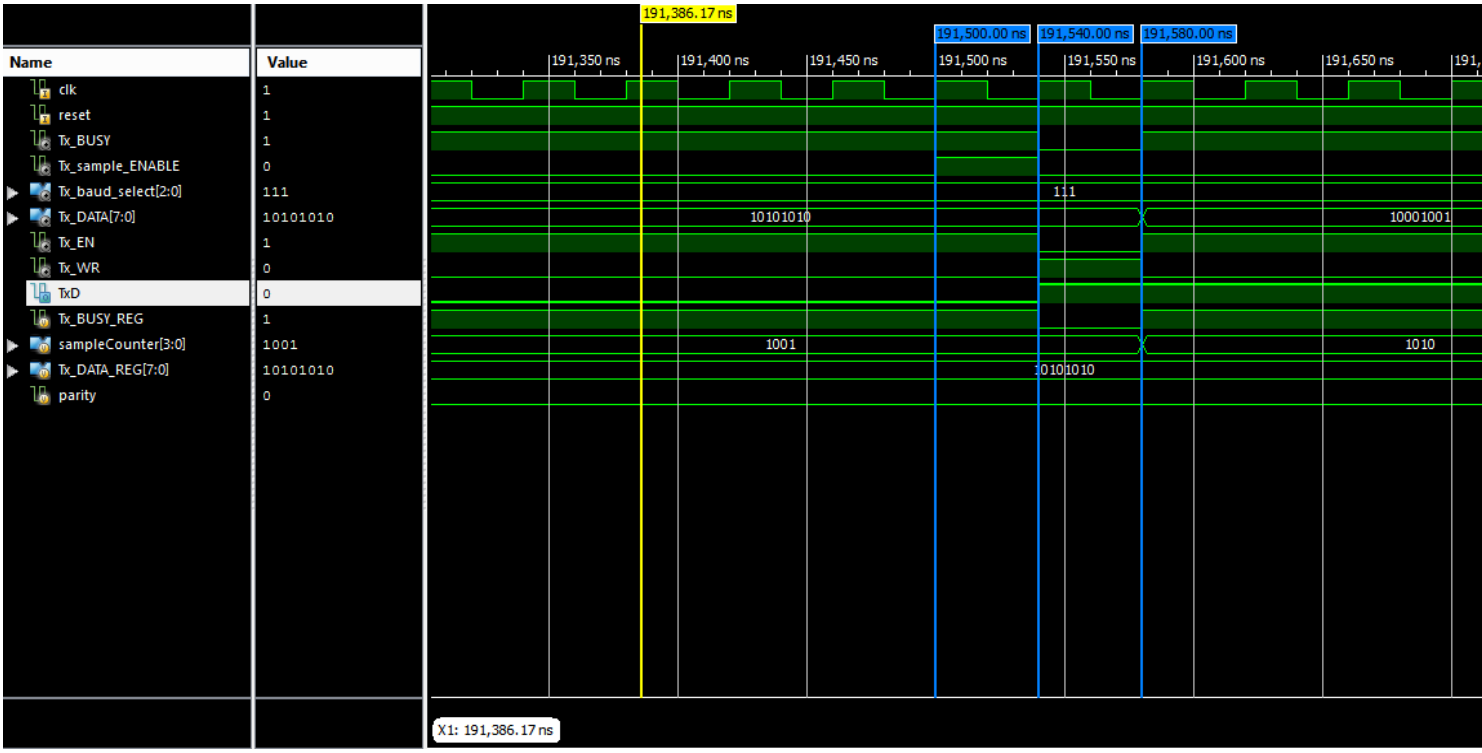
- Στο uart_transmitter.v, χρησιμοποιώ έναν 4bit counter καταστάσεων με αρχική τιμή counter = 4'b0000 και αυξάνεται κατά ένα. Για counter=0, στέλνεται το startBit, για 1μεχρι8 τα δεδομένα, για 9 το parity bit και για 10 το stop μπιτ. Αλλαγές στις καταστάσεις γίνονται μόνο όταν ο αποστολέας είναι ενεργοποιημένος και το σήμα Tx_enable είναι ενεργοποιημένο. Ο υπολογισμός του parity bit προκύπτει απτην συνθήκη if (data = 1) parity = ~parity

- Παρακάτω φαίνονται στιγμιότυπα αποστολής του ψηφίου 1 και φόρτωσης νέας 8άδας απτο σύστημα για αποστολή:



Στο παραπάνω σχήμα, την χρονικη στιγμη 52,300ns ενεργοποιείται το Tx_ENABLE και στον επόμενο κύκλο στέλνεται το bit 1. Ταυτόχρονα μεταβάλλεται και parity bit και ο sampleCounter. Τα υπόλοιπα σήματα δεν μεταβάλλονται αφού συνεχίζεται η αποστολή προς τον δέκτη

Παρακάτω φαίνεται στιγμιότυπο αποστολής του stop bit και αποστολής από το σύστημα της νέας 8αδας bit



- Την χρονική στιγμή 191,500.00 ns, το σύμα Tx_sample_ENABLE γίνεται 1 και στον επόμενο κύκλο στέλνεται στέλνεται το Tx_EN γίνεται 0 και σταματά η επικοινωνία με τον δέκτη. Ταυτόχρονο το Tx_WR γίνεται 1 για να σταλθούν στον επόμενο κύκλο δεδομένα απτο σύστημα. Την στιγμη 191,589, το Tx_DATA γίνεται 10001001 για να σταλθούν αργότερα στον παραλήπτη

- Την χρονική στιγμή 191,500.00 ns, το σύμα Tx_sample_ENABLE γίνεται 1 και στον επόμενο κύκλο στέλνεται στέλνεται το Tx_EN γίνεται 0 και σταματά η επικοινωνία με τον δέκτη. Ταυτόχρονο το Tx_WR γίνεται 1 για να σταλθούν στον επόμενο κύκλο δεδομένα

ΜΕΡΟΣ 3: Υλοποίηση uart receiver

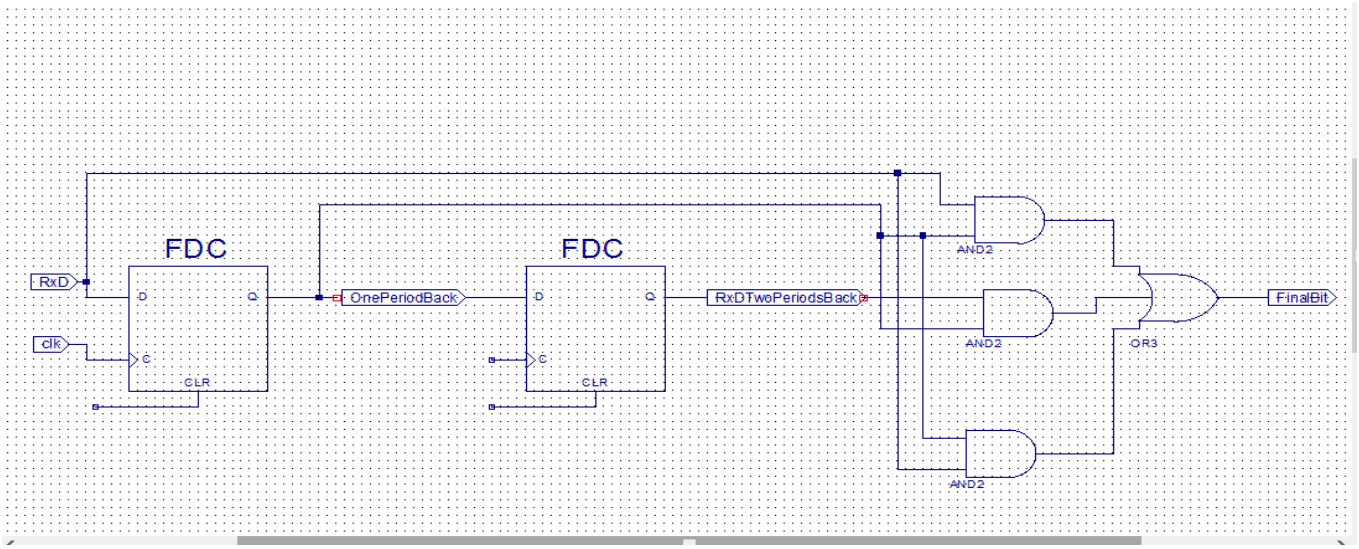
ΥΛΟΠΟΙΗΣΗ

- Κύριο module είναι το uart_receiver.v, το οποίο δέχεται ένα bit πήροφορίας απτόν δέκτη, ενώ καλεί το σύστημα(ReceiversSystemData.v) για την αποθηκευσω των byte

- Αρχικά ανιχνεύω το startBit την ίδια στιγμή που στέλνεται χωρίς καθυστέριση), χρησιμοποιώντας 2 flip flop όπως με το debounce για να ανιχεύσω την αλλαγή

- Μόλις ανιχνευτεί το startBit, ενεργοποιείται ο baud_controller του receiver που αναφέρεται πιο πριν, έχοντας ως σήμα reset το Rx_EN=1

- Για την δειγματοληψία χρησιμοποιώ τα 3 κεντρικότερα bit



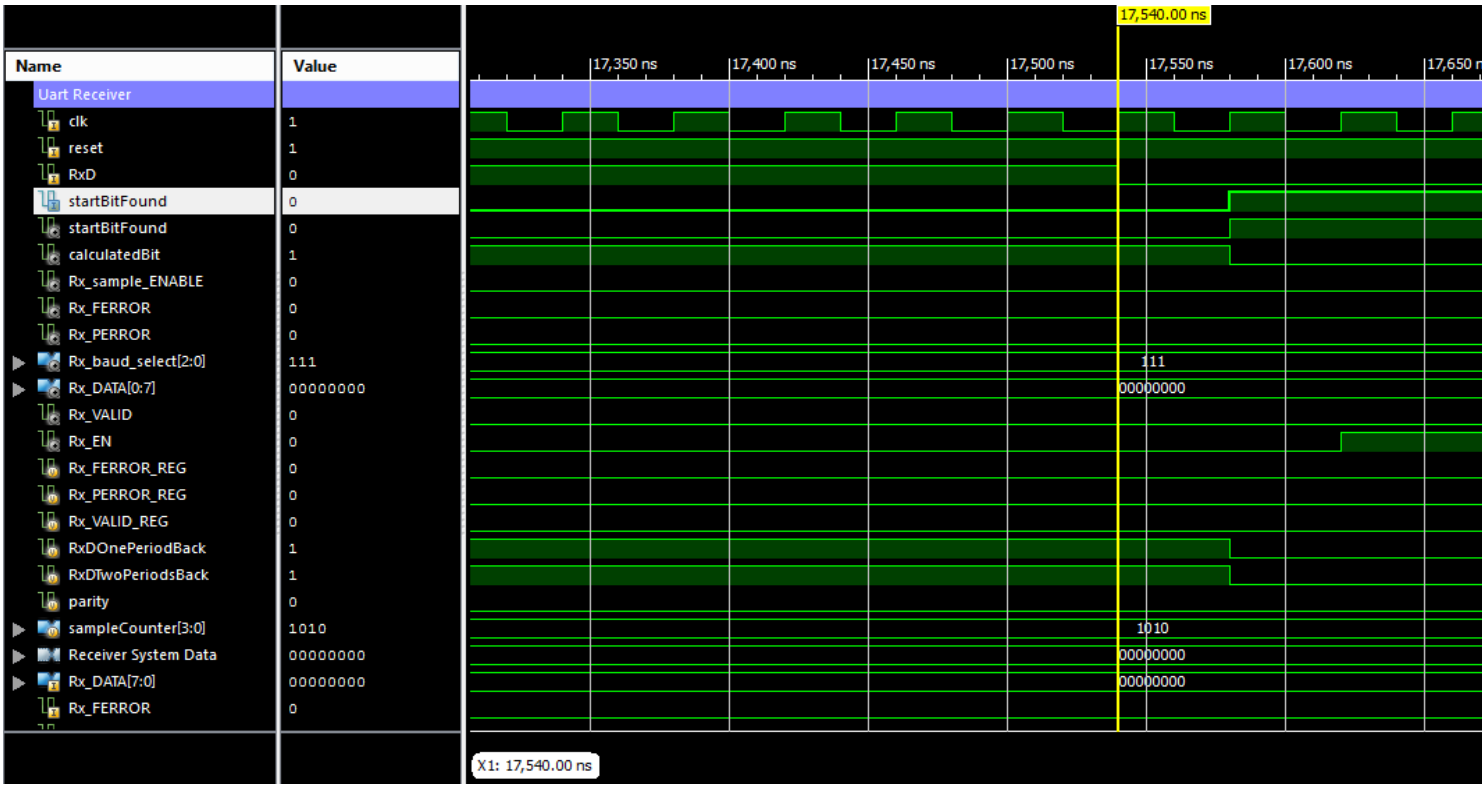
- Υπάρχουν αντίστοιχα σήματα στον δέκτη, Rx_EN και Rx_Sample_EN, για ενεργοποίηση του δέκτη και της δειγματοληψίας

- Στο uart_receiver.v κάνω τους ελέγχους για την εγκυρότητα των δεδόμενων, αν υπολογιστεί λάθως το parityBit και αντι για 1 το stopBit είναι 0. Στην περίπτωση αυυτή ενεργοποιώ τα σήματα Rx_PERROR και Rx_FERROR. Σε αντίθετη περίπτωση ενεργοποιείται το σήμα Rx_VALID

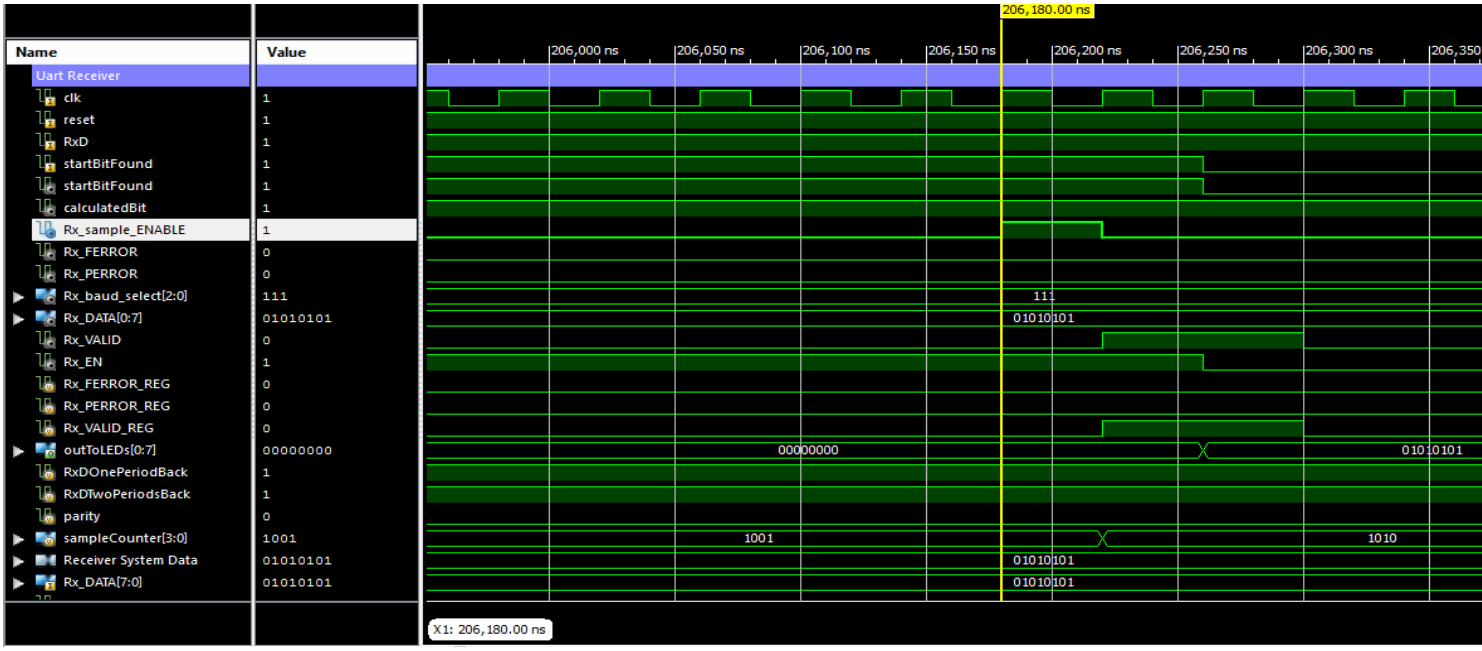
- Η διαδικασία λήψης ξεκινά με την αποστολή του startBit από τον δέκτη στο σύστημα. Μόλις ανιχνευτεί, ενεργοποιείται το σήμα Rx_EN και ξεκινά η λειτουργία του baud_controller. Το Rx_EN

παραμένει 1 όσο γίνεται δειγματοληψία. Μόλις τελειώσει η δειγματοληψία του stopBit, το Rx_EN γίνεται 0 και κάνουμε reset τον baud_controller

- Στο παρακάτω σιγμιότυπο φαίνεται η ανίχνευση του startBit από τον δέκτη την χρονική στιγμή 17,540ns. Στον αμέσως επόμενο κύκλο ενεργοποιείται το Rx_EN και ξεκινά η δειγματοληψία



- Ιδιαίτερη περίπτωση είναι η αποστολή του stopBit και απενεργοποίηση του δέκτη



- Την χρονική στιγμή 206,180ns γίνεται η δειγματοληψία του stopBit 1, ενώ τον αμέσως επόμενο κύκλο ενεργοποιείται το σήμα Rx_Valid για αποστολή δεδομένων στην μνήμη του συστήματος. Οι 2 κύκλοι αντί για έναν που είναι ενεργοποιημένο το σήμα δεν επηρεάζει την λειτουργία του δέκτη, διότι προηγείται η απενεργοποίηση του δέκτη πρώτα(Rx_VALID). Την στιγμή 206,250ns που σταματά την δειγματοληψία ο δέκτης , γίνεται η αποστολή της 8αδας bit στο σύστημα

ΜΕΡΟΣ 4: Σύνδεση uart_receiver με τον transmitter

- Η σύνδεση των 2 κυκλωμάτων γίνεται καλώντας απτον transmitter ένα instance του receiver με εισόδους το RxD. Για την προσομοίωση μόνο περνάω και τα clock, reset απτόν transmitter στον receiver

- Τα συστήμα πλαισίου δοκιμής στην εκφωνηση της άσκησης είναι τα Συστήματα των transmitter και receiver. Παρακάτω φαίνεται η αποστολή λήψη μυνημάτων με το Tx_DATA[0:7] να είναι η τιμή του transmitter, ενώ ο transmitter το λαμβάνει αυτό 2 κύκλους μετά, στο outToLEDs[0:7]


Name
Value
clk
1
reset
1
Tx_BUSY
1
Tx_sample_ENABLE
0
Tx_baud_select[2:0]
111
Tx_DATA[0:7]
10101010
Tx_EN
1
Tx_WR
0
TxD
1
Tx_BUSY_REG
1
sampleCounter[3:0]
1010
parity
0
receiver
clk
1
reset
1
outToLEDs[0:7]
startBitFound
Rx_FERROR
0
Rx_PERROR
0
Rx_VALID
0
Rx_baud_select[2:0]
111
Rx_EN
0
dataCounter[1:0]
11
/myTestBench/uart_Tx/uart_receiver0/RSD/reset
100.00 ns
0 ns
200,000 ns
400,000 ns
600,000 ns
800,000 ns
1,000,000 ns
1,200,000 ns
1,400,000 ns
X1: 100.00 ns