Υλοποίηση Μονάδας Γενικού Ασύγχρονου Δέκτη Αποστολέα (UART)

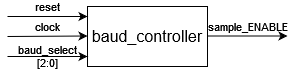
Εργαστήριο Ψηφιακών Συστημάτων (2023-24)

Ιωάννης Αθανασιάδης 03491

15/11/2023

# Μέρος Α – Ελεγκτής Baud Rate

Το *module* αυτό το μοιράζονται ο *Δέκτης* και *Αποστολέας* και ο στόχος του είναι να παρέχει στο κύκλωμα *τον κατάλληλο* ***παλμό δειγματοληψίας*** ανάλογα με το επιλεγμένο ***Baud Rate***.



Δεδομένου ότι έχουμε συχνότητα ρολογιού στα *100 MHz* πρέπει να βρούμε τις μέγιστες τιμές του μετρητή μας για να πετύχουμε το σωστό ***Baud Rate***.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BAUD\_SEL | Baud Rate | TBaud Rate / 16 (ns) | Τιμή Counter | Σχετικό Σφάλμα |
| 000 | 300 | 208333.3 | 20833 | 20 ns |
| 001 | 1200 | 52083.3 | 5208 | 20 ns |
| 010 | 4800 | 13020.8 | 1302 | 13 ns |
| 011 | 9600 | 6510.4 | 651 | 6 ns |
| 100 | 19200 | 3255.2 | 325 | 83 ns |
| 101 | 38400 | 1627.6 | 163 | 39 ns |
| 110 | 57600 | 1085.1 | 108 | 81 ns |
| 111 | 115200 | 542.5 | 54 | 40 ns |

Παρατηρούμε ότι η μέγιστη τιμή του ***counter*** είναι το 20833 επομένως για να βρούμε το μέγεθος του μετρητή σε bit βρίσκουμε το *άνω φράγμα* του *log220833,* δηλαδή ***15 bit.***

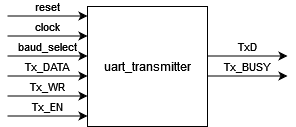
*Σημείωση:* Το *σχετικό σφάλμα* αν και σημαντικό, υπό τις ιδανικές καταστάσεις του εργαστηρίου δεν δημιουργεί μεγάλο πρόβλημα γιατί ο *Αποστολέας* και *Δέκτης* έχουν το ίδιο σφάλμα στην παραγωγή του ***Baud Rate***.

# Μέρος Β – Υλοποίηση UART Αποστολέα (Transmitter)

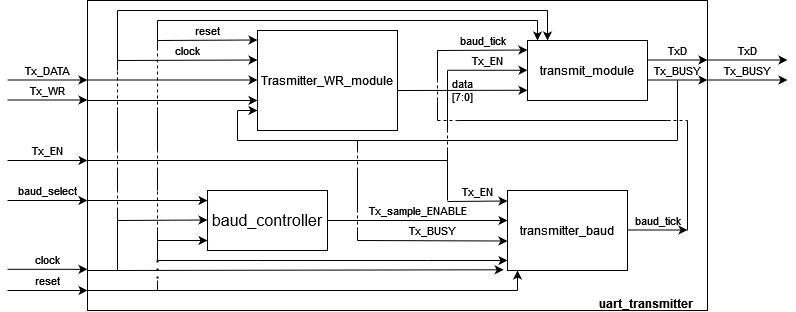
Το *top-level* *module* είναι το *uart\_transmitter:*

## Μονάδα *uart\_transmitter*

Χρησιμοποιούμε αυτή την μονάδα για ***top-level*** *module* και αποτελεί ότι έχει να κάνει με την ***μετάδοση*** δεδομένων μέσω *UART*.

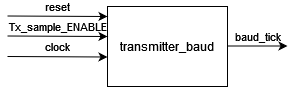


Η μονάδα αποτελείται από το module για την εγγραφή του μηνύματος, ***transmitter\_WR\_module***, τις μονάδες για την παραγωγή του αργού και γρήγορου baud rate, ***baud\_controller*** και ***transmitter\_baud*** αντίστοιχα, και τέλος την ***transmit\_module*** που αναλαμβάνει την αποστολή του μηνύματος.

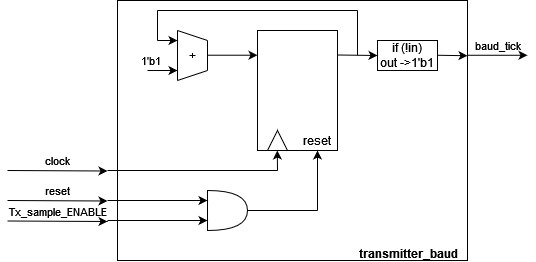


## Μονάδα *transmitter\_baud*

Το *module* αυτό χρησιμοποιείτε για να βρούμε το **σωστό** *baud rate* για τον *Αποστολέα*, δηλαδή για να το κάνουμε ***16 φορές*** *αργότερο* από την *έξοδο* του *baud\_controller*.

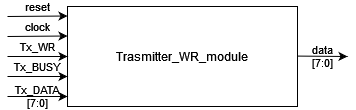


Πρακτικά αποτελείται από έναν *μετρητή* των ***4-bit*** που παράγει έναν θετικό **παλμό**, *με διάρκεια ενός κύκλου*.

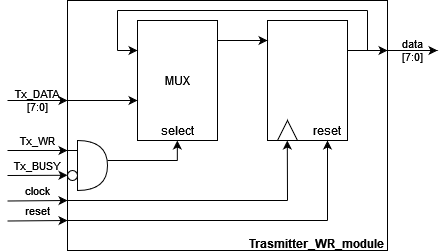


## Μονάδα *Trasmitter\_WR\_module*

Η μονάδα αυτή είναι υπεύθυνη για την *λήψη* των **δεδομένων** από το **σύστημα.**

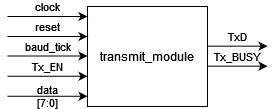


Εσωτερικά του *module* μεταφέρουμε το *Tx\_DATA* στο κομμάτι της μνήμης *data από το οποίο γίνεται η αποστολή του μηνύματος.* Για να το κάνουμε αυτό ελέγχουμε αν ο αποστολέας είναι σε *κατάσταση αποστολής* δεδομένων και αν το σήμα *Tx\_WR είναι σηκωμένο*.



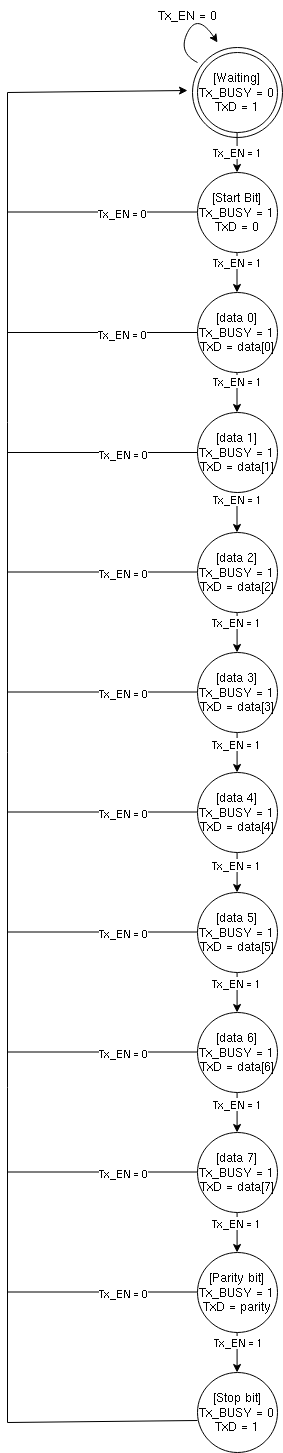
## Μονάδα *transmit\_module*

Αυτή η μονάδα αναλαμβάνει την **αποστολή δεδομένων** στην ενεργοποίηση του ***Tx\_EN.***



*Σημείωση:* Το ***Tx\_EN*** σημαίνει έναρξη της αποστολής μηνύματος, και ***όχι*** του *αποστολέα γενικότερα.* Η ***λήψη του μηνύματος*** Tx\_DATA από το σύστημα γίνεται με ***Tx\_EN=0* και *Tx\_WR=1*.**

*Εσωτερικά* αυτό το *module* περιέχει την ***FSM*** για την μετάδοση του ***data***. Παρακάτω ακολουθεί το **dataflow** της μηχανής καταστάσεων:



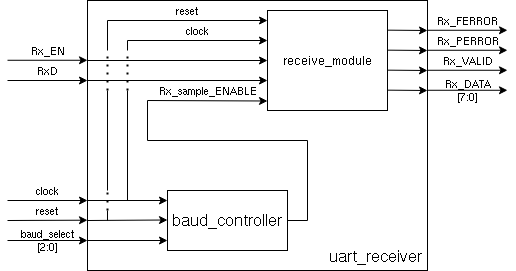
# Μέρος Γ – Υλοποίηση UART Δέκτη (Receiver)

Η top-level μονάδα αυτού του μέρους είναι η *uart\_receiver.*

## Μονάδα *uart\_receiver*

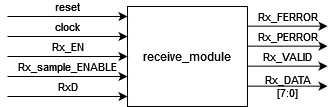


Η μονάδα αυτή εμφανίζει την **μηχανή καταστάσεων** για την λήψη των δεδομένων και το *module* ***baud\_controller*** για την δημιουργία του *baud rate*.



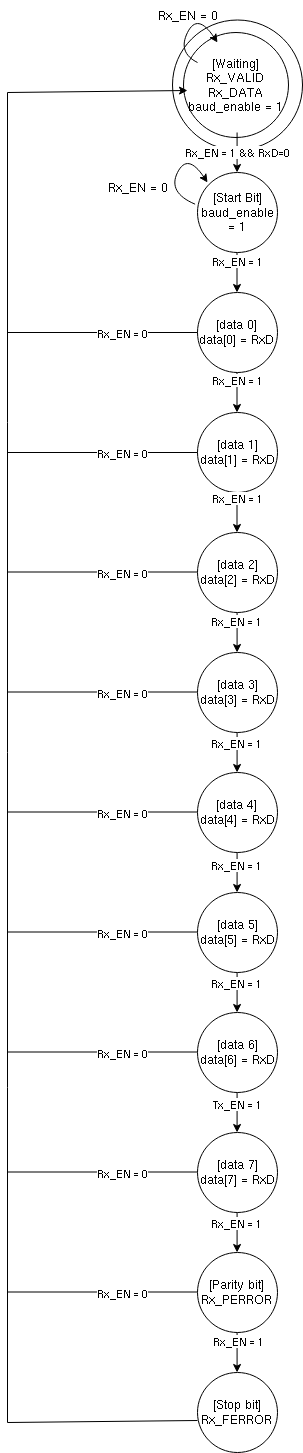
## Μονάδα *receive\_module*

Αυτή η μονάδα υλοποιεί την ***μηχανή καταστάσεων*** για την ανάγνωση του μηνύματος.



Η μονάδα πέρα από την *FSM* περιέχει την μονάδα ***receiver\_baud***, με σήμα ενεργοποίησης *baud\_enable*, η οποία παράγει το *baud rate* του *transmitter* με την σωστή *μετατόπιση* ώστε να γίνεται ανάγνωση στην μέση κάθε *bit*.

*Σημείωση:* Σε κάθε κατάσταση αναφέρονται τα **σήματα** που **αλλάζουν** *ενδεχομένως* τιμή. Για παράδειγμα το *Rx\_VALID* και *Rx\_DATA* στην *κατάσταση* ***waiting*** θα αλλάξουν τιμή *μόνο* αν έχουμε ***έγκυρη*** ανάγνωση μηνύματος.



# Μέρος Δ – Υλοποίηση UART Αποστολέα-Δέκτη (Transceiver)

Για το *τελευταίο* μέρος της εργασίας ολοκληρώθηκε επιπλέον και το ***προαιρετικό*** κομμάτι της στο οποίο δοκιμάζουμε την χρήση του *transceiver μ*ε *εμφάνιση* του μηνύματος που λαμβάνει ο δέκτης στην ***7-Segment οθόνη*** της *προηγούμενης* εργασίας.

## Μονάδα *uart\_system*

Αυτή η μονάδα αποτελεί το *top-level* του κυκλώματος που θα τρέξει στην ***FPGA***. Περιέχει κύκλωμα *συγχρονισμού* και *anti-bounce* για το κουμπί αποστολής νέου μηνύματος, κύκλωμα *συγχρονισμού* για το *reset button,* το κύκλωμα του *uart\_transceiver* και το κύκλωμα διαχείρισης του συστήματος, *system\_controller*.

## Μονάδα *uart\_transceiver*

Αυτή η μονάδα απλά εμφανίζει τις δύο μονάδες *uart\_transmitter* και *uart\_receiver* με σύνδεση *loop-back*, δηλαδή το *RxD* του ενός είναι το *TxD του άλλου*.

## Μονάδα *system\_controller*

Η μονάδα *system\_controller* είναι μία μηχανή καταστάσεων ως αντικαταστάτης της *initial* που συμβατικά έχει ένα *testbench,* η οποία όμως δεν είναι *συνθέσιμη*. Χρησιμοποιούμε 25-bit counter για να μεταβούμε στην επόμενη κατάσταση με καθυστέρηση σε ορισμένες περιπτώσεις.

Για κάθε μήνυμα υπάρχουν **3βασικές *καταστάσεις***:

1. Την ***write***, όπου γράφουμε μέσω του *Tx\_WR* το νέο μήνυμα που θέλουμε να στείλουμε,
2. Την ***sent***, όπου στέλνουμε το μήνυμα με την ακμή των σημάτων *Rx\_EN* και *Tx\_EN*,
3. Και την ***wait***, όπου περιμένει τον χρήστη να πατήσει το κουμπί επόμενου μηνύματος στο ενδιάμεσο του οποίου η οθόνη μας προβάλει την δεκαεξαδική αναπαράσταση του μηνύματος που στείλαμε.

Πιο αναλυτικά στέλνουμε τα μηνύματα ***0xAA***, ***0x55***, ***0xCC*** και ***0x89***:

