# Report II

Τσόγκας Παναγιώτης Νικόλαος - 3672

11/10/2024

## Σημείωση

Για το παρακάτω Report σε μορφή PDF, συνιστάται η ανάγνωση σε Dark Mode.

## Περίληψη

Η υλοποίηση του συστήματος UART πραγματοποιήθηκε μέσω της ανάπτυξης δύο μονάδων: του Transmitter και του Receiver. Για την εξασφάλιση της ακεραιότητας των δεδομένων, χρησιμοποιήθηκε ο έλεγχος χρονισμού με τη βοήθεια ενός Baud Controller, καθώς και η διαχείριση του σήματος Tx BUSY στον Transmitter και του σήματος Rx VALID στον Receiver, για την αποφυγή συγκρούσεων και την εγγύηση της σωστής μετάδοσης.

Η επαλήθευση του συστήματος πραγματοποιήθηκε με τη χρήση testbenches, τα οποία ελέγχουν την σωστή λειτουργία της σειριακής μετάδοσης, την ανίχνευση σφαλμάτων (π.χ. ισοτιμία και πλαισίωση), καθώς και την απόκριση του συστήματος σε χρονικά καθυστερημένες ή συγχρονισμένες ροές δεδομένων. Τα αποτελέσματα των πειραμάτων επιβεβαίωσαν την ορθότητα της υλοποίησης, δείχνοντας ότι το σύστημα μπορεί να διαχειριστεί αποτελεσματικά δεδομένα υψηλής ροής, ακόμη και όταν δίνονται νέοι χαρακτήρες αμέσως μετά την ολοκλήρωση μιας μετάδοσης.

Στην τελική φάση της ανάπτυξης, αγνοήθηκαν ορισμένα warnings σχετικά με τη ρύθμιση των BitStream constraints, καθώς δεν επηρεάζουν την τρέχουσα λειτουργικότητα του συστήματος.

## Εισαγωγή

Η παρούσα εργασία αφορά την υλοποίηση ενός συστήματος επικοινωνίας μέσω του πρωτοκόλλου UART (Universal Asynchronous Receiver Transmitter) με τη χρήση FPGA. Στο πλαίσιο αυτό, σχεδιάστηκαν και υλοποιήθηκαν δύο βασικές μονάδες: ο Transmitter (Αποστολέας) και ο Receiver (Δέκτης), οι οποίοι επικοινωνούν μεταξύ τους μέσω σειριακής μετάδοσης δεδομένων. Το σύστημα αυτό απαιτεί την κατάλληλη διαχείριση των χρονισμών, την ασύγχρονη επικοινωνία και την ανίχνευση σφαλμάτων, προκειμένου να διασφαλιστεί η ακεραιότητα των μεταδιδόμενων δεδομένων.

# A

## (0) Βοηθητική Επεξήγηση:

Στο πρωτόκολλο UART (Universal Asynchronous Receiver Transmitter), το οποίο χρησιμοποιείται ευρέως λόγω της απλότητας και της ευκολίας υλοποίησης, επιτρέπει την ασύγχρονη communication μεταξύ του Transmitter και του Receiver. Ο συγχρονισμός τους, παρότι δεν υπάρχει κοινό ρολόι, επιτυγχάνεται με τη χρήση ενός baud controller που καθορίζει τον ρυθμό μετάδοσης (Baud rate) και εξασφαλίζει ότι το UART τηρεί την κοινή ταχύτητα Baud (bits/sec).

Ο baud controller αποτελείται από έναν counter και έναν selector, οι οποίοι ορίζουν τις χρονικές στιγμές που αποστέλλονται τα σήματα συγχρονισμού μεταξύ των δύο μονάδων. Για να εξασφαλιστεί η ορθότητα των data, ο Receiver λειτουργεί σε 16πλάσια ταχύτητα από την προσυμφωνημένη Baud rate, δειγματοληπτώντας κάθε bit με x16 συχνότητα (λιγότερο στην συγκεκριμένη υλοποίηση). Αυτό επιτρέπει στον Receiver να εξετάζει το κάθε bit λεπτομερώς, εντοπίζοντας πιθανά λάθη συγχρονισμού ή διακυμάνσεις στην ταχύτητα.

Η υλοποίηση περιλαμβάνει δύο ανεξάρτητους baud controllers, έναν για τον Transmitter και έναν για τον Receiver, οι οποίοι λειτουργούν υπό τις ίδιες συνθήκες για κάθε διαδικασία μετάδοσης δεδομένων. Αυτός ο διπλός συγχρονισμός, με τον Receiver να δειγματοληπτεί πολλαπλάσια, διασφαλίζει την ακεραιότητα της communication και αποτρέπει απώλειες ή επαναλήψεις data.

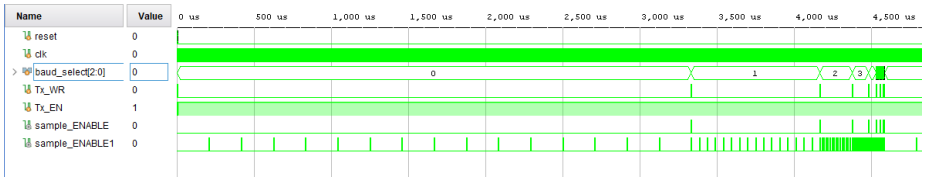
## (1) Υλοποίηση:

Για να βρούμε τις τιμές που αντιστοιχούν στον counter του Receiver ανάλογα με το ρολόι και το Baud Select, θα χρησιμοποιήσουμε την εξίσωση $T = \frac{1}{BaudRate \cdot 16}$. Είναι σημαντικό να προσέξουμε ότι ο τύπος αναφέρεται σε bits/sec, συνεπώς πρέπει να μετατρέψουμε τη μονάδα χρόνου sec σε μονάδες χρόνου της πλακέτας, δηλαδή σε clock cycles. Συνεπώς, ο τύπος παίρνει τη μορφή: $T = \frac{1}{BaudRate \cdot 16} \cdot 10^8$ Για τον Transmitter, πρέπει να προσέξουμε ότι για να βρούμε τους χτύπους ρολογιού που πρέπει να κάνει ο counter, πρέπει να χρησιμοποιήσουμε τον τύπο $(ReceiverCycles + 1) \cdot 16 - 1$ .Ένα απλό παράδειγμα επιβεβαίωσης είναι αν ο baud controller του Receiver χτυπάει ανά 4 κύκλους, κάτι που θα σήμαινε ότι ο Transmitter πρέπει να έχει κάνει $4 \cdot 16 = 64$ κύκλους, όμως ο counter ξεκινάει να μετράει από το 0, άρα οι κύκλοι τελικά θα είναι 3 και 63 αντίστοιχα. Με τη συγκεκριμένη μετατροπή, είναι σημαντικό να εξετάσουμε την απόκλιση που προκύπτει από τις πραγματικές τιμές, οι οποίες δεν μπορούν να χρησιμοποιηθούν ακριβώς λόγω της παρουσίας δεκαδικών ψηφίων. Για τον λόγο αυτό, εφαρμόζουμε στρογγυλοποίηση και λαμβάνουμε υπόψη τις αντίστοιχες τιμές σφαλμάτων που προκύπτουν από τη διαδικασία αυτή, οι οποίες εμφανίζονται παρακάτω:

| Baud Rate | Calculated Output | Rounded Output | Absolute Error | Relative Error |
|---|---|---|---|---|
| 300 | 20833.33 | 20833 | 0.33 | $1.60 \times 10^{-5}$ |
| 1200 | 5208.33 | 5208 | 0.33 | $6.40 \times 10^{-5}$ |
| 4800 | 1302.08 | 1302 | 0.08 | $6.40 \times 10^{-5}$ |
| 9600 | 651.04 | 651 | 0.04 | $6.40 \times 10^{-5}$ |
| 19200 | 325.52 | 326 | 0.48 | 0.00147 |
| 38400 | 162.76 | 163 | 0.24 | 0.00147 |
| 57600 | 108.51 | 109 | 0.49 | 0.00454 |
| 115200 | 54.25 | 54 | 0.25 | 0.00467 |

## (2) Επαλήθευση:

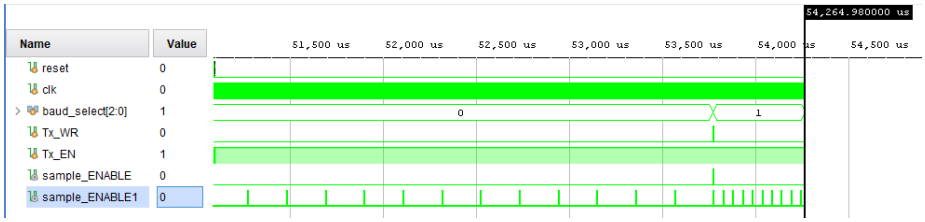
Για την επαλήθευση των controller, υπάρχει μια μονάδα testbench, η οποία ενεργοποιεί τα δύο controller ταυτόχρονα και ελέγχει αν το σήμα από αυτά φτάνει την ίδια στιγμή για κάθε BAUD_RATE. Αντίστοιχα, εκτυπώνει ένα μήνυμα για την επιτυχή ή λανθασμένη λειτουργία των μονάδων. Παρακάτω φαίνονται οι κυματομορφές από ένα τρέξιμο του testbench:




Σχήμα 1: Κυματομορφές controller_tb

## (3) Πείραμα/Τελική Υλοποίηση:

Ένα τελευταίο πείραμα σωστής υλοποίησης είναι να επιτρέψουμε στο testbench να αφήσει τα module ανοιχτά για αρκετή ώρα, έτσι ώστε να διαπιστώσουμε αν οι υπολογισμοί μας για τους χρόνους του transmitter είναι λάθος και αν με την πάροδο του χρόνου ο transmitter αποσυγχρονίζεται. Αυτό, όμως, δεν παρατηρήθηκε κατά το testing. Για την τελική υλοποίηση, η μονάδα πλέον χρησιμοποιεί ένα σήμα εισόδου Enable_controller, το οποίο κρατάει τη μονάδα σε κατάσταση reset όσο δεν χρησιμοποιείται ή όσο το module που την χρησιμοποιεί είναι απενεργοποιημένο.




Σχήμα 2: Υπο συνεχόμενη εκτέλεση

## (4) Συμπεράσματα-Παρατηρήσεις:

Η υλοποίηση των baud controllers επιβεβαιώθηκε μέσω των πειραμάτων και η ακεραιότητα των δεδομένων διασφαλίστηκε, όπως φαίνεται από τα αποτελέσματα του testbench. Η διαδικασία του υπολογισμού των χρονικών κύκλων για την Baud rate είναι ακριβής και οι στρογγυλοποιήσεις που πραγματοποιούνται για την επίλυση των δεκαδικών σφαλμάτων δεν προκαλούν σημαντική απόκλιση από τις πραγματικές τιμές. Στην τελικη υλοποίηση αγνοηθηκαν τα παρακατω Warning :
[Constraints 18-5210] No constraints selected for write.
Το παραπανω Warning αγγνοείται καθώς αναφέρεται σε παραγωγή BitStream η οποία δεν μας ενδιαφέρει ακόμα σε αυτο το κομμάτι της εργασίας
[XSIM 43-4099] .../Source Code/Controller src/baud_controller_t.v" Line 3. Module baud_controller_t doesn't have a timescale but at least one module in design has a timescale.
Το παραπανω Warning αγγνοείται καθώς το timescale θετετε αναθέτεται στο testbench.

# B

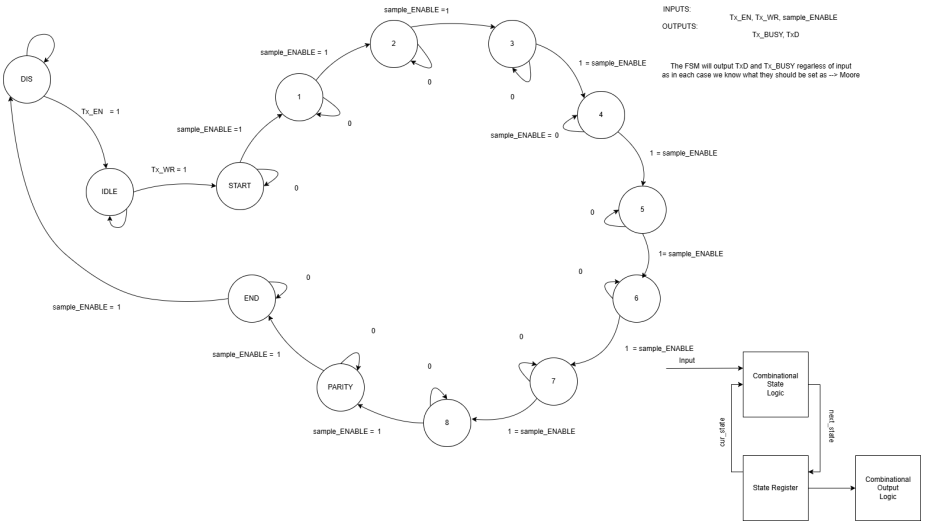
## (0) Βοηθητική Επεξήγηση:

Ο αποστολέας (Transmitter) στο πρωτόκολλο UART είναι υπεύθυνος για τη σειριακή μετάδοση δεδομένων από τη μονάδα αποστολής στον παραλήπτη. Στην υλοποίησή του, ο (Transmitter) ακολουθεί το προσυμφωνημένο Baud rate για να εξασφαλίσει τη σωστή χρονική αλληλουχία των δεδομένων. Η λειτουργία του συγχρονίζεται μέσω του ελεγκτή Baud rate, ο οποίος ρυθμίζει την ταχύτητα αποστολής, και του μετρητή δεδομένων, ο οποίος επιβλέπει τη σειρά μετάδοσης των bits.

Η αποστολή ξεκινά με το bit εκκίνησης (Start bit), ακολουθεί τα δεδομένα, και ολοκληρώνει με το bit τερματισμού (Stop bit). Για τη σωστή λειτουργία του, απαιτείται η χρήση του σήματος Tx_BUSY, το οποίο υποδεικνύει ότι η μονάδα είναι σε λειτουργία και δεν επιτρέπει νέες αποστολές μέχρι την ολοκλήρωση της τρέχουσας. Αυτή η διαδικασία εξασφαλίζει τη συνεπή και αξιόπιστη μετάδοση των δεδομένων μέσω του σειριακού καναλιού. Επιπλέον, για τον εντοπισμό σφαλμάτων υπολογίζεται το Parity bit, το οποίο αποστέλλεται μαζί με το μήνυμα στον Receiver, προκειμένου να αποφασιστεί αν η μετάδοση (transmission) έχει πραγματοποιηθεί σωστά. Συνεπώς το πακέτο που λαμβάνει ο Receiver μοιαζει καπως ετσι : {Start Bit, Data[0], ..., Data[7], Parity Bit, Stop Bit}.

## (1) Υλοποίηση:

Για την υλοποίηση του Transmitter, θα χρησιμοποιήσουμε έναν Finite State Machine (FSM), ο οποίος διαχειρίζεται τα διάφορα στάδια από τα οποία περνάει ο Transmitter για να στείλει επιτυχώς ένα σύμβολο σε έναν Receiver. Με μία αφηρημένη προσέγγιση, τα στάδια του FSM θα περιλαμβάνουν τα εξής: DISABLED, IDLE, και TRANSMITTING. Από αυτά, το τελευταίο, TRANSMITTING, θα



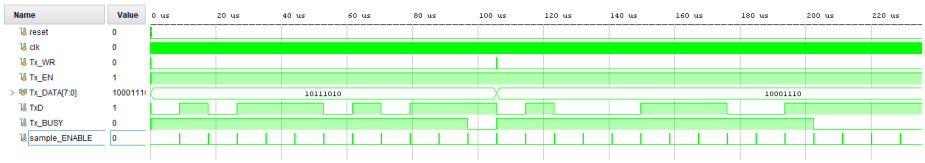

Σχήμα 3: Transmitter FSM

χωριστεί σε επιμέρους υποστάδια, ανάλογα με το σημείο στο οποίο βρισκόμαστε μέσα στην αλληλουχία των δεδομένων {Start Bit, Data[0], ..., Data[7], Parity Bit, Stop Bit}. Σε κάθε στάδιο του FSM, η πληροφορία θα αποστέλλεται κατάλληλα μέσω του σειριακού καναλιού TxD, ενώ το σήμα Tx_BUSY θα ενεργοποιείται για τη σωστή διαχείριση της μονάδας από κάποιο υψηλότερο module. Συγκεκριμένα, όταν το Tx_BUSY είναι 1, ο χειριστής του Transmitter θα πρέπει να φροντίζει να μην δεχτεί νέο σύμβολο, καθώς η μετάδοση του προηγούμενου δεν έχει ολοκληρωθεί.

Επιπλέον, χρησιμοποιείται ο Baud Controller του Transmitter για τον συγχρονισμό, όπως αναφέρθηκε προηγουμένως, ενώ στην είσοδό του Enable_controller εφαρμόζονται τα σήματα Tx_WR και Tx_EN με λογική AND (Tx_WR & Tx_EN).

## (2) Επαλήθευση:

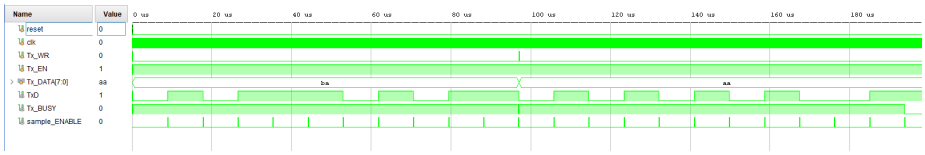
Για την επαλήθευση του Transmitter, έχει αναπτυχθεί μία μονάδα testbench που ελέγχει τη λειτουργία του. Συγκεκριμένα, η μονάδα επαληθεύει τη σωστή λειτουργία του σήματος Tx_BUSY και παρέχει στον Transmitter ένα νέο σύμβολο μόνο αφού έχει ολοκληρωθεί η μετάδοση του προηγούμενου. Επιπλέον, εκτυπώνει στο τερματικό τις τιμές που καταχωρούνται στο σειριακό κανάλι από τον Transmitter, οι οποίες θα πρέπει να ταυτίζονται με τα Data που έχουν δοθεί ως είσοδο.




Σχήμα 4: Κυματομορφές Post-timing transmitter testbench

## (3) Πείραμα/Τελική Υλοποίηση:

Ένα τελευταίο πείραμα για την επιβεβαίωση της ορθής υλοποίησης είναι να ξεκινήσει ένα νέο Transmission αμέσως μετά την ολοκλήρωση του προηγούμενου. Σε μια σωστή υλοποίηση, θα αναμένουμε ότι ο Transmitter θα λειτουργεί απρόσκοπτα ακόμα και σε αυτή την περίπτωση, παρόλο που είναι σπάνιο να δοθεί ένα νέο σύμβολο τόσο σύντομα μετά το προηγούμενο.




Σχήμα 5: Κυματομορφές υποπερίπτωσης Post-timing transmitter testbench

## (4) Συμπεράσματα-Παρατηρήσεις:

Η σχεδίαση και υλοποίηση του Transmitter στο πρωτόκολλο UART πέτυχε τη σωστή σειριακή μετάδοση των δεδομένων, όπως επαληθεύτηκε μέσω των πειραμάτων και του testbench. Η λειτουργικότητα του σήματος Tx_BUSY διασφαλίζει ότι ο αποστολέας δεν δέχεται νέο σύμβολο μέχρι την ολοκλήρωση της τρέχουσας μετάδοσης, αποτρέποντας συγκρούσεις δεδομένων.

Η ακρίβεια των χρονικών καθυστερήσεων επιτεύχθηκε μέσω της χρήσης του Baud Controller, ο οποίος παρέχει τον κατάλληλο συγχρονισμό.

Παρατηρήθηκε επίσης ότι η υλοποίηση λειτουργεί αποτελεσματικά ακόμα και όταν δίνεται νέο σύμβολο αμέσως μετά από την ολοκλήρωση του προηγούμενου Transmission. Αυτό αποδεικνύει την ανθεκτικότητα του συστήματος σε σενάρια υψηλής ροής δεδομένων.

Στην τελική έκδοση, αγνοήθηκαν τα εξής Warnings :
[Constraints 18-5210] No constraints selected for write.
Το παραπάνω Warning αγνοήθηκε καθώς σχετίζεται με ρυθμίσεις παραγωγής BitStream που δεν έχουν σημασία σε αυτό το στάδιο της ανάπτυξης.

# C

## (0) Βοηθητική Επεξήγηση:

Στο πρωτόκολλο UART, ο Δέκτης (Receiver) είναι υπεύθυνος για τη λήψη και αποκωδικοποίηση των σειριακών δεδομένων που αποστέλλονται από τον Αποστολέα. Όπως και στον Transmitter, ο Receiver συγχρονίζεται με τη χρήση ενός Baud Controller που ρυθμίζει τη συχνότητα δειγματοληψίας, επιτρέποντας την ακρίβεια στην ανάγνωση των bits. Ειδικότερα, ο Receiver λειτουργεί σε 16πλάσια συχνότητα από το προσυμφωνημένο Baud rate, διενεργώντας πολλαπλάσια δειγματοληψία σε κάθε βιτ για αυξημένη αξιοπιστία.

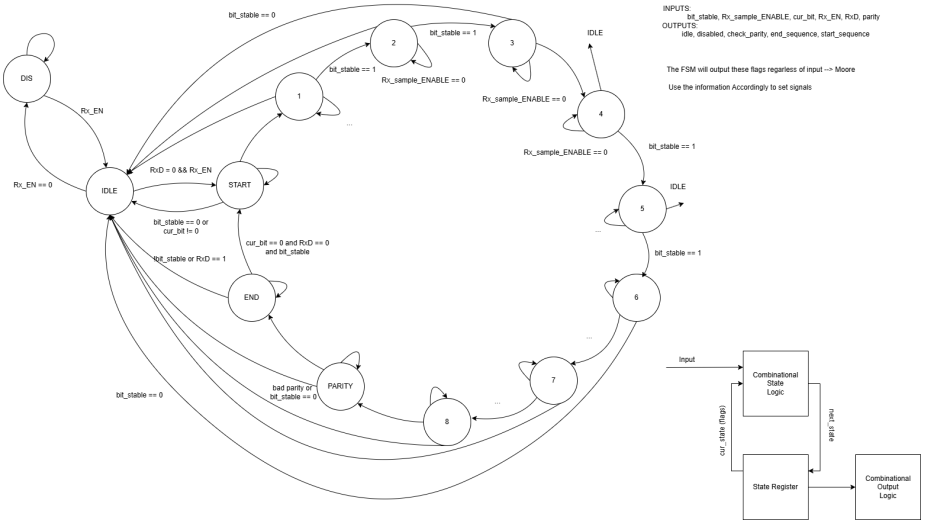
Η διαδικασία λήψης ξεκινά με την ανίχνευση του Start bit, το οποίο σηματοδοτεί την έναρξη μιας νέας επικοινωνίας. Κατά τη διάρκεια της λήψης, ο Receiver ελέγχει την ισοτιμία (παράγοντας Parity) και τη σωστή πλαισίωση (Framing) των δεδομένων. Εάν διαπιστωθεί κάποιο σφάλμα, το αντίστοιχο σήμα λάθους, Rx_PERROR για την ισοτιμία και Rx_FERROR για τη σωστή πλαισίωση, τίθεται σε υψηλή λογική τιμή 1.

Κατα την διάρκεια της επικοινωνίας τα δεδομένα μεταφέρονται στο Rx_DATA[7:0] και ενεργοποιείται το σήμα Rx_VALID, υποδεικνύοντας ότι τα δεδομένα είναι διαθέσιμα προς ανάγνωση από το σύστημα και τα δεδομένα έχουν ληφθεί σωστά. Τα σήματα Rx_DATA[7:0] και Rx_VALID παραμένουν σταθερά μέχρι να ανιχνευθεί νέο Start bit, διασφαλίζοντας ότι η πληροφορία παραμένει έγκυρη έως ότου ξεκινήσει νέα επικοινωνία από τον Αποστολέα.

## (1) Υλοποίηση:

Για την υλοποίηση του Receiver, όπως και στον Transmitter, χρησιμοποιείται ένας Finite State Machine (FSM), ο οποίος αποτελείται από τα στάδια DISABLED, IDLE και RECEIVING. Το τελευταίο στάδιο χωρίζεται σε επιμέρους υποστάδια που αντιστοιχούν στα επιμέρους δεδομένα. Κατά τη διάρκεια κάθε σταδίου του FSM, ο Receiver δειγματοληπτεί διαφορετικά σημεία κάθε bit για την εκτέλεση διαφόρων λειτουργιών. Η διαδικασία ξεκινά με την αποθήκευση της πληροφορίας από το σειριακό κανάλι TxD στο BUS εξόδου Rx_DATA, το οποίο λειτουργεί ως shift register για τη μεταφορά των δεδομένων.

Επιπλέον, γίνεται ανίχνευση πιθανών λαθών πλαισίωσης μέσω του ελέγχου διαδοχικών τιμών της εισόδου, από τις οποίες διατηρείται η μεσαία τιμή για την αποθήκευση στο Rx_DATA. Σημαντικό είναι να σημειωθεί ότι τα δύο υποσυστήματα, Transmitter και Receiver, συχνά λειτουργούν υπό διαφορετικά ρολόγια, γεγονός που δημιουργεί το κλασικό πρόβλημα του συγχρονισμού της ασύγχρονης εισόδου




Σχήμα 6: Receiver FSM

με το ρολόι του Receiver, το οποίο αποτελεί πρόβλημα στις Mealy FSM. Για την επίλυση αυτού, χρησιμοποιείται ένα synchronizer module επιπλέον του Receiver Baud Controller, εξασφαλίζοντας τον ορθό συγχρονισμό της μονάδας.

Για την υλοποίηση των σημάτων επιτυχίας ή σφάλματος, χρησιμοποιούνται καταχωρητές, καθώς η λειτουργία τους απαιτεί μνήμη. Εναλλακτικά, τα σήματα αυτά θα μπορούσαν να θεωρηθούν ως επιπλέον καταστάσεις στο FSM, εμπλουτίζοντας περαιτέρω την αρχιτεκτονική του.

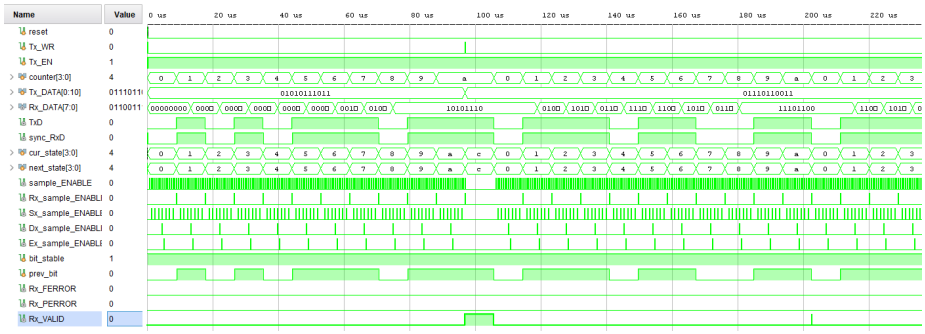
## (2) Επαλήθευση:

Για την επαλήθευση του Transmitter, έχει αναπτυχθεί μία μονάδα testbench που ελέγχει τη λειτουργία του μέσω του συνδυασμού διαφόρων tasks. Στο testbench αυτό, προσομοιώνουμε τη λειτουργία του Transmitter με τη βοήθεια του αντίστοιχου Baud Controller, χρησιμοποιώντας έναν for loop που αποστέλλει κάθε φορά το επόμενο bit από μία ακολουθία, η οποία καθορίζεται από τον χρήστη.

Η μονάδα πρέπει να λειτουργεί σωστά, υπό την προϋπόθεση ότι ο συγχρονισμός έχει επιτευχθεί σύμφωνα με τα μηνύματα του testbench και ότι το περιεχόμενο του καταχωρητή Rx_DATA είναι ταυτόσημο με το Tx_DATA. Επιπλέον, το σήμα του Receiver πρέπει να παραμένει σταθερό μέχρι να γίνει επαναφορά (reset) ή να ληφθεί νέο Start Bit.

Για την προσομοίωση της κατάστασης σφάλματος πλαισίωσης, αρκεί να αποσυγχρονίσουμε το Baud_Rate του Baud Controller που ελέγχει το for loop. Επίσης, για την προσομοίωση σφάλματος ισοτιμίας, αρκεί να χρησιμοποιήσουμε μία

σειρά bit με λανθασμένη τιμή στο parity bit, ώστε να ανιχνευθεί το σφάλμα από τον Receiver.

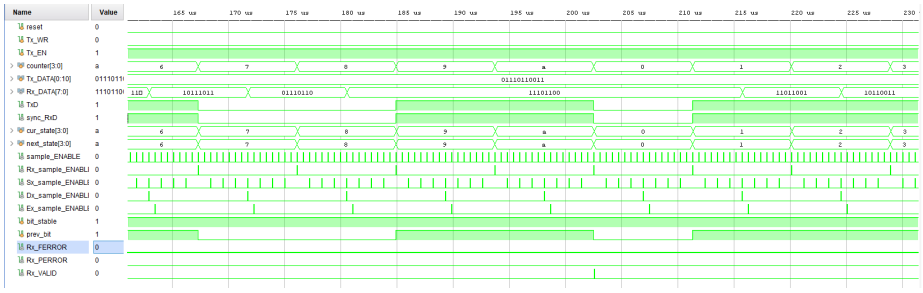



Σχήμα 7: Κυματομορφές Receiver Testbench

## (3) Πείραμα/Τελική Υλοποίηση:

Ένα τελευταίο πείραμα για την επιβεβαίωση της ορθής υλοποίησης είναι η εκκίνηση μιας νέας μετάδοσης (Transmission) αμέσως μετά την ολοκλήρωση της προηγούμενης, με σκοπό ο Receiver να ξεκινήσει ξανά την αποθήκευση πληροφοριών σχεδόν ακαριαία. Το σενάριο αυτό εξετάστηκε παραπάνω ως πιθανό και για τον Transmitter.

Στην περίπτωση αυτή, θα ήταν ιδανικό να αποθηκεύσουμε το μήνυμα σε έναν buffer, σε περίπτωση που δεν προλάβει να αξιοποιηθεί από τον χρήστη του Receiver. Επιπλέον, θα μπορούσε να υλοποιηθεί ένα σήμα που να ειδοποιεί ότι υπάρχει μήνυμα στη μνήμη ή κάποιο σήμα παράλειψης (overflow), κάτι που όμως είναι εκτός του σκοπού αυτής της εργασίας. Στις κυματομορφές, το παραπάνω θα αποτυπωθεί με έναν μικρό παλμό (spike) στο Rx_VALID, ο οποίος θα σηματοδοτεί την άμεση διαθεσιμότητα των δεδομένων.




Σχήμα 8: Κυματομορφές υποπερίπτωσης Receiver Testbench

### (4) Συμπεράσματα-Παρατηρήσεις:

Η υλοποίηση του Receiver στο πρωτόκολλο UART ολοκληρώθηκε επιτυχώς, εξασφαλίζοντας την ορθή λήψη και αποκωδικοποίηση των σειριακών δεδομένων από τον Transmitter. Μέσω της χρήσης ενός Finite State Machine (FSM) με στάδια DISABLED, IDLE και RECEIVING και τη συνδρομή του Baud Controller, ο Receiver συγχρονίζεται και δειγματοληπτεί αξιόπιστα τα δεδομένα. Το σήμα Rx_VALID παραμένει σταθερό μέχρι την έναρξη νέας επικοινωνίας, εξασφαλίζοντας την εγκυρότητα των δεδομένων.

Το testbench επαλήθευσης περιέλαβε ελέγχους συγχρονισμού και ανίχνευσης σφαλμάτων ισοτιμίας (Rx_PERROR) και πλαισίωσης (Rx_FERROR), επιβεβαιώνοντας τη σταθερότητα και αξιοπιστία του Rx_DATA και την ακεραιότητα των δεδομένων. Ένα πείραμα έδειξε ότι ο Receiver μπορεί να ξεκινήσει νέα λήψη αμέσως μετά την ολοκλήρωση της προηγούμενης, χωρίς απώλεια δεδομένων, ενώ μια προσθήκη buffer θα μπορούσε να καλύψει τη διαχείριση μηνυμάτων σε περιπτώσεις ταχείας ροής.

Στην τελική έκδοση, αγνοήθηκαν τα εξής Warnings :
[Constraints 18-5210] No constraints selected for write.
Το παραπάνω Warning αγνοήθηκε καθώς σχετίζεται με ρυθμίσεις παραγωγής BitStream που δεν έχουν σημασία σε αυτό το στάδιο της ανάπτυξης.

## D.1

### (0) Βοηθητική Επεξήγηση:

Ο UART ελεγκτής χρησιμοποιεί τον Transmitter και τον Receiver για την ομαλή μετάδοση και λήψη δεδομένων, ενσωματώνοντας επιπρόσθετη λογική για την αποτροπή εισερχόμενων μηνυμάτων όταν ο Transmitter είναι απασχολημένος, καθώς και για την ασφάλιση της επικοινωνίας σε περίπτωση εμφάνισης σφάλματος.

### (1) Υλοποίηση:

Για την υλοποίηση του UART χρησιμοποιείται ένας απλός Finite State Machine (FSM) με στάδια ENABLED, FERROR και PERROR. Επιπλέον, ενσωματώνονται οι μονάδες Transmitter και Receiver για τη μετάδοση δεδομένων.

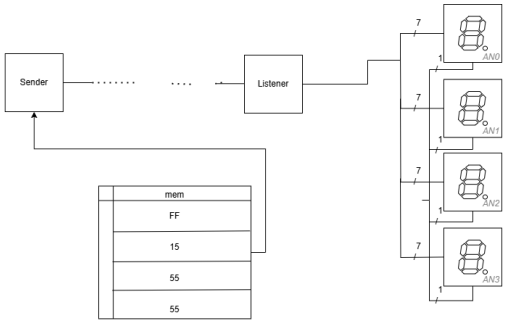
Ο FSM του UART παρακολουθεί τα σήματα Tx_BUSY, Rx_FERROR και Rx_PERROR, και αποφασίζει αν θα απενεργοποιήσει τις μονάδες και θα διακόψει την επικοινωνία ή θα επιτρέψει την εισαγωγή νέου συμβόλου στον Transmitter. Σε περίπτωση που ο Transmitter είναι απασχολημένος, το UART κρατά προσωρινά το εισερχόμενο μήνυμα έως ότου ολοκληρωθεί η τρέχουσα αποστολή και παραληφθεί ένα έγκυρο Tx_WR. Έτσι, ο χρήστης μπορεί να φορτώσει νέο μήνυμα στη μνήμη ακόμα και πριν τελειώσει η αποστολή του προηγούμενου.

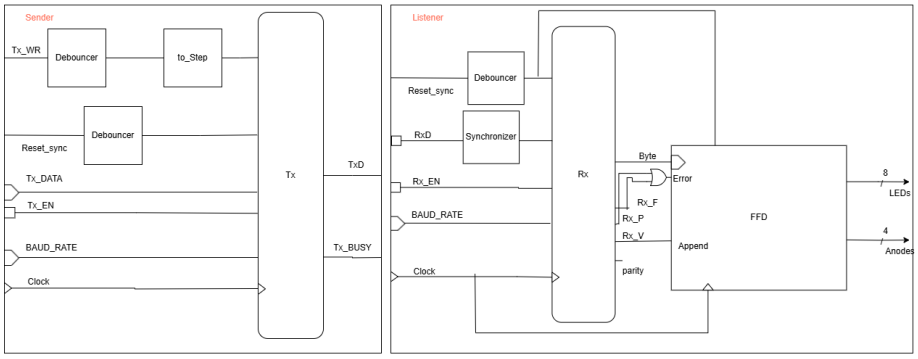
Επίσης, περιλαμβάνονται δύο μονάδες Debouncer για την εξάλειψη τυχόν παρεμβολών ή θορύβου στα σήματα που προέρχονται από την αλληλεπίδραση του χρήστη μέσω πλήκτρων.

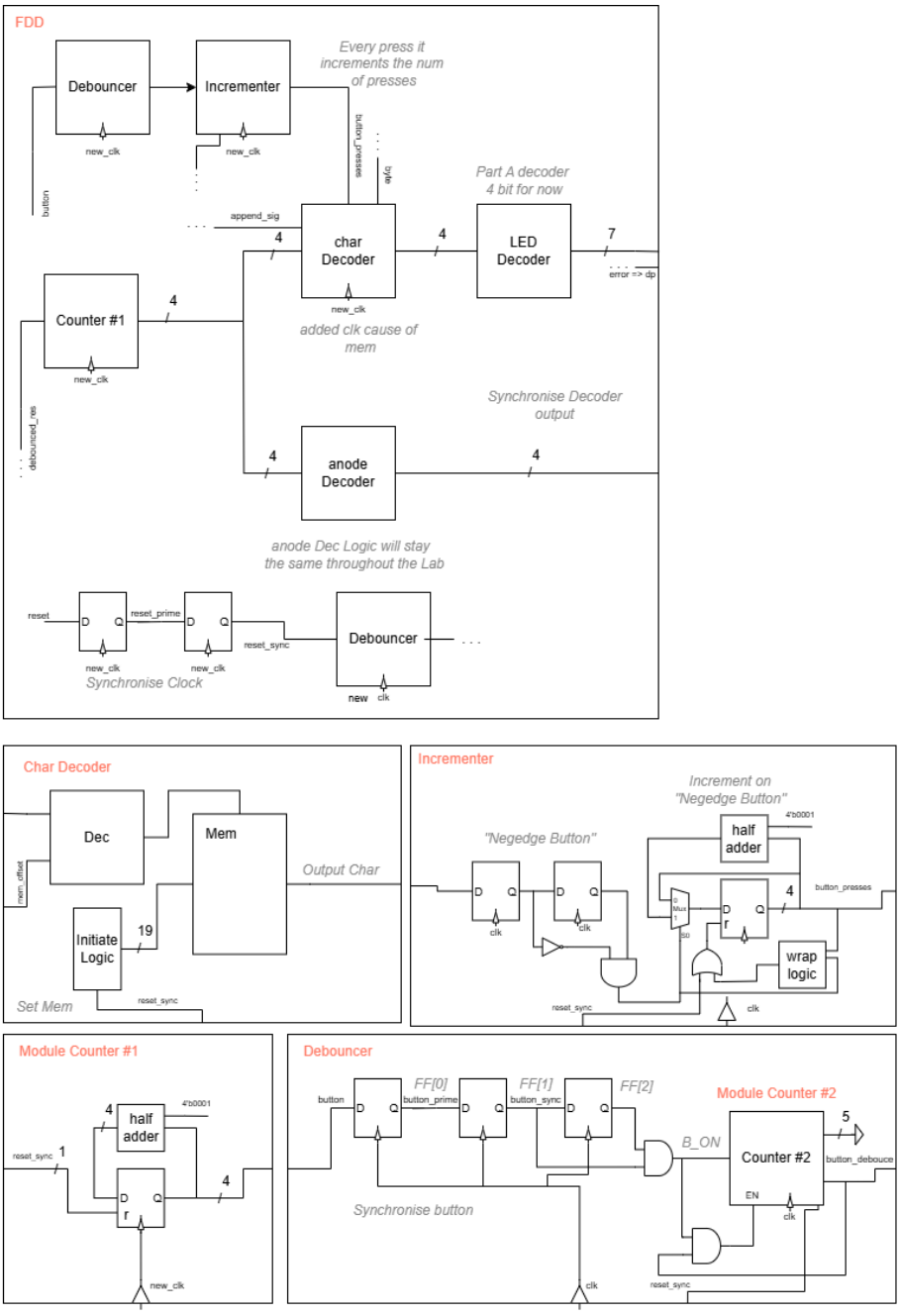
## (2) Επαλήθευση:

Για την επαλήθευση του Transmitter, έχει αναπτυχθεί ένα testbench που ελέγχει τη λειτουργία της μονάδας UART. Το testbench, όπως φαίνεται στα σχόλια, μπορεί να χρησιμοποιηθεί τόσο για τη λειτουργική προσομοίωση (functional simulation) της κανονικής λειτουργίας του UART, όσο και για την προσομοίωση χρονισμού (timing simulation), που επαληθεύει την απόκριση του συστήματος σε συνθήκες αποθήκευσης διαδοχικών μηνυμάτων.

Συγκεκριμένα, στο timing simulation οι καθυστερήσεις του σήματος Tx_BUSY και η ασύγχρονη εκτέλεση της εντολής while επιτρέπουν τη φόρτωση ενός νέου μηνύματος στον UART αμέσως μετά την ολοκλήρωση της πρώτης αποστολής. Με αυτή τη ρύθμιση, η αποθήκευση του δεύτερου μηνύματος γίνεται αμέσως μόλις ο Transmitter είναι διαθέσιμος, διασφαλίζοντας την ορθή διαχείριση της ροής δεδομένων. Για βέλτιστη απόδοση, μια μικρή καθυστέρηση μπορεί να προστεθεί πριν από την εκκίνηση του while loop στο testbench, ώστε να συγχρονιστεί ακριβώς με την ολοκλήρωση της πρώτης μετάδοσης. [η]



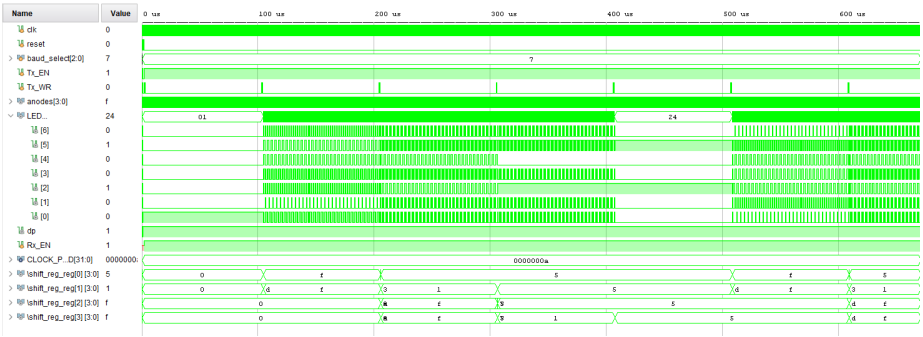






Σχήμα 9: UART-Optional Part

## (3) Πείραμα/Τελική Υλοποίηση:

Για την επαλήθευση της υλοποίησης, σχεδιάστηκαν δύο κυκλώματα για σύνθεση στην FPGA. Το πρώτο κύκλωμα οδηγεί τον Transmitter, παρέχοντας την καθορισμένη ακολουθία των τεσσάρων δεδομένων. Το δεύτερο κύκλωμα διαβάζει το ληφθέν δεδομένο από τον Receiver και, μέσω τροποποιημένου οδηγού 7-τμημάτων, απεικονίζει τα δύο δεκαεξαδικά ψηφία στην οθόνη. Η επιλογή του Baud Rate πραγματοποιείται μέσω των σταθερών διακοπτών της FPGA, επιτρέποντας την άμεση ρύθμιση της ταχύτητας επικοινωνίας.



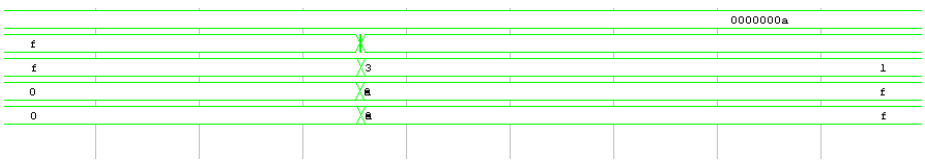

# D.2

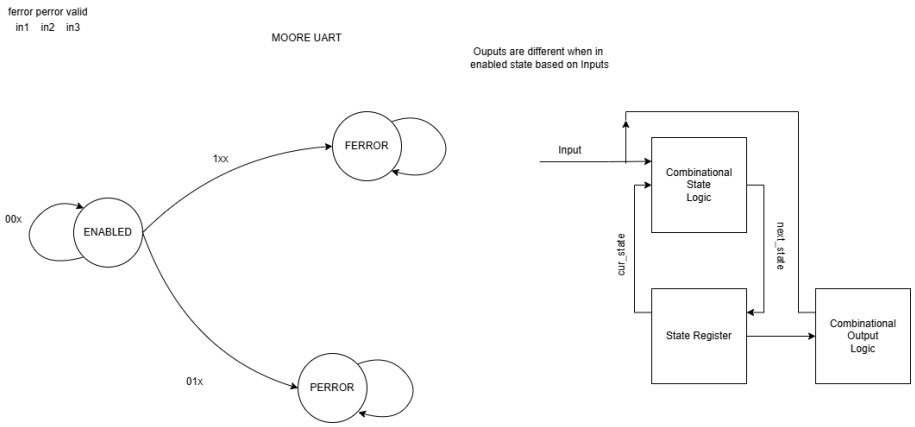
## (1) Υλοποίηση:

Για την υλοποίηση των δύο μονάδων Listener και Sender, αρχικά θα χρησιμοποιήσουμε τις ήδη υλοποιημένες μονάδες Transmitter και Receiver που αναπτύχθηκαν προηγουμένως. Επιπλέον, θα χρειαστούμε μερικούς Debouncer για να εξαλείψουμε τυχόν παρεμβολές από τα σήματα εισόδου και έναν Synchronizer, ο οποίος θα εξασφαλίσει σωστό συγχρονισμό μεταξύ διαφορετικών ρολογιών (όπως είχε χρησιμοποιηθεί στο προηγούμενο σχέδιο, όπου ο Synchronizer βρισκόταν εκτός του Receiver).

Επειδή τώρα θα χρησιμοποιούμε κουμπιά για την είσοδο, θα απαιτηθεί ένα νέο module που θα τροποποιεί το σήμα ενός κουμπιού σε έναν παλμό, διασφαλίζοντας τη σωστή λειτουργία των σημάτων εισόδου για τον Transmitter και τον Receiver.

Τέλος, θα χρησιμοποιήσουμε το Seven Segment Driver από την πρώτη εργασία, με κάποιες τροποποιήσεις. Κάποιες από τις βασικές αλλαγές περιλαμβάνουν



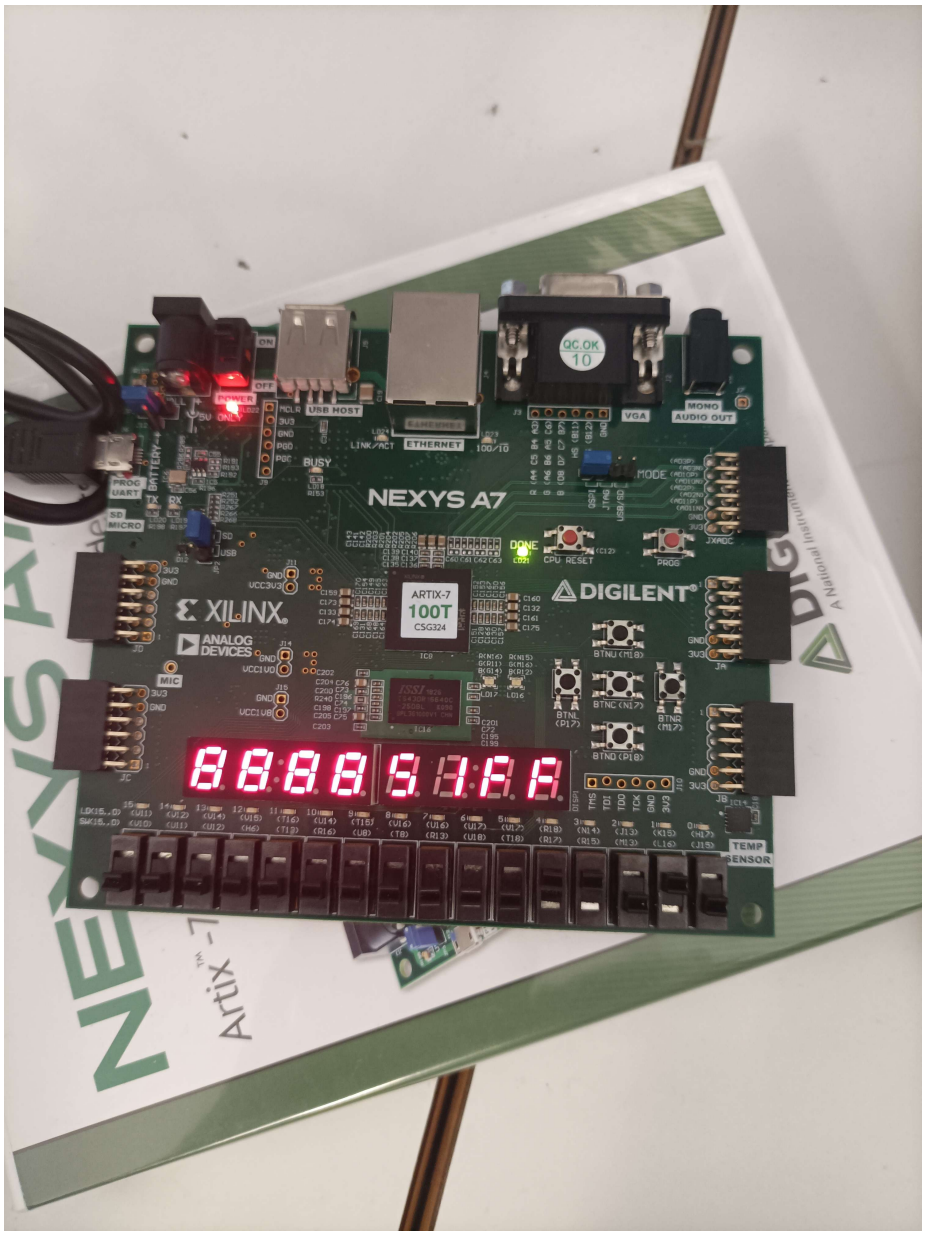



Σχήμα 10: UART FSM

τα νέα σήματα στη μονάδα Character Decoder, η οποία περιέχει τη μνήμη μηνυμάτων. Αυτή η μνήμη μπορεί τώρα να αλλάξει μέσω του σήματος append_byte, δίνοντας ένα νέο βψτε στην είσοδο της μονάδας. Αξιζει να σημειωθεί οτι στον κώδικα έχει χρησιμοποιηθεί λογική μαζί με το reset για ττην απενεργοποίηση των μονάδων, αυτό είναι κακή πρακτική και συνήθως έχουμε ένα extra σήμα Enable.

## (4) Συμπεράσματα-Παρατηρήσεις:

Αξίζει να σημειωθεί ότι λόγω της διαφοράς ρολογιού ανάμεσα στον οδηγό του 7-τμηματικού και τον υπόλοιπο κύκλωμα, παρατηρούνται μικρές καθυστερήσεις στην φόρτωση των νέων βψτε στους καταχωρητές της μνήμης του ἤαραχτερ Δεςοδερ. Ωστόσο, αυτές οι καθυστερήσεις είναι τόσο μικρές που δεν είναι ορατές με το μάτι και επομένως αγνοούνται στην αξιολόγηση της λειτουργίας του συστήματος. Όταν το πρόγραμμα δοκιμάστηκε στην πλακέτα φάνηκε να χρειάζεται 2 πατήματα κουμοπιού στην αρχή για να ξεκινήσει την λειτουργία του. Η εικόνα είναι mirrored καθώς δεν πρόλαβα να αλλάξω τα Anode στα ςοντραιντς στο εργαστήριο. Στην τελική έκδοση, αγνοήθηκαν τα εξής Warnings :
[Synth 8-350] instance 'MMCME2_BASE_inst' of module 'MMCME2_BASE' requires 18 connections, but only 5 given [SSD src/FourDigitDriver.v":22]
Το παραπάνω Warning αγνοήθηκε καθώς τα σήματα στα οποία αναφέρεται δεν μασ χρησιμεύουν. Το σήμα lock δεν χρησιμοποιείται στην υλοποίηση αυτη.




Σχήμα 11: ff15...