2η Εργαστηριακή αναφορά

Εργαστήριο Ψηφιακών Κυκλωμάτων

Καραμέλλιος Σωτήριος 2237

ΠΕΡΙΛΗΨΗ

Στην παρούσα αναφορά περιγράφονται τα εξής. Αρχικά, γίνεται μια σύντομη περιγραφή του συστήματος που θα υλοποιήσουμε καθώς και των δεδομένων που θα χρησιμοποιήσουμε για τον έλεγχό του. Έπειτα γίνεται μια πιο αναλυτική αναφορά στο κάθε module του συστήματος τόσο όσο προς την λειτουργία του αλλά και προς την επαλήθευση της λειτουργίας του.

ΕΙΣΑΓΩΓΗ

Ο στόχος της δεύτερης εργασίας ήταν η υλοποίηση ενός συστήματος σειριακής επικοινωνίας, το οποίο λειτουργεί χρησιμοποιόντας το πρωτόκολο uart  (Universal Asynchronous Receiver Transmitter - Γενικού Ασύγχρονου Δέκτη Αποστολέα). Το σύστημα αποτελείται από έναν αποστολέα, έναν δέκτη καθώς και μία μονάδα ελέγχου η οποία είναι υπεύθυνη μόνο για να κοιτάει τα σήματα ελέγχου και να αλλάζει τα δεδομένα στην είσοδο του αποστολέα. Η μονάδα ελέγχου παρέχει στον αποστολέα μια αλληλουχία τεσσάρψν διαφορετικών συμβόλων των 8-bit τα οποία είναι τα εξής(στο δυαδικό σύστημα):

10101010, 01010101, 11001100, 10001001

Πιο συγκεκριμένα και ο παραλήπτης και ο αποστολέας περιλαμβάνουνε εκτός από την λογική τους, και ένα module baud\_controller το οποίο δέχεται ένα σήμα baud\_select και βάση αυτού αποφασίζει και ορίζει την συχνότητα με βάση την οποία θα λειτουργήσουν ο αποστολέας και ο δέκτης. Η συχνότητα αυτή ορίζεται από το πρωτόκολλο uart στο 16 x baud\_rate.

Μέρος Α

case (baud\_select)

3'b000: begin

if (cycle\_counter == 14'b10100010110001) begin

sample\_ENABLE = 1;

cycle\_counter = 14'b00000000000000;

end

end

3'b001: begin

if (cycle\_counter == 14'b00101000101100) begin

sample\_ENABLE = 1;

cycle\_counter = 14'b00000000000000;

end

end

3'b010: begin

if (cycle\_counter == 14'b00001010001011) begin

sample\_ENABLE = 1;

cycle\_counter = 14'b00000000000000;

end

end

3'b011: begin

if (cycle\_counter == 14'b00000101000110) begin

sample\_ENABLE = 1;

cycle\_counter = 14'b00000000000000;

end

end

3'b100: begin

if (cycle\_counter == 14'b00000010100011) begin

sample\_ENABLE = 1;

cycle\_counter = 14'b00000000000000;

end

end

3'b101: begin

if (cycle\_counter == 14'b00000001010001) begin

sample\_ENABLE = 1;

cycle\_counter = 14'b00000000000000;

end

end

3'b110: begin

if (cycle\_counter == 14'b00000000110110) begin

sample\_ENABLE = 1;

cycle\_counter = 14'b00000000000000;

end

end

3'b111: begin

if (cycle\_counter == 14'b00000000011011) begin

sample\_ENABLE = 1;

cycle\_counter = 14'b00000000000000;

end

end

endcase

Όπως φαίνεται και παραπάνω για το module baud\_controller υλοποιήθηκε ένας απλός αποκωδικοποιητής για το σήμα baud\_select με βάση το οποίο ορίζεται η συχνότητα λειτουργείας του υπόλοιπου κυκλώματος. Η αποκωδικοποίηση και η επιλογή των τιμών έγινε με βάση τον παρακάτω πίνακα. Ο πίνακας με τον υπολογισμό των κύκλων έγινε βάση του τύπου  T\_sc = 1/(16 x Baud Rate)

|  |  |
| --- | --- |
| BAUD\_SEL | **Baud Rate** |
| 000 | 300 bits/sec |
| 001 | 1200 bits/sec |
| 010 | 4800 bits/sec |
| 011 | 9600 bits/sec |
| 100 | 19200 bits/sec |
| 101 | 38400 bits/sec |
| 110 | 57600 bits/sec |
| 111 | 115200 bits/sec |

|  |  |
| --- | --- |
| BAUD\_SEL | cycles |
| 000 | 10.416,666 |
| 001 | 2.604,166 |
| 010 | 651,042 |
| 011 | 325,52 |
| 100 | 162,76 |
| 101 | 81,38 |
| 110 | 54,253 |
| 111 | 27,126 |

Λόγω της ύπαρξης των δεκαδικών ψηφίων σαφώς υπάρχουν αποκλήσεις από τις ιδανικές τιμές οι οποίες περιλαμβάνονται στον παρακάτω πίνακα.

|  |  |
| --- | --- |
| BAUD\_SEL | **error** |
| 000 | 3,2\*10-5 |
| 001 | 6,4\*10-5 |
| 010 | 6,45\*10-5 |
| 011 | 1,5\*10-3 |
| 100 | 1,5\*10-3 |
| 101 | 4,7\*10-3 |
| 110 | 4,6\*10-3 |
| 111 | 4.6\*10-3 |

Ο δίπλα πίνακας περιέχει τα σχετικά σφάλματα που προκύπτουν από την στρογγυλοποίηση των δεκαδικών τιμών των κύκλων. Ο τύπος με βάση τον οποίον υπολογήσαμε τα σφάλματα είναι ο εξής: error = Δh/h0 όπου Δh = |h-h0| και h0 η πραγματική(δεκαδική) τιμή των κύκλων και h η στρογγθλοποιημένη(ακέραια) τιμή τους. Όπως βλέπουμε αριστερά τα σφάλματα μας είναι μικρά για τις μικρές ταχύτητες μετάδοσης και αυξάνονται στις υψηλές.

Για την επαλήθευση του κυκλώματος χρησιμοποιήθηκε ένα απλό testbench το οποίο αλλάζει το baud\_select και μετά μέσω των κυμματομορφών επαληθεύουμε την αλλαγή και τις τιμές των διαφορετικών περιόδων που προκύπτουν.

Μέρος Β

Στο μέρος Β υλοποιήσαμε έναν transmitter για το uart κανάλι μας. Αρχικά ο transmitter περιλαμβάνει ένα module baud\_controller και ασύγχρονο reset. Ακόμα ο αποστολέας λειτουργεί σε ταχύτητα 16 φορές μικρότερη από τον δέκτη, πράγμα που υλοποιείται με έναν μετρητή 4-bits. Σύμφωνα με το uart όταν το κανάλι δεν χρησιμοποιείται το txd(η έξοδος του αποστολέα) είναι 1. Έπειτα ο transmitter δέχεται ένα σήμα tx\_en το οποίο τον ενεργοποιεί η τον απενεργοποιεί ανάλογα με την τιμή του. Αν ο transmitter είναι ενεργοποιημένος τότε υπάρχουν 5 καταστάσεις στις οποίες μπορεί να βρίσκεται. Αυτές είναι οι εξής:

Κατάσταση IDLE: Σε αυτή την κατάσταση ο transmitter περιμένει να παραλάβει τα νέα προς μετάδοση δεδομένα. Αυτό συμβαίνει όταν το testbench μας κάνει το σήμα tw\_wr 1. Όταν γίνει αυτό ο αποστολέας αποθηκεύει το byte που βρίσκεται στην είσοδο του σε εσωτερικούς καταχωρητές και ξεκινάει την διαδικασία της μετάδοσης μεταβαίνοντας στην κατάσταση START

Κατάσταση START: Σε αυτήν την κατάσταση ο αποστολέας κρατάει για μια περίοδό του το txd στο 0 για να δηλώσει ότι ξεκινάει η μετάδοση του επόμενου byte και μετά προχωράει στην κατάσταση DATA.

Κατάσταση DATA: Σε αυτήν την κατάσταση χρησιμοποιούμε έναν μετρητή για να γνωρίζουμε σε ποιο bit είμαστε. Το txd γίνεται ίσο με το αντίστοιχο bit που στέλνουμε σε κάθε περίοδο του αποστολέα. Ταυτόχρονα υπολογίζουμε και το parity bit, το οποίο μετράει τον αριθμό των άσσων του μηνύματος και αν αυτός είναι μονός τότε γίνεται 1, ενώ αν είναι ζυγός γίνεται 0. Αυτό υλοποιείται με την συνάρτηση xor. Αφού αποσταλούν όλα τα bits περνάμε στην κατάσταση PARITY.

Κατάσταση PARITY: Σε αυτήν την κατάσταση αποστέλλουμε μέσω του txd το parity bit για μια περίοδο και μετά περνάμε στην κατάσταση STOP

Κατάσταση STOP: Σε αυτήν την φάση ο αποστολέας κάνει ξανά το σήμα txd 1 για να ενημερώσει ότι η αποστολή έχει τελειώσει και έπειτα ξανά περνάει στην κατάσταση IDLE.

Η επαλήθευση του κυκλώματος έγινε μέσω μια μονάδας ελέγχου η οποία άλλαζε τις τιμές στην είσοδο του αποστολέα και έπειτα ελέγχαμε μέσω των κυμματομορφών τα αποτελέσματα τις προσομοίωσης.

Μέρος Γ

Στο μέρος αυτό υλοποιήσαμε έναν δέκτη για το uart κανάλι μας. Όπως αναφέρθηκε και πριν ο δέκτης λειτουργεί σε ταχύτητα 16 φορές μεγαλύτερη από τον αποστολέα για να καταφέρνει να κάνει sample τα bits στην μέση περίπου της διάδοσής τους. Και ο δέκτης όπως και ο αποστολέας περιέχει ένα module baud\_control.Ακόμα ο δέκτης συγχρονίζει το σήμα rxd που παραλαμβάνει από την έξοδο του αποστολέα πριν το χρησιμοποιήσει. Επίσης όμοια με τον αποστολέα, ο δέκτης περιλαμβάνει και αυτός ένα σήμα ενεργοποίησής του το rx\_en. Αν ο δέκτης είναι ενεργοποιημένος τότε υπάρχουν 4 καταστάσεις στις οποίες μπορεί να βρίσκεται:

Κατάσταση START: Αυτή είναι η αρχική κατάσταση του δέκτη. Σε αυτήν την κατάσταση ο δέκτης περιμένει ελέγχοντας σε κάθε περίοδό του την είσοδο rxd. Με το που γίνει 0, ο δέκτης θεωρεί ότι αυτό είναι σημάδι ότι ξεκίνησε η αποστολή καινούργιου byte. Έτσι μετράει 16 περιόδους για να συγχρονιστεί και μετά περνάει στην κατάσταση DATA.

Κατάσταση DATA: Σε αυτήν την κατάσταση ο δέκτης αφού έχει συγχρονιστεί με τα δεδομένα που έρχονται δειγματοληπτεί περίπου στην μέση της μετάδοσης, δηλαδή στην 8η περίοδό του. Έτσι μετράει 8 περιόδους, δειγματοληπτεί το επόμενο bit το οποίο αποθηκεύεται σε εσωτερικό 8bit καταχωρητή, σύμφωνα με τον μετρητή, και έπειτα μετράει τις επόμενες 7 και περνάει στο επόμενο bit. Ταυτόχρονα υπολογίζει και το parity bit των εισερχόμενων δεδομένων με τον ίδιο τρόπο που το υπολογίζει και ο αποστολέας. Όταν φτάσει στο τελευταίο bit στην 16η περίοδό του τότε περνάει στην κατάσταση PARITY .

Κατάσταση PARITY: Σε αυτήν την κατάσταση ο δέκτης δειγματοληπτεί το parity bit με τον ίδιο τρόπο που δειγματοληπτεί και τα δεδομένα και έπειτα περνάει στην κατάσταση STOP

Κατάσταση STOP: Σε αυτήν την κατάσταση μετράμε και δειγματολειπτούμε το τελευταίο bit. Σε περίπτωση που δεν είναι 1, όπως θα έπρεπε να είναι, τότε έχουμε framing error και για αυτό σηκώνεται το σήμα ferror. Αν το τελευταίο bit δειγματοληφθεί και είναι (σωστά) 1 τότε ελέγχουμε αν το parity bit που λάβαμε είναι ίσο με το parity bit που υπολογίσαμε. Αν είναι διαφορετικά τότε έχουμε parity error και σηκώνεται το σήμα perror. Και στις δύο αυτές περιπτώσεις τα δεδομένα που δειγματολήφθησαν δεν προάγονται στην έξοδο και η έξοδος παραμένει 8’b00000000. Αν δεν συμβεί κανένα από αυτά τα δύο errors τότε προάγουμε τα δεδομένα μας στην έξοδο του δέκτη. Σε κάθε περίπτωση μετά την 8η περίοδο όποιο και να είναι το αποτέλεσμα των παραπάνω ελέγχων μεταφερόμαστε στην κατάσταση IDLE και περιμένουμε το επόμενο byte.

Η επαλήθευση του μέρους Γ έγινε με ένα testbench το οποίο περνάει με μετρημένες καθυστερήσεις διάφορες αλληλουχίες 0 και 1 στην είσοδο rxd και ελέγχουμε από τα αποτελέσματα rxvalid, ferror, perror και rxdata αν όλα διαβάστηκαν σωστά. Στο testbench περιέχονται περιπτώσεις οι οποίες είναι πειραγμένες έτσι ώστε να προκύπτουν τα ferror και perror.

Μέρος Δ

Σε αυτό το μέρος υλοποιήθηκε ένα module uart το οποίο περιλαμβάνει ένα ολόκληρο κανάλι μετάδοσης uart. Το κανάλι αυτό περιέχει έναν αποστολέα και έναν δέκτη τους οποίους συνδέει με την υλοποίηση της σύνδεσης txd(έξοδος αποστολέα)->rxd(είσοδος δέκτη).

Η επαλήθευση αυτού του μέρους έγινε με ένα πλαίσιο ελέγχου testbench το οποίο ελέγχει αν το σήμα busy είναι ένα και αν είναι περιμένει μέχρι να γίνει 0 και μετά περνάει καινούργια τιμή στον αποστολέα. Τέλος, γίνεται έλεγχος μέσω τον κυμματομορφών για την σωστή λειτουργία των σημάτων rxvalid, rxferror και rxperror καθώς και για την σωστή μεταφορά των δεδομένων με σύγκρισή τους.