HY220: Εργαστήριο Ψηφιακών Κυκλωμάτων

Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης

Χειμερινό Εξάμηνο 2016

Εργαστήριο 3:
Υλοποίηση Ελεγκτή Μνήμης

1 Δεκεμβρίου έως 11 Δεκεμβρίου 2016 (11 μέρες)

1.1 Σκοπός της Εργαστηριακής Άσκησης

Σε αυτήν την Άσκηση θα επιχειρήσουμε να δημιουργήσουμε κύκλωμα που να υλοποιεί έναν ελεγκτή μνήμης 256 θέσεων (depth), με πλάτος 8-bit (width) στην κάθε θέση. Ο ελεγκτής μνήμης στην συνέχεια πρέπει να συνδεθεί με ένα κύκλωμα σειριακής επικοινωνίας (UART) (βλ. παρακάτω σχήμα), το οποίο προωθεί εντολές για ανάγνωση και εγγραφή δεδομένων μέσω σειριακής σύνδεσης.

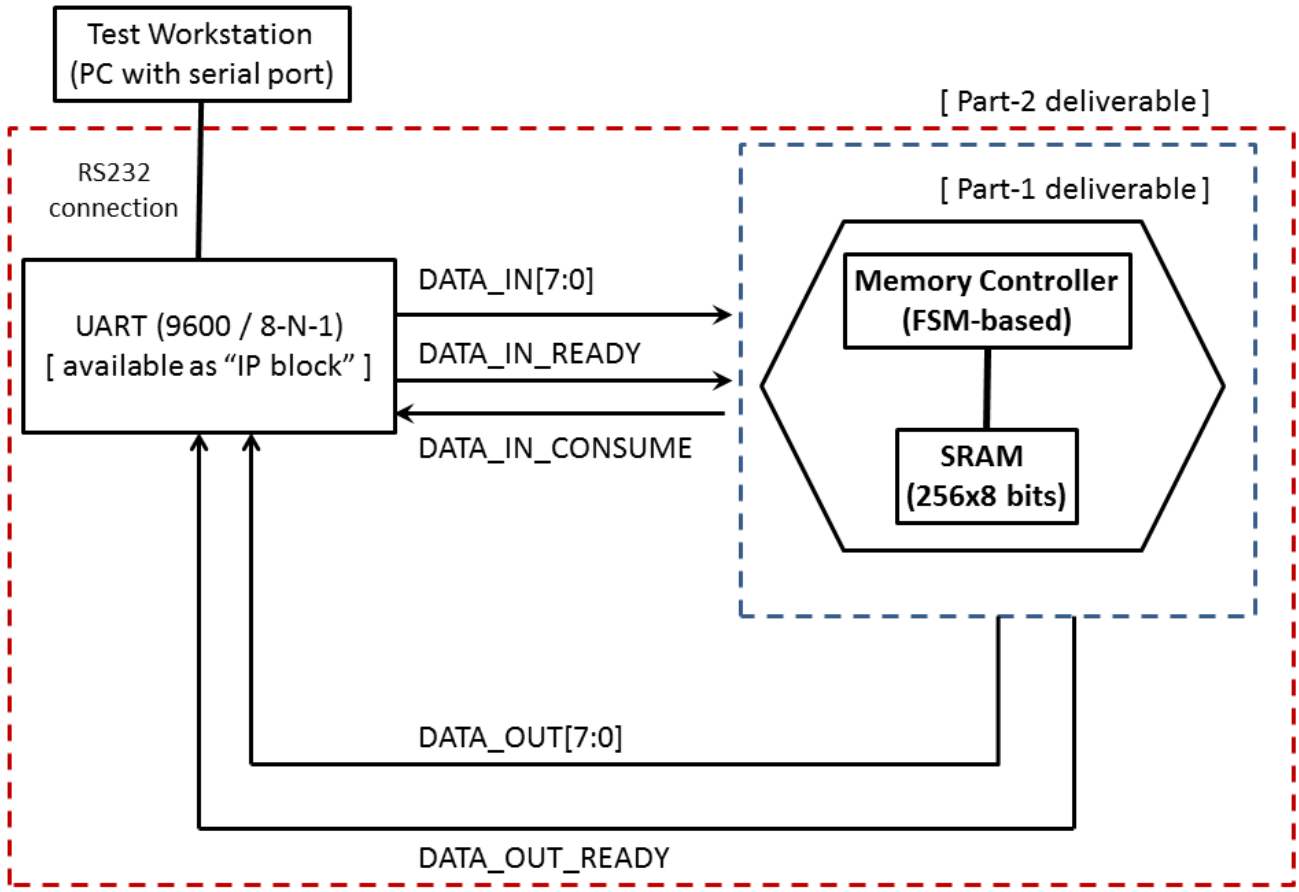

Figure 1: Σύστημα ελέγχου μνήμης (256 θέσεις, 8-bit δεδομένα), συνδεδεμένο με σειριακή διεπαφή (interface).

Οι εντολές εγγραφής ή ανάγνωσης και τα δεδομένα που περιέχουν αυτές αν είναι εντολές εγγραφής, δίδονται από το σειριακό τερματικό ως ακολουθίες χαρακτήρων, τους παραλαμβάνει το

UART, τους αποθηκεύει στην ουρά του Receiver του έναν προς έναν και τους διαθέτει στον ελεγκτή που θα υλοποιήσετε σε αυτό τον εργαστήριο.

Στις περιπτώσεις που ο ελεγκτής θα δέχεται εντολή ανάγνωσης τα αποτελέσματα θα στέλνονται στο σειριακό τερματικό.
Με χρήση του σειριακού τερματικού θα πρέπει να επιδείξετε την λειτουργία του συστήματος αυτού.

1.2 Σχεδιασμός Συστήματος και Αναφορά

Πρώτο βήμα για την υλοποίηση της εργασίας είναι ή σύνταξη αναφοράς παρόμοιας με την πρότυπη αναφορά που έχει αναρτηθεί στον site του μαθήματος και πρέπει να περιγράφει κάθε βήμα της ανάπτυξης του συστήματός σας.

Για την λογική ελέγχου της μνήμης (BRAM block) η οποία παράγεται από το εργαλείο σχεδίασης Xilinx ISE και ΔΕΝ χρειάζεται να την υλοποιήσετε εσείς, θα χρειαστεί να σχεδιάσετε κατάλληλη μηχανή πεπερασμένων καταστάσεων (Finite State Machine – FSM) με την βοήθεια της οποίας ο ελεγκτής θα αναγνωρίζει τι είδους εντολή αντιπροσωπεύουν οι χαρακτήρες που έλαβε από το UART.
Οι εντολές δίδονται από το σειριακό τερματικό ως ακολουθίες χαρακτήρων ASCII, με το εξής συντακτικό:

- ENQ: **<W><SPACE><ADDRESS><DATA><CR>**
- DEC: <**R><SPACE><ADDRESS>**<**CR**>

Τα DATA και ADDRESS είναι μία ποσότητα 8 bit το καθένα που αναπαρίσταται από 2 δεκαεξαδικά ψηφία. Το ADDRESS πεδίο και στις δύο εντολές καθορίζει τη θέση μνήμης όπου θα γίνει η εγγραφή ή η ανάγνωση.
Επειδή το τερματικό ‘στέλνει’ και ‘καταλαβαίνει’ ASCII χαρακτήρες, ενώ εμείς αποθηκεύουμε 8bit ποσότητες στη μνήμη αλλά και χρειαζόμαστε τον 8-bit αριθμό που αναπαριστούν οι ASCII χαρακτήρες ώστε να ορίσουμε την διεύθυνση, είναι προφανές ότι πρέπει να φτιάξετε έναν Decoder ο οποίος δέχεται ASCII χαρακτήρες και να μετατρέπει τον ASCII χαρακτήρα στον αριθμό που θέλουμε σύμφωνα με τη δεκαεξαδική αναπαράσταση. Δηλαδή θα παίρνει από την μία είσοδο τους χαρακτήρες ASCII 0-9 και A-F (κεφαλαία), και θα βγάζει αριθμό 0-15 στην έξοδο.
Επίσης πρέπει να φτιάξετε έναν Encoder που στην περίπτωση των DATA να κάνει την αντίστροφη δουλειά. Να παίρνει από τη μία είσοδο αριθμούς 0-15. Και από την έξοδο να βγάζει χαρακτήρες ASCII 0-9 και A-F (κεφαλαία).

Στην περίπτωση όπου ο ελεγκτής διαπιστώσει ότι διαχειρίζεται μία εντολή εγγραφής, πρέπει να αποθηκεύει το πεδίο <DATA> στην θέση μνήμης <ADDRESS>. Πχ, δίδεται η εντολή **W 5340 <CR>**. Το W σηματοδοτεί εντολή εγγραφής. Η διεύθυνση στην οποία αναφέρεται η εντολή είναι η 0x53 = 8’b01010011 και ΟΧΙ τους χαρακτήρες ASCII ‘5’= 8’b00110101 και ‘3’= 8’b00110011. Η τιμή που πρέπει να αποθηκευτεί στη μνήμη είναι το 0x40 = 8’b0100000. ΟΧΙ τους χαρακτήρες ASCII ‘4’=8’b00110100 και ‘0’=8’b00110000.

Αντίστοιχα, στην περίπτωση όπου ο ελεγκτής διαπιστώσει ότι διαχειρίζεται μία εντολή ανάγνωσης, πρέπει να διαβάσει το byte από την μνήμη που αντιστοιχεί στην ζητούμενη θέση και να στείλει στο UART τους κατάλληλους ASCII χαρακτήρες για εκτύπωση στο τερματικό. Το τελευταίο, θα αναλάβει να αποστείλει τους χαρακτήρες στο σειριακό τερματικό.
Πχ, δίδεται η εντολή **R 53<CR>**. Η μνήμη στη θέση 0x53 έχει μέσα data για κατανάλωση την τιμή

8’b10110001=0x71. Πρέπει να εκτυπωθεί στο τερματικό οι δύο ASCII χαρακτήρες ‘7’ και ‘1’. ΟΧΙ ο ASCII χαρακτήρας 0x71= ‘q’.

Στην προσομοίωση δεν χρειάζεται να περιλάβετε το UART, θα προσομοιώσετε τα σήματα εισόδου και εξόδου του UART με “test vectors".
Στο επόμενο βήμα, το κύκλωμα ελέγχου ουράς δεδομένων πρέπει να συνδυαστεί με πραγματικό κύκλωμα UART (το οποίο θα δοθεί έτοιμο για χρήση).

Κάθε εντολή τερματίζεται με τον ειδικό χαρακτήρα carriage return (<CR>).

1.3 Συγγραφή του Κώδικα Verilog και Προσομοίωση

Αφού ολοκληρώσετε το implementation σκέλος της αναφοράς, πρέπει να το μετατρέψετε σε κώδικα Verilog. Αμέσως μετά πρέπει να δημιουργήσετε ένα testbench module με το οποίο θα δοκιμάσετε να προσομοιώσετε το σύστημα.

1.4 Δημιουργία UCF και Τοποθέτηση

Αφού ολοκληρώσετε τα προηγούμενα βήματα, πρέπει να δημιουργήσετε το Αρχείο Περιορισμών Χρήστη (User Constraints File – UCF) δηλώνοντας τις εισόδους και τις εξόδους που θα χρησιμοποιήσετε στην πλακέτα και την τάση που θα τους ασκήσετε, όπως δείχνει το .ucf αρχείο που λάβατε στο εργαστήριο 0 το οποίο είναι κομμάτι του master ucf που έχει δοθεί από τους κατασκευαστές.

1.5 Παράδοση

Στις 11/12/2016 θα παραδώσετε την αναφορά σας και τον κώδικα, και την επόμενη εβδομάδα (12/12-16/12) θα εξεταστείτε ελέγχοντας το σύστημά σας στην πλακέτα.

Για την παράδοση χρησιμοποιείτε την εφαρμογή TURNIN από τα μηχανήματα linux του τμήματος. Γράφουμε turnin lab3-report@hy220 για την αναφορά ενώ για τον κώδικα turnin lab3-code.

Για τις ώρες της εξέτασης κλείνετε timeslot μέσω του Rendezvous.

ΥΠΕΝΘΥΜΙΣΗ!! :

ΠΑΡΑΔΙΔΕΤΕ ΜΑΖΙ ΜΕ ΟΛΑ ΤΑ ΑΡΧΕΙΑ ΠΟΥ ΠΕΡΙΕΧΟΥΝ ΤΟΝ VERILOG ΚΩΔΙΚΑ ΣΑΣ, ΕΝΑ ΑΡΧΕΙΟ ΤΟ ΟΠΟΙΟ ΠΕΡΙΕΧΕΙ ΤΟ TOP MODULE. ΤΟ TOP MODULE ΠΡΕΠΕΙ ΝΑ ΠΕΡΙΕΧΕΙ ΤΟΝ ΕΛΕΓΚΤΗ ΣΑΣ ΚΑΙ ΤΟ UART ΚΑΘΩΣ ΚΑΙ ΤΗΝ ΣΥΝΔΕΣΗ ΜΕΤΑΞΥ ΤΟΥΣ, ΔΙΟΤΙ ΑΥΤΟ ΕΙΝΑΙ ΤΟ ΟΛΟΚΛΗΡΩΜΕΝΟ ΣΥΣΤΗΜΑ ΣΤΟ ΟΠΟΙΟ ΘΑ ΕΞΕΤΑΣΤΕΙΤΕ.