HY220: Εργαστήριο Ψηφιακών Κυκλωμάτων

Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης
Χειμερινό Εξάμηνο 2020

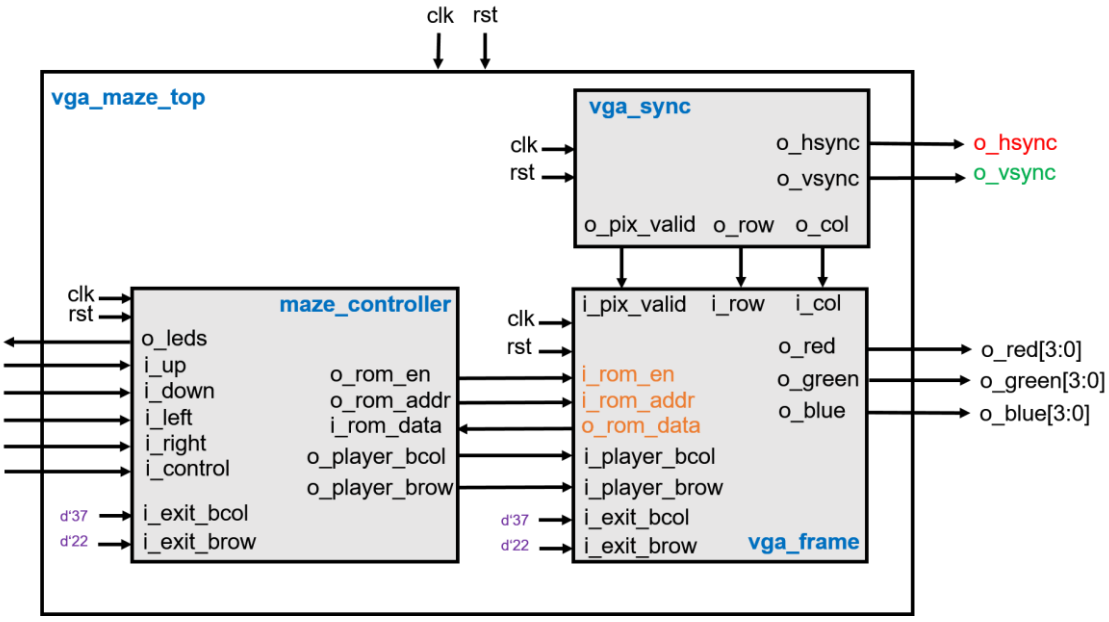
4η Σειρά Ασκήσεων

Ημερομηνία Παράδοσης: 31/01/2021 23:59

Άσκηση 4.1

Στην άσκηση αυτή θα υλοποιήσετε την τελευταία φάση του παιχνιδιού που περιλαμβάνει τον έλεγχο των κινήσεων και της θέσης του παίκτη μέσω μηχανών πεπερασμένων καταστάσεων (FSM). Η άσκηση αυτή βασίζεται στο υλικό των Ασκήσεων 2.1 & 3.1 και σας δίνεται επιπλέον κώδικας για μια δίπορτη ROMs που θα χρησιμοποιηθεί για τον έλεγχο των κινήσεων του παίκτη γύρω από τους τοίχους του λαβυρίνθου. Επίσης σας δίνεται ένα κατάλληλο testbench και reference outputs και θα μπορείτε να χρησιμοποιήσετε τον προσομοιωτή VGA (VGA Simulator) για να βλέπετε τι θα εμφανίζονταν σε μια πραγματική οθόνη από τον κώδικά σας.

Για την *Άσκηση 4.1* το σχέδιο που σας δίνεται είναι τροποποιημένο σε σχέση με το σχέδιο της *Άσκησης 3.1* και φαίνεται στην παρακάτω εικόνα:



Το σχέδιο έχει την εξής ιεραρχία:

- **vga_maze_top** *(vga_maze_top.sv)*: σας δίνεται μια πρώτη έκδοση και σε σχέση με την *Άσκηση 3.1* έχει προστεθεί το μπλοκ maze_controller, πόρτες για κουμπιά (up, down, left, right, control) και leds, και επιπλέον πόρτες στο vga_frame για να μπορεί ο maze_controller να έχει πρόσβαση στην ROM του λαβυρίνθου. Οι συνδέσεις φαίνονται στο σχήμα. Θα χρειαστεί να τροποποιήσετε αυτό το αρχείο σύμφωνα με την εκφώνηση.
- **vga_sync** *(vga_sync.sv)*: Το μπλοκ αυτό υλοποιεί το χρονισμό του πρωτοκόλλου VGA για ανάλυση 640 x 480 και πρέπει να είναι η υλοποίηση σας από την *Άσκηση 2.1*.
- **vga_frame** *(vga_frame.sv)*: Το μπλοκ αυτό είναι υπεύθυνο για τη δημιουργία των χρωμάτων RGB (red/green/blue) για κάθε pixel που εμφανίζεται στην οθόνη για να

εμφανιστεί ο λαβύρινθος και ο παίκτης στην οθόνη και πρέπει να είναι η υλοποίηση σας από την *Άσκηση 3.1* με επιπλέον αλλαγές όπως περιγράφονται παρακάτω.

- **maze_controller** *(maze_controller.sv)*: Το μπλοκ αυτό είναι υπεύθυνο για τον έλεγχο των κινήσεων και της θέσης του παίκτη μέσω μηχανών πεπερασμένων καταστάσεων (FSM). Λεπτομέρειες για το τι θα πρέπει να υλοποιήσετε περιγράφονται παρακάτω.
- **rom** *(rom.sv)*: Το μπλοκ σας δίνεται έτοιμο και δημιουργεί μια σύγχρονη ROM παραμετρικού μεγέθους και την αρχικοποιεί με τα περιεχόμενα ενός αρχείου που δίνεται επίσης ως παράμετρος. Οι πόρτες της ROM είναι: clk, en, addr, dout. Η πόρτα en (enable) σηματοδοτεί μια νέα ανάγνωση, η πόρτα addr (address) είναι για να δοθεί η είσοδος για τη διεύθυνση που θέλουμε να διαβάσουμε και η πόρτα εξόδου dout (data out) βγάζει τα περιεχόμενα της διεύθυνσης addr στον «επόμενο κύκλο». Η παράμετρος size ορίζει τον αριθμό των θέσεων της ROM, η παράμετρος width το πλάτος σε bits της κάθε θέσης της ROM (16-bits στην περίπτωσή μας) και η παράμετρος file δηλώνει το αρχείο που θα χρησιμοποιηθεί για αρχικοποίηση των περιεχομένων της ROM. Επίσης στο φάκελο roms σας δίνονται 3 αρχεία (maze1.rom, player.rom, exit.rom) που θα χρησιμοποιηθούν για αρχικοποίηση των δεδομένων των διαφόρων ROM που θα χρησιμοποιήσουμε για το εργαστήριο (περισσότερες λεπτομέρειες δίνονται παρακάτω).
- **rom_dp** *(rom_dp.sv)*: Το μπλοκ σας δίνεται έτοιμο και είναι μια δίπορτη ROM βασισμένη στην μονόπορτη ROM που περιγράφεται παραπάνω.
- **debouncer** *(debouncer.sv):* Το μπλοκ αυτό λύνει το πρόβλημα της «αναπήδησης» των μηχανικών κουμπιών που χρησιμοποιούνται για την κίνηση του παίκτη και σας δίνεται έτοιμο και είναι συνδεδεμένο καταλλήλως στο vga_maze_top.
- **vga_tb** *(vga_tb.sv)*: ένα testbench *για προσομοίωση* που δημιουργεί το ρολόι (25 MHz – 40 ns), το reset, και τα σήματα από τα κουμπιά. Το testbench αποθηκεύει την έξοδο του κυκλώματός σας σε ένα αρχείο (vga_log.txt) με το κατάλληλο format έτσι ώστε να μπορείτε να χρησιμοποιήσετε τον VGA Simulator για να βλέπετε τι θα εμφανίζεται στην οθόνη από τον κώδικά σας. Περισσότερες λεπτομέρειες για τον VGA Simulator παρακάτω.
- **Reference output**: Στο φάκελο reference υπάρχει ένα πρότυπο ***vga_log.txt*** output που είναι αυτό που θα πρέπει να δημιουργεί ένας σωστός κώδικας. Μπορείτε να κάνετε diff το αρχείο που παράγει ο κώδικάς σας με το reference output για να εντοπίσετε λάθη.
- **VGA Simulator:** Μέσα στο φάκελο vga-simulator υπάρχει μια ιστοσελίδα που μπορείτε να ανοίξετε τοπικά στον web browser σας. Εκεί μπορείτε να επιλέξετε το log file που έχει δημιουργηθεί από την προσομοίωσή σας και όταν πατήσετε το κουμπί submit τότε θα εμφανιστεί σε μια εικονική VGA οθόνη η έξοδός σας. Μπορείτε να το δοκιμάσετε επίσης με το reference output. Μην αλλάξετε τις παραμέτρους που υπάρχουν ήδη στη σελίδα! Credits: Ο VGA Simulator έχει δημιουργηθεί από τον Eric Eastwood στο παρακάτω website http://ericeastwood.com/lab/vga-simulator/

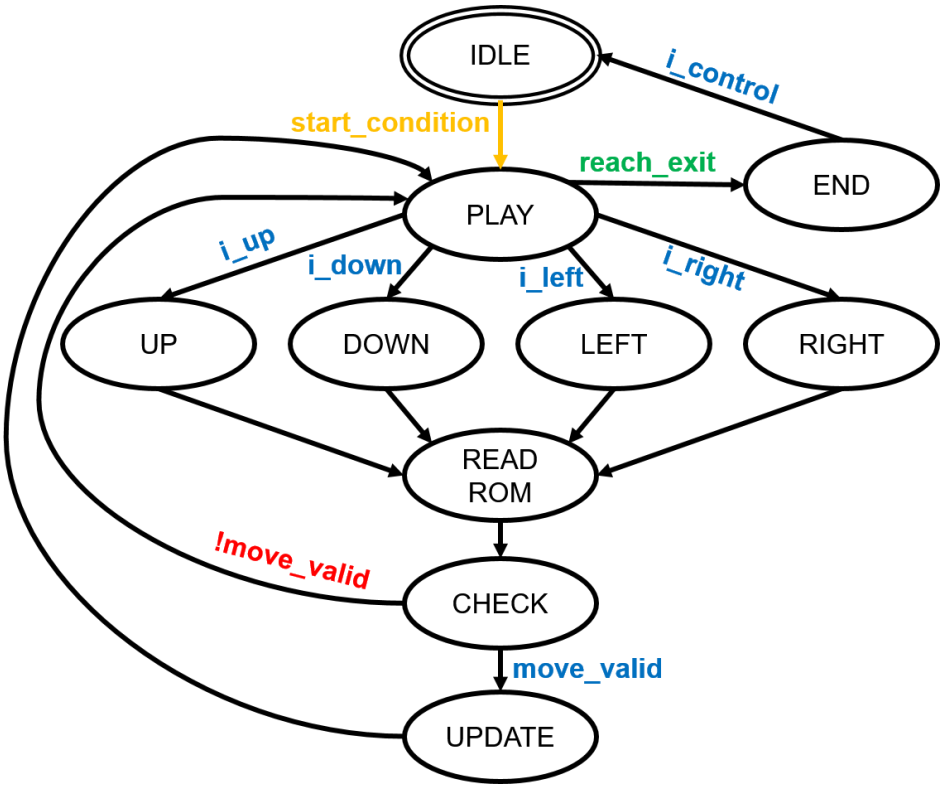
Τι πρέπει υλοποιήσετε για την Άσκηση 4.1:

Για την άσκηση αυτή αρχικά πρέπει να τροποποιήσετε το module **vga_frame** που υλοποιήσατε για την *Άσκηση 3.1* έτσι ώστε να χρησιμοποιεί τη δίπορτη ROM (rom_dp) για την maze_rom και να βγάζει την δεύτερη πόρτα της ROM (en_b, addr_b, dout_b) στις αντίστοιχες πόρτες του module (i_rom_en, i_rom_addr, o_rom_data). *Οι υπόλοιπες ROM παραμένουν ως έχουν.*

Στη συνέχεια θα πρέπει να υλοποιήσετε το module **maze_controller** του οποίου ένας άδειος σκελετός σας δίνεται. Το module πρέπει να υλοποιεί την βασική λειτουργία του παιχνιδιού. Με βάση τις εισόδους i_control, i_up, i_down, i_left, και i_right που προέρχονται από τα κουμπιά, πρέπει να δημιουργεί τις εξόδους o_player_bcol και o_player_brow που δείχνουν την τρέχουσα θέση του παίκτη στήλη/γραμμή (έτσι το module vga_frame θα εμφανίζει τον παίκτη στην

κατάλληλη θέση κάθε φορά). Για να αποφασιστεί αν ο παίκτης μπορεί να μετακινηθεί σε επόμενη θέση θα πρέπει να ελεγχθεί αν η επόμενη/υποψήφια θέση στο λαβύρινθο (διαβάζοντας την ROM) «πέφτει» πάνω σε τοίχο και δεν είναι έγκυρη ή υπάρχει διάδρομος και έτσι η κίνηση είναι έγκυρη. Επίσης αν ο παίκτης φτάσει στην έξοδο που δίνεται από τις πόρτες i_exit_bcol και i_exit_brow τότε το παιχνίδι ολοκληρώνεται.

Αρχικά στο παιχνίδι ο παίκτης ξεκινάει από την θέση (column, row) = (1, 0). Αν πατηθεί το κουμπί i_right θα πρέπει να αυξηθεί η στήλη κατά 1. Αν πατηθεί το κουμπί i_left θα πρέπει να μειωθεί η στήλη κατά 1. Αν πατηθεί το κουμπί i_down θα πρέπει να αυξηθεί η γραμμή κατά 1. Αν πατηθεί το κουμπί i_up θα πρέπει να μειωθεί η γραμμή κατά 1. Όλες οι παραπάνω κινήσεις πρέπει να γίνονται υπό την προϋπόθεση ότι δεν «πέφτουν» σε τοίχο του λαβυρίνθου και δεν βγαίνουν εκτός ορίων. Για την υλοποίηση των βημάτων ελέγχου και την ενημέρωση της θέσης του παίκτη σας δίνεται στο παρακάτω σχήμα το διάγραμμα καταστάσεων της FSM που πρέπει να υλοποιήσετε. *Η FSM είναι απλοποιημένη και υπονοεί την παραμονή στην ίδια κατάσταση αν δεν υπάρχει νέα είσοδος.*



Αρχικά η FSM είναι στην κατάσταση IDLE και περιμένει να πατηθεί «3 συνεχόμενες φορές» το κουμπί i_control, αυτό ονομάζεται στο σχήμα σαν συνθήκη start_condition. Όταν συμβεί το start_condition πρέπει να μεταβαίνει στην κατάσταση PLAY όπου περιμένει την επόμενη κίνηση του παίκτη. Αν παρεμβληθεί πάτημα άλλου κουμπιού εκτός του i_control τότε πρέπει να μετρηθούν από την αρχή 3 «συνεχόμενα» πατήματα του i_control.

Στην κατάσταση PLAY η FSM περιμένει την επόμενη κίνηση του παίκτη και αναλόγως με το ποιο κουμπί θα πατηθεί η FSM μεταβαίνει σε μία από τις καταστάσεις UP/DOWN/LEFT/RIGHT. Στην κατάσταση PLAY επίσης ελέγχεται αν ο παίκτης έχει φτάσει στην έξοδο του λαβυρίνθου (reach_exit) δηλαδή αν η τρέχουσα θέση είναι ίση με τη θέση εξόδου του λαβυρίνθου που έρχεται από τις πόρτες i_exit_bcol και i_exit_brow. Αν ο παίκτης έχει φτάσει στην έξοδο τότε η FSM μεταβαίνει στην κατάσταση END.

Στις καταστάσεις UP/DOWN/LEFT/RIGHT πρέπει να δημιουργηθεί μια νέα «υποψήφια» θέση αυξάνοντας ή μειώνοντας καταλλήλως τη στήλη (column) ή τη γραμμή (row). Για την υποψήφια θέση πρέπει να κρατάτε σε καταχωρητές (έστω new_bcol και new_brow) την υποψήφια στήλη και υποψήφια γραμμή οι οποίες θα έχουν προκύψει από την τρέχουσα στήλη και τρέχουσα γραμμή αναλόγως. Μετά η FSM μεταβαίνει στην κατάσταση READROM.

Στην κατάσταση READROM η FSM χρησιμοποιεί την υποψήφια γραμμή και στήλη για να διαβάσει την κατάλληλη διεύθυνση από την ROM έτσι ώστε να «δει» αν υπάρχει τοίχος (ανατρέξτε στην Άσκηση 3.1 για δείτε πως πρέπει διευθυνσιοδοτήσετε την maze_rom) και μεταβαίνει στην κατάσταση CHECK.

Στην κατάσταση CHECK εμφανίζονται τα δεδομένα της ROM και πρέπει να ελεγχθεί αν η κίνηση είναι έγκυρη (move_valid). Η κίνηση είναι έγκυρη αν τα δεδομένα που διαβάζετε από την ROM δεν είναι RGB(0,0,0) δηλαδή δεν είναι «μαύρο» (δεν είναι τοίχος). Αν η κίνηση είναι έγκυρη (move_valid) τότε μεταβαίνει στην κατάσταση UPDATE. Αν η κίνηση δεν είναι έγκυρη (!move_valid) τότε η FSM αγνοεί την κίνηση και επιστρέφει στην κατάσταση PLAY.

Στην κατάσταση UPDATE ενημερώνεται η τρέχουσα θέση με την νέα θέση και επιστρέφει στην κατάσταση PLAY.

Στην κατάσταση END η FSM περιμένει το πάτημα του i_control για να μεταβεί στην κατάσταση IDLE.

Κωδικοποιήστε τις καταστάσεις με τις παρακάτω τιμές: IDLE=1, PLAY=2, UP=3, DOWN=4, LEFT=5, RIGHT=6, READROM=7, CHECK=8, UPDATE=9, END=10. Οδηγείστε την έξοδο o_leds με την κατάσταση της FSM για να μπορείτε να βλέπετε σε ποια κατάσταση βρίσκετε η FSM.

Τι πρέπει να παραδώσετε:

Μέχρι την ημερομηνία της προθεσμίας πρέπει να παραδώστε σε ένα zip/tar.gz αρχείο (e.g. hw4.zip ή hw4.tar.gz) τα παρακάτω:

1. Τον SystemVerilog RTL «ΟΛΩΝ» των αρχείων του σχεδίου.

Στείλτε το **hw4.zip** σας με e-mail στο hy220@csd.uoc.gr με τίτλο: *"HW4 – Ονοματεπώνυμο – ΑΜ"*.

Οι κώδικες θα ελέγχονται για αντιγραφές με ειδικό λογισμικό!