HY220: Εργαστήριο Ψηφιακών Κυκλωμάτων

Τμήμα Επιστήμης Υπολογιστών, Πανεπιστήμιο Κρήτης
Χειμερινό Εξάμηνο 2020

2η Σειρά Ασκήσεων

Ημερομηνία Παράδοσης: Δευτέρα 21/12/2020 23:59

Άσκηση 2.1

Η άσκηση αυτή είναι η πρώτη φάση για το παιχνίδι «Λαβύρινθος» (Maze). Περιληπτικά, το τελικό παιχνίδι θα εμφανίζει σε VGA οθόνη ένα λαβύρινθο και τη φιγούρα ενός παίκτη. Ο χρήστης θα μπορεί να μετακινήσει τον παίκτη με «κουμπιά» και ο σκοπός είναι να τον οδηγήσει στην έξοδο το ταχύτερο δυνατόν.

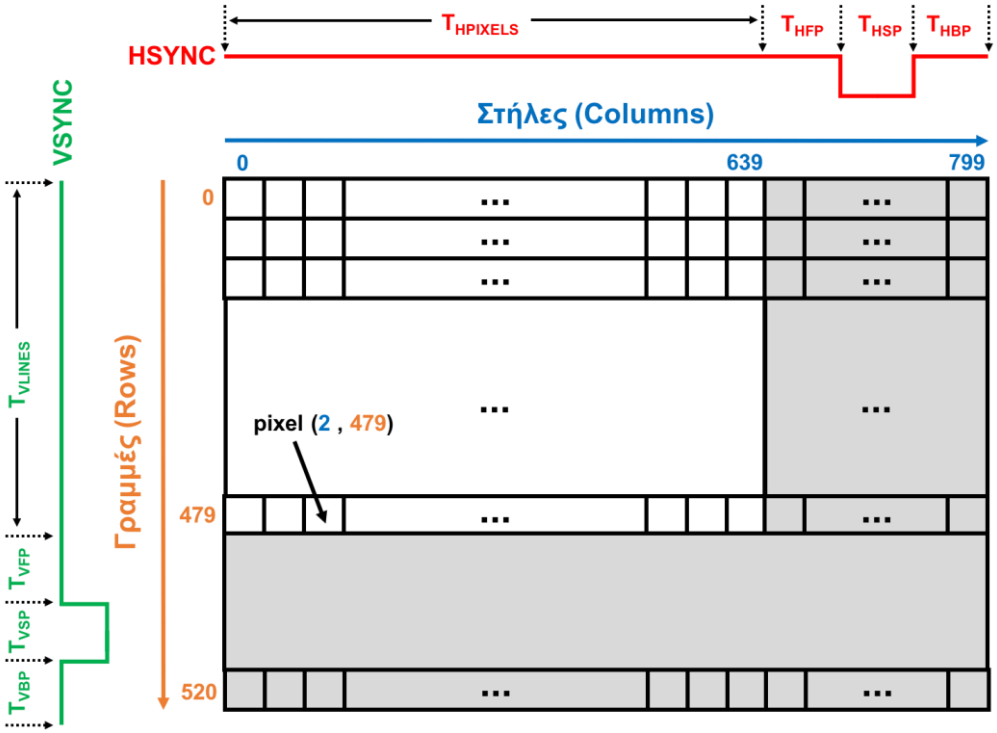
Στην *Άσκηση 2.1* θα υλοποιήσετε την πρώτη φάση που περιλαμβάνει την οδήγηση μιας οθόνης VGA για την εμφάνιση οριζόντιων γραμμών με διαφορετικά χρώματα. Για την άσκηση αυτή θα σας δοθεί ο σκελετός αρκετών κομματιών σε SystemVerilog, ένα κατάλληλο testbench, reference outputs και ένας προσομοιωτής VGA (VGA Simulator) που σας δίνει τη δυνατότητα να βλέπετε τι θα εμφανίζονταν σε μια πραγματική οθόνη από τον κώδικά σας.

Χρονισμός VGA 640 x 480 @ 60Hz

Ένα πλήρες frame (εικόνα) αποτελείται από pixels που το καθένα περιέχει τιμές για τα χρώματά του σε RGB (red/green/blue). Το frame αποτελείται από στήλες (columns) και γραμμές (rows). Για κάθε γραμμή πρέπει να δίνονται τα pixels της κάθε στήλης σε διαδοχικούς κύκλους ρολογιού και μόλις ολοκληρωθεί ο απαιτούμενος αριθμός από pixels (columns) τότε πρέπει να σηματοδοτηθεί το τέλος της γραμμής έτσι ώστε να ξεκινήσει η επόμενη γραμμή (N+1) από την αρχή στην στήλη 0. Η σηματοδοσία για το τέλος της γραμμής γίνεται με τον παλμό HSYNC (horizontal synchronization – οριζόντιος συγχρονισμός). Όταν τελειώσουν όλες οι γραμμές τότε πρέπει να σηματοδοτηθεί το τέλος της εικόνας (frame) έτσι ώστε να ξεκινήσει το επόμενο frame από την αρχή στη γραμμή 0 και τη στήλη 0. Η σηματοδοσία για το τέλος του frame γίνεται με τον παλμό VSYNC (vertical synchronization – κατακόρυφος συγχρονισμός). Στην παρακάτω εικόνα φαίνεται το παράδειγμα ενός frame 640 x 480 (640 στήλες και 480 γραμμές).

Για κάθε γραμμή απαιτείται χρόνος $T_{HPIXELS}$ για τα ενεργά pixels που θέλουμε να εμφανίσουμε και μετά πρέπει να ακολουθήσει ο παλμός HSYNC (με κόκκινο στην εικόνα) ο οποίος είναι active-low για χρονισμό VGA 640 x 480. Για τη δημιουργία του παλμού HSYNC (δηλαδή μετά τα ενεργά pixels $T_{HPIXELS}$) απαιτείται κάποιος κενός χρόνος πριν τον παλμό T_{HFP} (horizontal front porch), κατά τη διάρκεια του παλμού T_{HSP} (horizontal sync pulse) και μετά τον παλμό T_{HBP} (horizontal back porch). Σε όλη αυτή τη διάρκεια τα pixels είναι «κενά», δεν λαμβάνονται υπόψιν και δεν εμφανίζονται (με γκρι στο δεξί μέρος της εικόνας).

Για κάθε εικόνα (frame) απαιτείται χρόνος T_{VLINES} για τις ενεργές γραμμές που θέλουμε να εμφανίσουμε και μετά πρέπει να ακολουθήσει ο παλμός VSYNC (με πράσινο στην εικόνα) ο οποίος είναι active-low για χρονισμό VGA 640 x 480. Για τη δημιουργία του παλμού VSYNC (δηλαδή μετά τις ενεργές γραμμές T_{VLINES}) απαιτείται κάποιος κενός χρόνος πριν τον παλμό T_{VFP} (vertical front porch), κατά τη διάρκεια του παλμού T_{VSP} (vertical sync pulse) και μετά τον παλμό T_{VBP} (vertical back porch). Σε όλη αυτή τη διάρκεια τα pixels είναι «κενά», δεν λαμβάνονται υπόψιν και δεν εμφανίζονται (με γκρι στο κάτω μέρος της εικόνας).

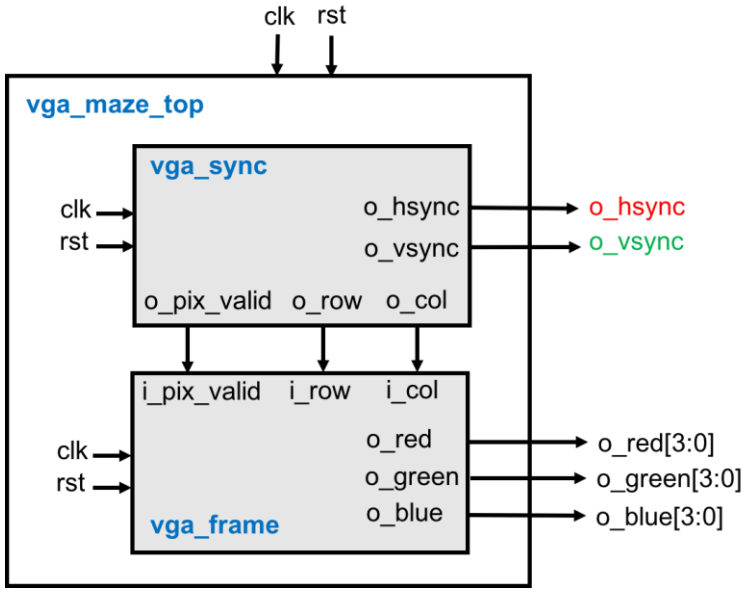


Για ανάλυση (resolution) 640 x 480 με ρυθμό ανανέωσης 60 Hz (60 frames το δευτερόλεπτο) απαιτείται ρολόι 25MHz (περίοδος ρολογιού 40 ns) και οι απαιτούμενοι χρόνοι φαίνονται στους πίνακες παρακάτω:

Horizontal	Clock Cycles	Time (ns)
$T_{HPIXELS}$	640	25600
T_{HFP}	16	640
T_{HSP}	96	3840
T_{HBP}	48	1920
Total	800	32000

Vertical	Rows	Clock Cycles	Time (us)
T_{VLINES}	480	384000	15360
T_{VFP}	10	8000	320
T_{VSP}	2	1600	64
T_{VBP}	29	23200	928
Total	521	416800	16672

Για την *Άσκηση 2.1* σας δίνεται το σχέδιο που φαίνεται στην παρακάτω εικόνα:



Το σχέδιο έχει την εξής ιεραρχία:

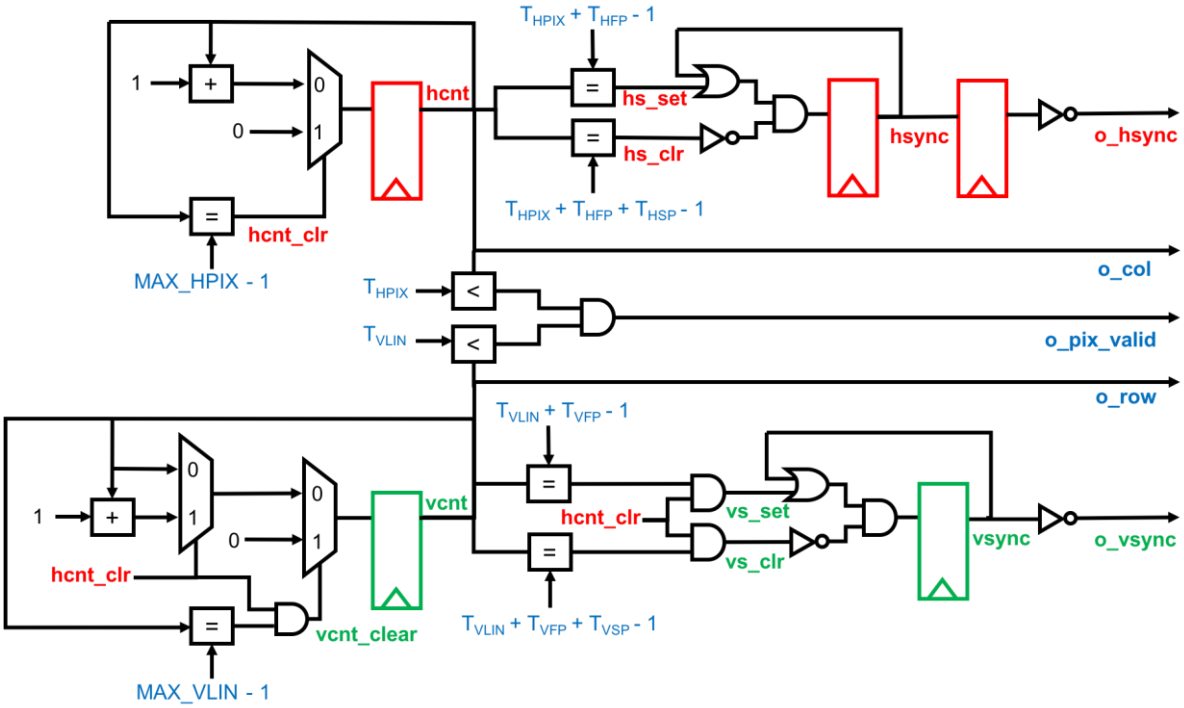
- **vga_maze_top** *(vga_maze_top.sv)*: περιέχει instances από 2 μπλοκ, το **vga_sync** και το **vga_frame** (εξηγούνται παρακάτω), τα οποία συνδέονται όπως φαίνεται στο σχήμα.
- **vga_sync** *(vga_sync.sv)*: Το μπλοκ αυτό είναι υπεύθυνο για την υλοποίηση του χρονισμού του πρωτοκόλλου VGA για ανάλυση 640 x 480 και πρέπει να δημιουργεί 2 βασικά σήματα: (1) το HSYNC (horizontal synchronization – οριζόντιος συγχρονισμός) και (2) το VSYNC (vertical synchronization – κατακόρυφος συγχρονισμός). Επίσης δημιουργεί και 3 σήματα προς το μπλοκ **vga_frame** που δίνουν τις τρέχουσες συντεταγμένες των pixel που πρέπει να εμφανιστούν στην οθόνη. Το σήμα *o_col* δίνει τη τρέχουσα τιμή της στήλης (column), το σήμα *o_row* δίνει την τρέχουσα τιμή της γραμμής (row) και το σήμα *o_pix_valid* σηματοδοτεί αν οι τιμές αυτές (*o_col* και *o_row*) είναι έγκυρες (valid). Λεπτομέρειες για το τι θα πρέπει να υλοποιήσετε για αυτό το μπλοκ περιγράφονται παρακάτω.
- **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 και σας δίνεται έτοιμο. Το μπλοκ δέχεται σαν είσοδο τις συντεταγμένες του pixel μέσω των εισόδων *i_col* για τη γραμμή και *i_row* για τη στήλη και το σήμα *i_pix_valid* που δείχνει αν το pixel είναι έγκυρο. Με βάση τις συντεταγμένες παράγει στον «επόμενο κύκλο» τιμές για τα τρία χρώματα έτσι ώστε να εμφανίζονται στην οθόνη εναλλάξ κόκκινες, πράσινες, μπλε και μαύρες μπάρες που έχουν ύψος 16 γραμμές. Στην πλακέτα του εργαστηρίου το κάθε χρώμα είναι 4-bit οπότε μπορούμε να δημιουργήσουμε 2^4 x 2^4 x 2^4 = 4096 διαφορετικά χρώματα. Η τιμή RGB (15, 0, 0) δημιουργεί κόκκινο χρώμα, η τιμή RGB (0, 15, 0) πράσινο χρώμα, η τιμή RGB (0, 0, 15) μπλε χρώμα και η τιμή RGB (0, 0, 0) μαύρο χρώμα. Η τιμή RGB (15, 15, 15) δημιουργεί άσπρο χρώμα και υπάρχουν όλοι οι υπόλοιποι συνδυασμοί που δημιουργούν αποχρώσεις π.χ. το RGB (2, 4, 1) δημιουργεί μια απόχρωση του πράσινου.
- **vga_tb** *(vga_tb.sv)*: ένα testbench *για προσομοίωση* που δημιουργεί το ρολόι (25 MHz – 40 ns) και το reset. 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 αυτό που παράγει ο δικός σας κώδικάς με το reference output για να εντοπίσετε λάθη κατά την

προσομοίωση. Το format είναι συμβατό με τον VGA Simulator και είναι πολύ απλό. Σε κάθε γραμμή περιέχει το χρόνο σε ns ακολουθούμενο από τις τιμές των σημάτων hsync (1-bit), vsync (1-bit), red (4-bits), green (4-bits), blue (4-bits).

- **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 την Άσκηση 2.1:

Πρέπει να υλοποιήσετε το μπλοκ **vga_sync** του οποίου ένα άδειο module σας δίνεται. Με βάση τις προδιαγραφές του χρονισμού VGA 640 x 480 που παρουσιάστηκαν παραπάνω σας δίνεται το παρακάτω σχηματικό με πύλες που υλοποιεί τις προδιαγραφές. Περιλαμβάνει 2 μετρητές τον hcnt και τον vcnt που μετρούν τις στήλες και τις γραμμές αντίστοιχα. Μετά τους μετρητές υπάρχουν set-clear flip-flops για να δημιουργηθούν κατάλληλα οι παλμοί HSYNC και VSYNC. Επίσης δημιουργούνται τα σήματα *o_col*, *o_row* και *o_pix_valid* που δέχεται το μπλόκ **vga_frame** (σας δίνεται έτοιμο). Για τον παλμό HSYNC έχει προστεθεί ένας κύκλος καθυστέρηση έτσι ώστε το μπλοκ **vga_frame** να βγάλει την έξοδό του (pixel) συγχρονισμένα με το μπλοκ **vga_sync** στον «επόμενο κύκλο».



Θα πρέπει να υλοποιήσετε σε SystemVerilog RTL το κύκλωμα για το μπλοκ **vga_sync**. Θα πρέπει να προσομοιώσετε και να επαληθεύσετε με το έτοιμο testbench και τα reference outputs τον κώδικά σας. Θα πρέπει να μπορείτε να δείτε την έξοδο που θα έβγαζε το κύκλωμα με τον προσομοιωτή VGA Simulator. Μη ξεχάσετε να βάλετε reset στους καταχωρητές!

Άσκηση 2.2

Για την *Άσκηση 2.2* θα πρέπει να σχεδιάσετε σε SystemVerilog έναν 64-bit barrel shifter (hw2_2_barrel.sv) που κάνει περιστροφή (κυκλικό shift) σε επίπεδο byte (8-bit) δεξιά ή αριστερά με τις παρακάτω προδιαγραφές:

- Λειτουργία με ρολόι 100MHz (clk100) και σύγχρονο active-low reset (rst_n)
- Πόρτα εισόδου 64-bits για την φόρτωση τιμής στο barrel shifter (i_value)
- Πόρτα εισόδου 1-bit για τη σηματοδότηση φόρτωσης (i_load)
- Πόρτα εισόδου 1-bit για τη σηματοδότηση περιστροφής αριστερά (i_rotate_left)
- Πόρτα εισόδου 1-bit για τη σηματοδότηση περιστροφής δεξιά (i_rotate_right)
- Πόρτα εξόδου 64-bits για την έξοδο του barrel shifter (o_barrel_value)

Ο barrel shifter κάθε φορά που έρχεται το σήμα i_load τότε φορτώνει την τιμή i_value σε έναν εσωτερικό καταχωρητή. Όταν έρθει το σήμα i_rotate_left/i_rotate_right τότε κάνει αριστερή/δεξιά περιστροφή κατά K bytes στο περιεχόμενο του καταχωρητή (το K είναι σταθερά) και δείχνει το αποτέλεσμα στην έξοδο o_barrel_value στον επόμενο κύκλο. Η έξοδος o_barrel_value πρέπει να παραμένει σταθερή στην προηγούμενη τιμή σε περίπτωση που δεν έρχεται σήμα που δεν αλλάζει την κατάσταση του barrel shifter (π.χ. δεν υπάρχει νέο i_load ή i_rotate_left ή i_rotate_right) ή όταν υπάρχει ταυτόχρονα i_rotate_left και i_rotate_right. Σαν K χρησιμοποιήστε το πηλίκο της ακέραιας διαίρεσης (floor) του τελευταίου ψηφίου του ΑΜ με το 2 και μετά προσθέστε 1 (πχ αν το ΑΜ σας είναι 1037 τότε 7/2=3.5 άρα K = 3+1 = 4).

Παραδώστε τον κώδικα του hw2_2_barrel.sv μαζί με ένα κατάλληλο testbench (hw2_2_tb.sv) που να δοκιμάζετε τουλάχιστον 3 διαφορετικά σενάρια φόρτωσης και περιστροφής σε διαδοχικές χρονικές στιγμές. **Επίσης σχεδιάστε και παραδώστε ένα σχήμα με το κύκλωμα του hw2_2_barrel (hw2_2_barrel_circuit σε μορφή png/jpg/pdf)** που υλοποιήσατε δείχνοντας τα βασικά ψηφιακά στοιχεία όπως πύλες, καταχωρητές, πολυπλέκτες κτλ. (όπως δείχνουμε στις διαλέξεις).

Τι πρέπει να παραδώσετε:

Μέχρι τη ημερομηνία της προθεσμίας πρέπει να παραδώστε σε ένα zip/tar.gz αρχείο (e.g. hw2.zip ή hw2.tar.gz) τα παρακάτω:

1. Τον SystemVerilog RTL κώδικα του μπλοκ **vga_sync**
2. Τον SystemVerilog κώδικα **hw2_2_barrel.sv**
3. Το SystemVerilog testbench **hw2_2_tb.sv**
4. Το σχήμα του **hw2_2_barrel_circuit**

Στείλτε το **hw2.zip** σας με e-mail στο hy220@csd.uoc.gr με τίτλο: *"HW2 – Ονοματεπώνυμο – ΑΜ"*.

Οι κώδικες θα ελέγχονται για αντιγραφές με ειδικό λογισμικό!