Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Πανεπιστήμιο Θεσσαλίας

ECE333 - Εργαστήριο Ψηφιακών Συστημάτων

Εργαστηριακή Εργασία 3η

Περιεχόμενα

1.Τίτλος

Όνομα: Τσελεπή Ελένη

ΑΕΜ:03272

Ημερομηνία:21/11 - 18/12

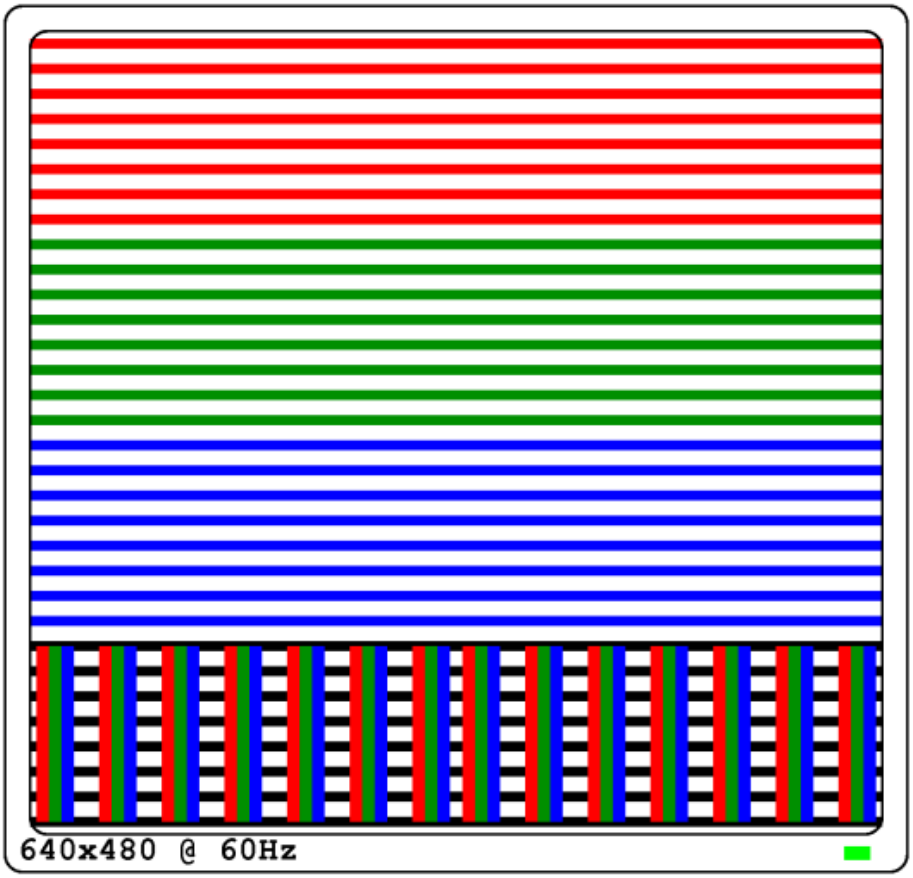
Τίτλος εργασίας: Εργαστηριακή Εργασία 3η, Υλοποίηση Ελεγκτή VGA

2.Περίληψη

Το παρακάτω κείμενο αποτελεί την εργαστηριακή αναφορά για την 3η εργαστηριακή εργασία όπου στόχος της αναφοράς είναι η σαφής περιγραφή της διαδικασίας σχεδίασης, επαλήθευσης, δοκιμής και τελικής υλοποίησης του κυκλώματος. Παρακάτω αναφέρονται αναλυτικά όλα τα μέρη υλοποίησης της εργασίας καθώς και στιγμιότυπα από τις κυματομορφές στο vivado.

3.Εισαγωγή

Ο στόχος της 3ης εργαστηριακής εργασίας ήταν η υλοποίηση ενός Ελεγκτή/Οδηγού θύρας οθόνης VGA (Video Graphics Array), για να επιτευχθεί η οδήγηση μιας συμβατικής οθόνης και η εμφάνιση εικόνας σε αυτήν. Η εικόνα ορίζεται σε μια Video RAM(VRAM) με τη χρήση εσωτερικής RAM των FPGA(Block RAM – BRAM).



Η προτεινομένη εικόνα ελέγχου

Οι προδιαγραφές οδήγησης της οθόνης είναι :

- Ανάλυση 640 x 480 pixels
- Ρυθμός ανανέωσης 60Hz.

4. Μέρος Α - Υλοποίηση VRAM

Η VRAM αποτελεί κεντρικό τμήμα του Ελεγκτή VGA. Το μέγεθος της είναι συνάρτηση της ανάλυσης μιας και αυτή ορίζει τον αριθμό των εμφανιζόμενων pixels και του αριθμού των χρωμάτων ανα pixel . Έτσι για τρία χρώματα που επιτρέπει η FPGA ανά pixel απαιτούνται 3 bits άρα η συνολική μνήμη που χρειάζεται είναι

640 *480*3=115.2K Bytes. Για την υλοποίηση της VRAM θα χρησιμοποιήσουμε την εσωτερική μνήμη της FPGA (Block RAM – BRAM). Λόγω λοιπόν , περιορισμών της

FPGA σε τόσο μεγάλη προσπελάσιμη μνήμη BRAM (115.2 Kbytes) το μέγεθος της VRAM, που θα υλοποιηθεί με BRAM θα πρέπει να είναι το 1/5 σε κάθε διάσταση, δηλ. 128 × 96, και να υποστηρίζει οκτώ χρώματα. Έτσι τελικά χρησιμοποιήσαμε **τρεις** BRAM μια για κάθε χρώμα και μεγέθους 16K x 1 για να υλοποιήσουμε την
VRAM 128 x 96.

Dataflow Μέρους Α

Η είσοδος της VRAM είναι η διεύθυνση(14bit 2^14= 16.384addresses για τις 12.288 θέσεις) ενός pixel, ενώ η έξοδος της παράγει τις τιμές των χρωμάτων του συγκεκριμένου pixel. Κάθε BRAM ενός χρώματος έχει μέγεθος 12.288 bits (128 * 96 =12.288 bits) . Η αρχικοποίηση των BRAM έγινε μέσω παραμέτρων .INIT . Η αρχικοποίηση γίνεται ανά 256 bits επομένως μια αρχικοποίηση είναι 2 γραμμές των 128 bit . Θέλουμε να υλοποιήσουμε 96 γραμμές των 128 bit άρα χρειαζόμαστε 48 init για να αρχικοποιήσουμε κάθε μνήμη BRAM. Το μοτίβο που ακολούθησα για την απεικόνιση της εικόνας είναι 2 άσπρες γραμμές – 2 κόκκινες ή πράσινες ή Μπλε – 2 άσπρες γραμμές κτλπ για το πρώτο μέρος με τις εναλλάξ οριζόντιες γραμμές με κενά. Το πρώτο μέρος διαρκεί 24 γραμμές για κάθε χρώμα άρα συνολικά 72 γραμμές (96 συνολικά γραμμές / 4 =24 γραμμές για κάθε σκέλος) .Ενώ για το δεύτερο μέρος με τις κατακόρυφες γραμμές των τριών χρωμάτων σε φόντο οριζόντιων μαύρων θεώρησα τις 2 πρώτες γραμμές άσπρες ακολουθεί 1 μαύρη γραμμή και 1 γραμμή με πολύχρωμο μοτίβο που θα αναλυθεί παρακάτω και μετά ακολουθούν 2 πολύχρωμες γραμμές με άσπρο φόντο 2 πολύχρωμες με μαύρο φόντο κτλπ μέχρι την τελευταία γραμμή που την θεωρώ μαύρη .

<u>Πολύχρωμο μοτίβο</u>

Για το πολύχρωμο μοτίβο σκέφτηκα κάθε γραμμή να έχει με 2bit μαύρο 2bit κόκκινο 2 bit πράσινο 2 bit μπλε έτσι ώστε 128 bit ανά γραμμή / 8 bit του μοτίβου να έχω 16 πολύχρωμες στήλες όπως είναι η προτεινομένη εικόνα . Το ίδιο μοτίβο ακολουθώ και για τις άσπρες γραμμές μόνο που τώρα είναι 2 bit λευκό 2 bit κόκκινο 2 bit πράσινο και 2 bit μπλε .

Επειδή στην αρχικοποίηση έχουμε 64 ψηφία κάθε ψηφίο είναι 4 bit (64*4=256 bits).Επιπλέον γνωρίζουμε πως για την απεικόνιση του λευκού χρειάζονται και τα 3 χρώματα και επίσης δεν πρέπει να παραλείψουμε πως η μνήμη είναι ορισμένη ως [255:0] επομένως τα least significant bit βρίσκονται αριστερά άρα η αρχικοποίηση για κάθε BRAM έχει ως εξής .

BRAM-RED

-Για κάθε άσπρη γραμμή βάζουμε άσσους (f=4'b1111)

-Για κάθε κόκκινη γραμμή βάζουμε άσσους (f=4'b1111)

-Στο πολύχρωμο μέρος για το μαύρο φόντο έχουμε (**00 00 11 00** = 0 **c**)

-Στο πολύχρωμο μέρος για το άσπρο φόντο έχουμε (**00 00 11 11** = **0 f**)

BRAM-GREEN

-Για κάθε άσπρη γραμμή βάζουμε άσσους (f=4'b1111)

-Για κάθε πράσινη γραμμή βάζουμε άσσους (f=4'b1111)

-Στο πολύχρωμο μέρος για το μαύρο φόντο έχουμε (**00 11 00 00** =**3 0**)

-Στο πολύχρωμο μέρος για το άσπρο φόντο έχουμε (**00 11 00 11** = **3 3**)

BRAM-BLUE

-Για κάθε άσπρη γραμμή βάζουμε άσσους (f=4'b1111)

-Για κάθε μπλε γραμμή βάζουμε άσσους (f=4'b1111)

-Στο πολύχρωμο μέρος για το μαύρο φόντο έχουμε (**11 00 00 00** = **c 0**)

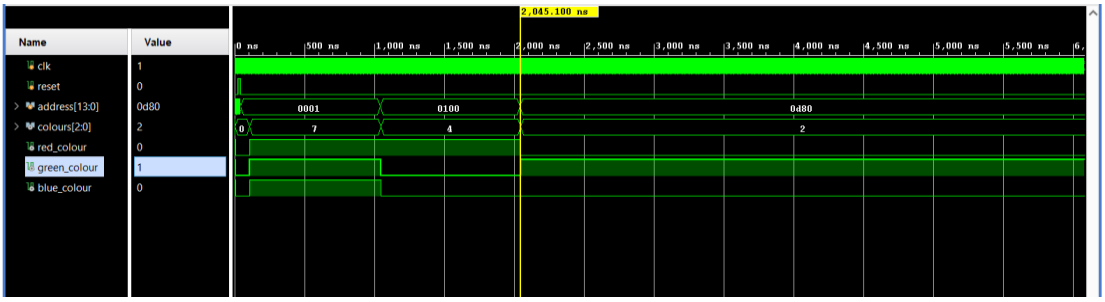
-Στο πολύχρωμο μέρος για το άσπρο φόντο έχουμε (**11 00 00 11** = **c 3**)

Για την εσωτερική μνήμη(BRAM) χρησιμοποίησα το αρχείο BRAM_SINGLE_MACRO το οποίο το βρήκα στο vivado->tools->Language Templates-> Device macro instantiation-> Artix-7-> Single Port RAM . Εκεί σε κάθε init έδωσα τις τιμές που προαναφέρθηκαν και επιπλέον έθεσα WRITE_WIDTH=1, READ_WIDTH=1, WE=0(χρειάζεται μόνο να διαβάζει και όχι να γραφεί στην μνήμη) και EN=1(enable για την έξοδο) ώστε κάθε φορά η BRAM ενός χρώματος να διαβάζει και να έχει ως έξοδο 1 bit χρώματος . Δηλαδή συνολικά η VRAM παράγει τις τιμές των χρωμάτων 3 bit (1 για κάθε χρώμα) ανά pixel.

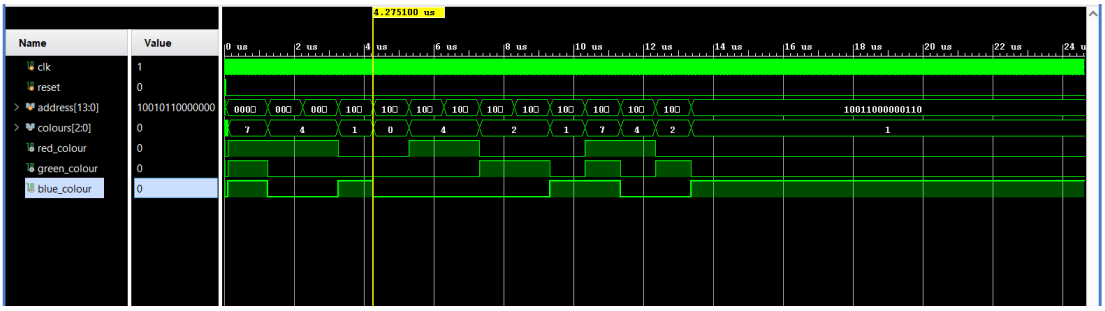
Σημείωση: Τις παραμέτρους για τα ports στη BRAM τις βρήκα στην αρχή αλλάζοντας τιμές και βλέποντας τις κυματομορφές αλλά και μελετώντας το manual από την σελίδα 40 και μετά 7Series Memory Resources

Τα αποτελέσματα από το testbench για το Μέρος Α

Για τον έλεγχο του Α Μέρους δίνοντας διαφορετικές τιμές στην διεύθυνση address έβλεπα τις διαφορετικές τιμές των εξόδων των BRAMs.Συγκεκριμένα όπως φαίνεται και στο παρακάτω στιγμιότυπο για address=1 δηλαδή το πρώτο bit που περιμένουμε να είναι λευκό η έξοδος χρωμάτων είναι 3'b111 ενώ για διεύθυνση 256 που είναι η τρίτη γραμμή που είναι κόκκινη η έξοδος των χρωμάτων είναι 3'b100.Τέλος για την διεύθυνση 3456 1 πρώτη πράσινη γραμμή (24 γραμμές με οριζόντιες άσπρες και κόκκινες +2 γραμμές ασπρες+1 γραμμή πράσινη = 27 *128 =3456) έχουμε έξοδο 3'b010 όπου είναι αναμμένο μόνο το πράσινο χρώμα .



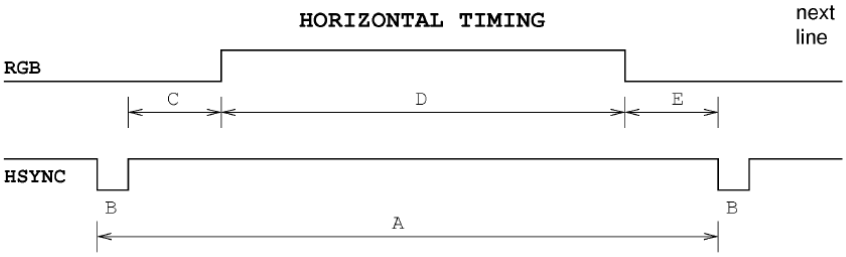
Στιγμιότυπο από το testbench του Α Μέρους, στο οποίο φαίνονται πως για διαφορετική address αλλάζει η έξοδος των χρωμάτων .



Στιγμιότυπο από το testbench του Α Μέρους , στο οποίο φαίνονται πως διεύθυνση address=75 που είναι πρώτη μαύρη πολύχρωμη γραμμή άλλα και για 76 την πρώτη πολύχρωμη άσπρη γραμμή τα χρώματα είναι τα αναμενόμενα με μοτίβο μαύρο κόκκινο πράσινο και μπλε .

5. Μέρος Β - Υλοποίηση HSYNC και Οριζόντιου Μετρητή Pixel

Για το Μέρος Β υλοποιήσαμε το σήμα HSYNC το οποίο χρονίζει την οριζόντια συντεταγμένη της οθόνης για ανάλυση 640 x 480 και Ρυθμό ανανέωσης 60Hz.



Οριζόντιος Συγχρονισμός του VGA Οδηγού

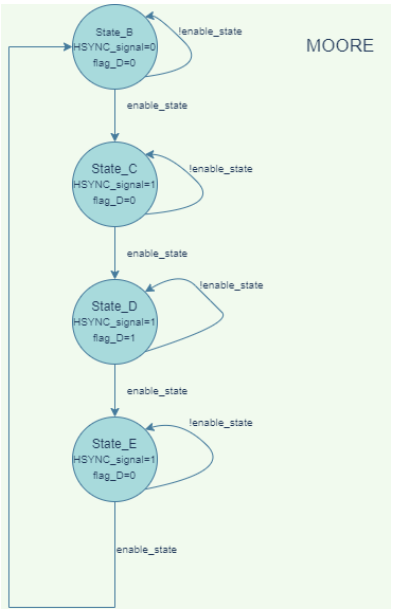
Το σήμα HSYNC υλοποιήθηκε ως μια μηχανή πεπερασμένων καταστάσεων με κάθε κατάσταση της FSM να είναι ένα από τα διαστήματα (B,C,D,E) που φαίνονται στην παραπάνω εικόνα . Ο χρονισμός των διαστημάτων B, C, D, E εξαρτάται από την ανάλυση και τον Ρυθμό Ανανέωσης γι αυτό μας δόθηκε πίνακας με τις κατάλληλες τιμές των διαστημάτων για ανάλυση 640×480 και Ρυθμό Ανανέωσης 60Hz τον οποίο παραθέτω παρακάτω .

Διάστημα	Περιγραφή	Περιγραφή στα Αγγλικά	Τιμή
A	Χρόνος Σάρωσης Γραμμής	Scanline Time	32 μsec
B	Πλάτος Παλμού HSYNC	HSYNC Pulse Width	3.84 μsec
C	Πίσω Όψη	Back Porch	1.92 μsec
D	Χρόνος Απεικόνισης	Display Time	25.6 μsec
E	Μπροστινή Όψη	Front Porch	0.640 μsec
O	Συνολικός Χρόνος Εικόνας	Total Frame Time	16.67 msec
P	Πλάτος Παλμού VSYNC	VSYNC Pulse Width	64 μsec
Q	Πίσω Όψη	Back Porch	928 μsec
R	Χρόνος Ενεργής Απεικόνισης	Active Video Time	15.36 msec
S	Μπροστινή Όψη	Front Porch	320 μsec

Πίνακας χρονισμού διαστημάτων HSYNC , VSYNC

Οι συγκεκριμένοι χρόνοι έγιναν κύκλοι ρολογιού διαιρώντας με την περίοδο του ρολογιού το οποίο είναι στα 10ns.

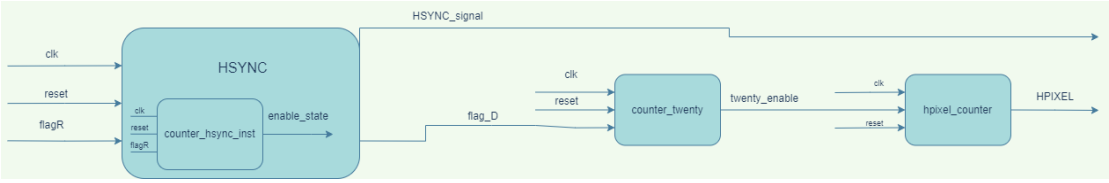
Για να πέτυχω αυτούς τους χρονισμούς χρησιμοποίησα ένα counter(counter_hsync) ο οποίος μετράει τις τιμές από **0-3199** (3200 κύκλοι ρολογιού είναι ολόκληρο το σήμα HSYNC A=3200 κύκλοι=B+C+D+E) . Όταν ο μετρητής φτάσει στις τιμές των B-1, B+C-1, B+C+D - 1, B+C+D+E - 1 ο μετρητής έχει ως έξοδο ένα σήμα enable_state το οποίο είναι η είσοδος στην FSM και καθορίζει την επόμενη κατάσταση . Αξίζει να αναφερθεί πως η τιμή -1 που προανέφερα είναι ώστε το CurrentState που θα γίνει έναν κύκλο αργότερα να είναι σωστά συγχρονισμένο με κάθε διάστημα , και πως τα B,C,D,E έχουν γίνει define “*times_hsync.vh”.*



Η **FSM** του Μέρους Β με 4 καταστάσεις η οποία είναι μια MOORE FSM διότι η έξοδος της δεν καθορίζεται από την είσοδο αλλά μόνο από την τρέχουσα κατάσταση . Σε κάθε κατάσταση αναφέρεται και η έξοδος στο διπλανό σχήμα . Αν έρθει το σήμα enable_state από τον μετρητή η FSM πηγαίνει σε επόμενη κατάσταση αλλιώς παραμένει στην ίδια κατάσταση . Το HSYNC είναι στο 0 μόνο στην κατάσταση Β .

Όπως προαναφέραμε στο Μέρος Α η μνήμη BRAM είναι το 1/5 των διαστάσεων γι' αυτό χρειάζεται η VRAM να μεγεθύνεται και να γεμίζει οθόνη ανάλυσης 640X480. Γνωρίζουμε πως η εικόνα είναι ενεργή στα διαστήματα D οριζοντίως, και R κατακορύφως και πως για το HSYNC το διάστημα D διαρκεί 2560 κύκλους /640 = 4 κύκλοι για κάθε στοιχείο της μνήμης . Επειδή όμως η μνήμη μας είναι στο 1/5 των διαστάσεων χρειάζεται κάθε στοιχείο της μνήμης να μένει ανά 4*5=20 κύκλους. Άρα κάθε θέση της μνήμης θα εκτυπώνεται 5 φορές οριζόντια και κάθε γραμμή 5 φορές κάθετα για να πέτυχουμε την μεγέθυνση . Για την υλοποίηση αυτής της σκέψης χρησιμοποίησα έναν μετρητή (counter_ twenty) ο οποίος μετράει όταν η FSM είναι στην κατάσταση Display Time D από το 0-19 και ανά 20 κύκλους στέλνει ένα σήμα (twenty_enable) το οποίο αυξάνει τον μετρητή pixels HPIXEL ο οποίος επιδεικνύει το εκάστοτε ενεργό pixel της VRAM. ΄Ετσι, η διεύθυνση pixel της VGA, προκύπτει με τη συνένωση των κατακόρυφων και οριζόντιων μετρητών, δηλ. address = {VPIXEL, HPIXEL} , όπου καθένα από τα VPIXEL HPIXEL είναι 7bit (HPIXEL μέχρι 127 = 2^7 και VPIXEL μέχρι 95 άρα 7 bit).

Όλα αυτά που προαναφέρθηκαν παρουσιάζονται και στο παρακάτω διάγραμμα ροής

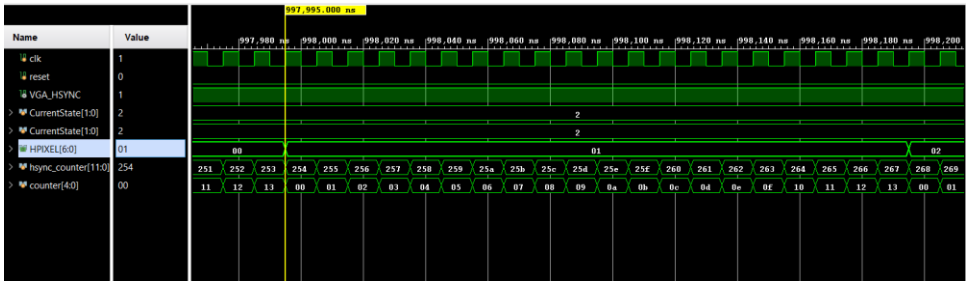


Dataflow Μέρους Β

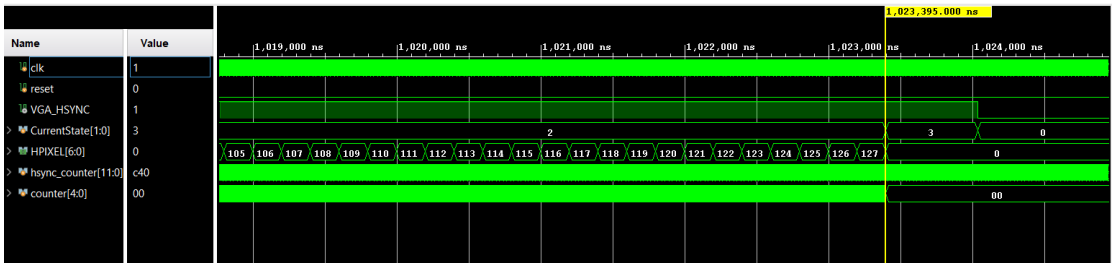
Σημείωση: Ο μετρητής HSYNC ξεκινάει να μετράει όταν το VSYNC βρίσκεται στην κατάσταση Active Video Time γι αυτό έχει ως είσοδο το flagR που δείχνει πότε το VSYNC βρίσκεται στην κατάσταση R.

Τα αποτελέσματα από το testbench για το Μέρος Β

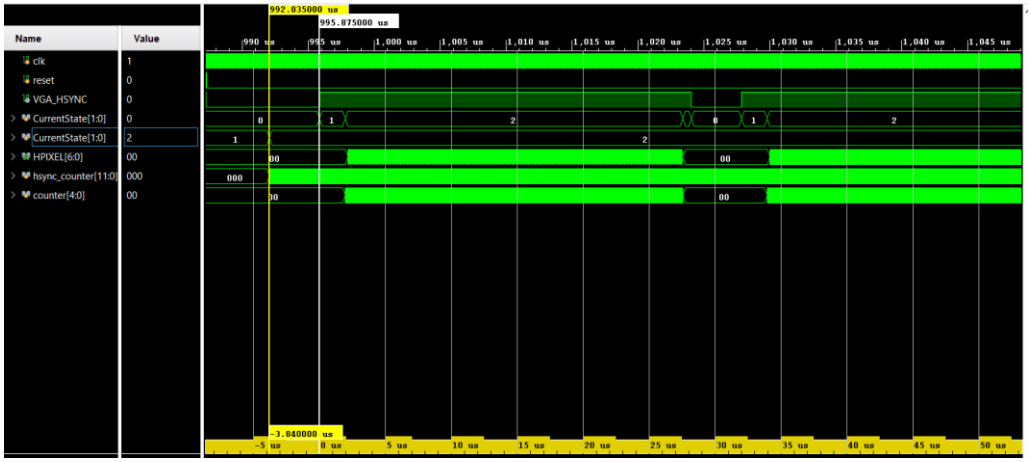
Για την επαλήθευση του Β μέρους έλεγξα τους χρόνους κάθε κατάστασης και όπως θα δείτε και από τα στιγμιότυπα επαληθεύονται οι χρόνοι από τον πίνακα . Επιπλέον έλεγξα ότι ο counter_hsync ξεκινάει μόνο όταν το VSYNC είναι στην κατάσταση R και πως το HPIXEL παίρνει τιμές μόνο στην κατάσταση D αλλά και πως αυξάνεται ανά 20 κύκλους ρολογιού όπως εξήγησα παραπάνω .



Στο συγκεκριμένο στιγμιότυπο παρατηρούμε πως το HPIXEL αλλάζει ανά 20 κύκλους ρολογιού.

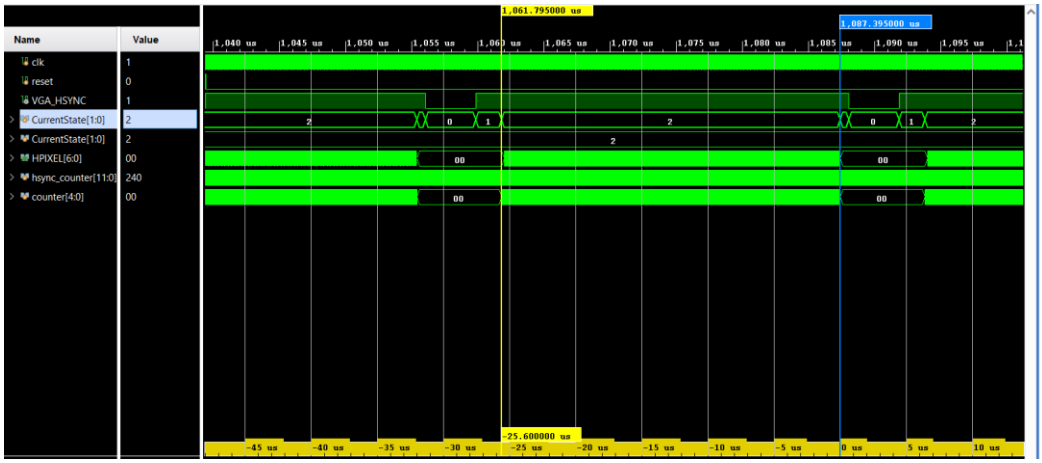


Στο συγκεκριμένο στιγμιότυπο παρατηρούμε πως όταν το HSYNC είναι στο τέλος της κατάστασης D το HPIXEL έχει την τιμή 127(το τελευταίο bit κάθε γραμμής).

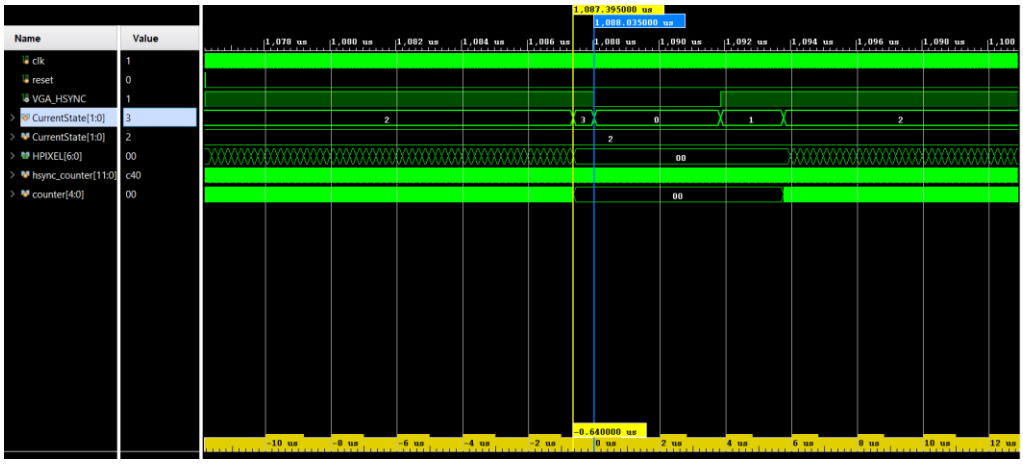


Στο συγκεκριμένο στιγμιότυπο παρατηρούμε πως το HSYNC ξεκινάει μαζί με το 2 state του VSYNC την κατάσταση R , πως το HPIXEL αυξάνεται μόνο μέσα στην κατάσταση D (3η κατάσταση με αριθμό 2 στο στιγμιότυπο) αλλά και πως η διάρκεια της πρώτης κατάστασης του HSYNC , η κατάσταση B είναι 3,84μs όπως περιμέναμε από τον πίνακα χρονισμού .

Στο συγκεκριμένο στιγμιότυπο παρατηρούμε πως η 2η κατάσταση του HSYNC , η κατάσταση C διαρκεί 1.92μsec όπως περιμέναμε από τον πίνακα χρονισμού .



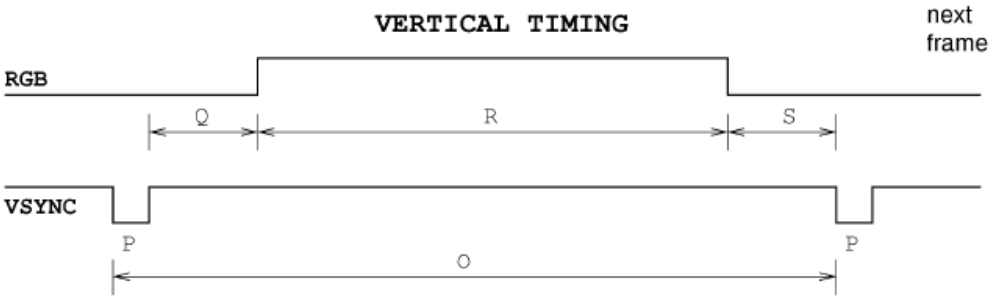
Στο συγκεκριμένο στιγμιότυπο παρατηρούμε πως η 3η κατάσταση του HSYNC , η κατάσταση D διαρκεί 25.6μsec όπως περιμέναμε από τον πίνακα χρονισμού .



Στο συγκεκριμένο στιγμιότυπο παρατηρούμε πως η 4η κατάσταση του HSYNC , η κατάσταση E διαρκεί 0.64μsec όπως περιμέναμε από τον πίνακα χρονισμού .

6. Μέρος Γ - Υλοποίηση VSYNC και Κατακόρυφου Μετρητή Pixel—Ολοκλήρωση του Ελεγκτή/Οδηγού VGA

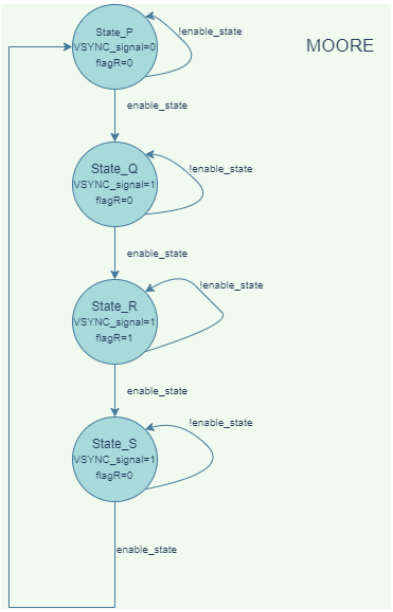
Για το Μέρος Γ υλοποιήσαμε το σήμα VSYNC το οποίο χρονίζει την κατακόρυφη συντεταγμένη της οθόνης για ανάλυση 640 x 480 και Ρυθμό ανανέωσης 60Hz.



Κατακόρυφος Συγχρονισμός του VGA Οδηγού

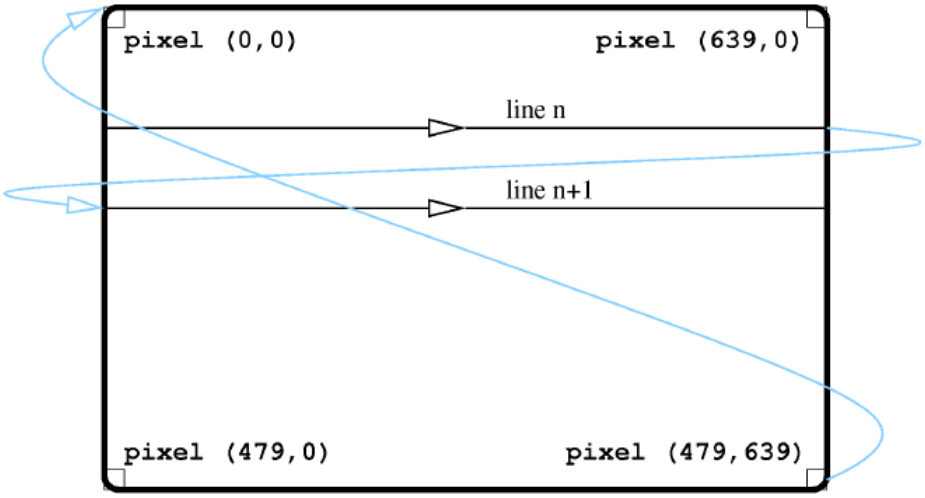
Το σήμα VSYNC εν αντιστοιχία με το Μέρος Β ορίστηκε ως μια μηχανή πεπερασμένων καταστάσεων με κάθε κατάσταση της FSM να είναι ένα από τα διαστήματα (P,Q,R,S) που φαίνονται στην παραπάνω εικόνα. Από τον πίνακα που παρατέθηκε στο Β Μέρος για τα διαστήματα χρονισμού βρήκα τα διαστήματα κάθε κατάστασης και τα υλοποίησα με έναν μετρητή . Συγκεκριμένα, υλοποίησα έναν counter (counter_vsync) ο οποίος μετράει τον χρονισμό ολοκλήρου του σήματος VSYNC δηλαδή από το 0-1.667.199 (1.677.200 κύκλοι ρολογιού είναι όλο το σήμα VSYNC O=P+Q+R+S). Όταν ο μετρητής φτάσει στις τιμές των P-1, P+Q-1, P+Q+R - 1, P+Q+R+S - 1 ο μετρητής έχει ως έξοδο ένα σήμα enable_state το οποίο είναι η είσοδος στην FSM και καθορίζει την επόμενη κατάσταση. Αξίζει να αναφερθεί πως η τιμή -1 που προανέφερα είναι ώστε το CurrentState που θα γίνει έναν κύκλο αργότερα να είναι σωστά συγχρονισμένο με κάθε διάστημα , και πως τα P,Q,R,S έχουν γίνει define *“times_vsync.vh”.*

Για να πέτυχουμε την μεγέθυνση της VRAM για τη κατακόρυφη συνταραγμένη της οθόνης όπως προαναφέρθηκε λίγο και στο Μέρος Β θα πρέπει να εμφανίζουμε την ιδιά γραμμή 5 φορές κατακόρυφα. Γνωρίζουμε πως η εικόνα είναι ενεργή στο διάστημα R κατακορύφως και πως για το VSYNC το διάστημα R διαρκεί 1.536.000 κύκλους /480 = 3200 , που όπως είδαμε και στο μέρος Β αντιστοιχεί σε μια γραμμή. Άρα, για να εμφανίσουμε την ίδια γραμμή 5 φορές κατακόρυφα πρέπει ανά 3200*5=16000 κύκλους ρολογιού να αλλάζουμε το VPIXEL . Για τον λόγο αυτό χρησιμοποίησα έναν counter (counter_five) ο οποίος μετράει όταν η FSM είναι στην κατάσταση Actice Video Time R από το 0-15999 και ανά 16000 κύκλους στέλνει ένα σήμα (five_enable) το οποίο αυξάνει τον μετρητή pixels VPIXEL .



Η **FSM** του Μέρους Γ με 4 καταστάσεις η οποία είναι μια MOORE FSM διότι η έξοδος της δεν καθορίζεται από την είσοδο αλλά μόνο από την τρέχουσα κατάσταση . Σε κάθε κατάσταση αναφέρεται και η έξοδος στο διπλανό σχήμα . Αν έρθει το σήμα enable_state από τον μετρητή η FSM πηγαίνει σε επόμενη κατάσταση αλλιώς παραμένει στην ίδια κατάσταση . Το VSYNC είναι στο 0 μόνο στην κατάσταση P.

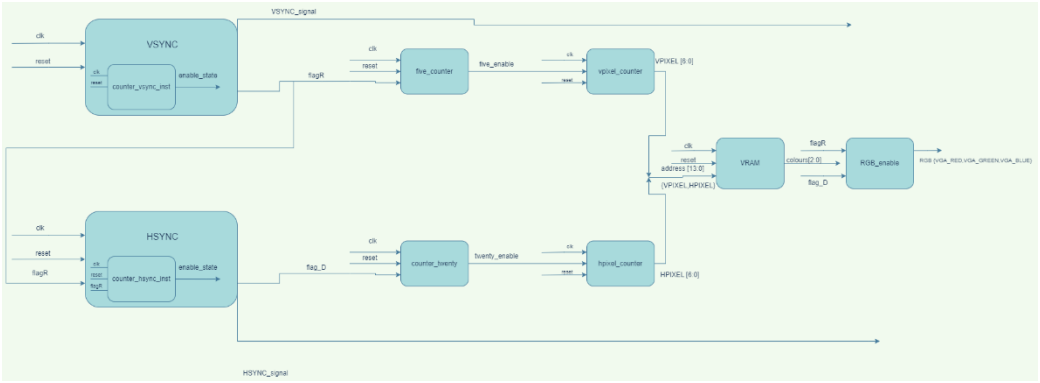
Η κατακόρυφη σάρωση συγχρονίζεται καταλλήλως με την οριζόντια, όπως περιγράφεται στην παρακάτω εικόνα.



Για κάθε γραμμή από τις 96 (VPIXEL 0-95) το HSYNC εμφανίζει κάθε στοιχείο της γραμμής από 0-127 . Αυτές οι τιμές μεγεθύνονται όπως προαναφέρθηκε και στα μέρη Β και Γ .

Το HSYNC ενεργοποιείται με την κατάσταση Active Video Time κατάσταση R ώστε για κάθε γραμμή το HSYNC να αυξάνεται από 0-127 για κάθε pixel της γραμμής και έτσι η διεύθυνση pixel της VGA, προκύπτει με τη συνένωση των κατακόρυφων και οριζόντιων μετρητών, δηλ. address = {VPIXEL, HPIXEL} . Τα χρώματα VGA_RED,VGA_GREEN,VGA_BLUE είναι ενεργοποιημένα μόνο για τα διαστήματα D και R και είναι απενεργοποιημένα στο 0 για οποιαδήποτε άλλη κατάσταση των HSYNC, VSYNC . Η τελευταία παρατήρηση υλοποιήθηκε στο module RGB_enable που ελέγχω σε ποια κατάσταση είναι οι FSMs και αναλόγως ορίζω τα χρώματα για τα VGA_RED,VGA_GREEN,VGA_BLUE.

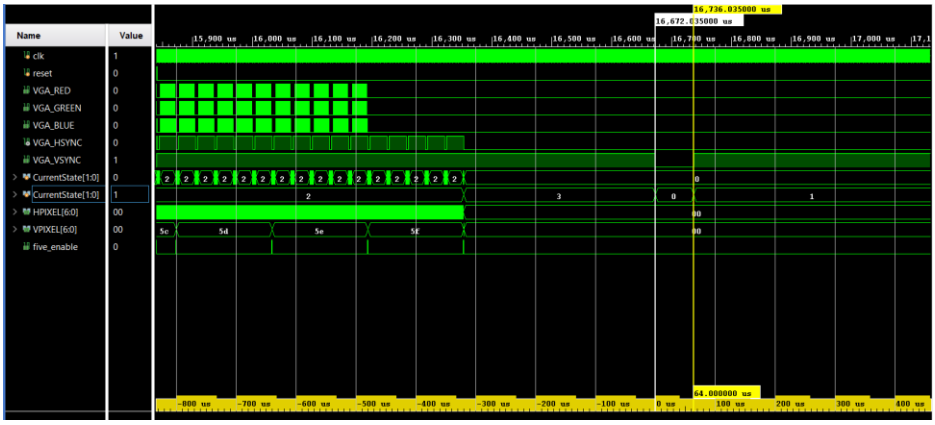
Όλα αυτά που προαναφέρθηκαν παρουσιάζονται και στο παρακάτω διάγραμμα ροής



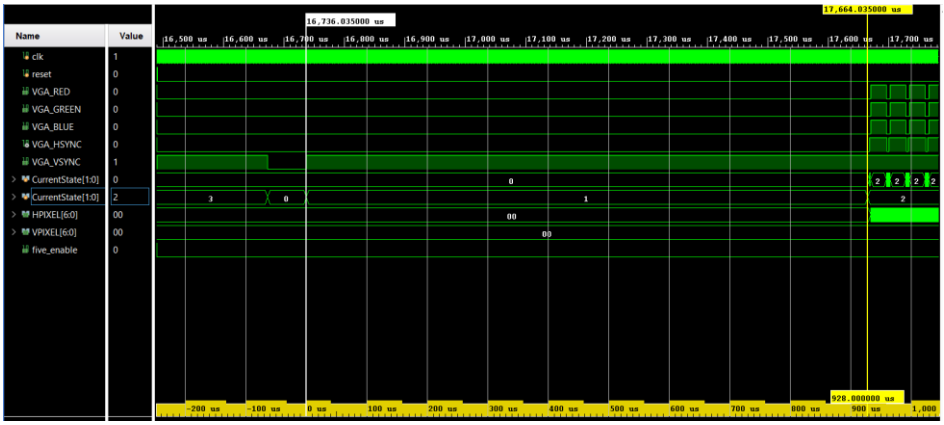
Dataflow Μέρους Γ

<u>Τα αποτελέσματα από το testbench για το Μέρος Γ</u>

Για την επαλήθευση του Γ μέρους έλεγξα τους χρόνους κάθε κατάστασης και όπως θα δείτε και από τα στιγμιότυπα επαληθεύονται οι χρόνοι από τον πίνακα . Επιπλέον έλεγξα ότι το VPIXEL αυξάνεται ανά 16000 κύκλους ρολογιού όπως εξήγησα παραπάνω αλλά και τους συγχρονισμούς με το σήμα HSYNC.

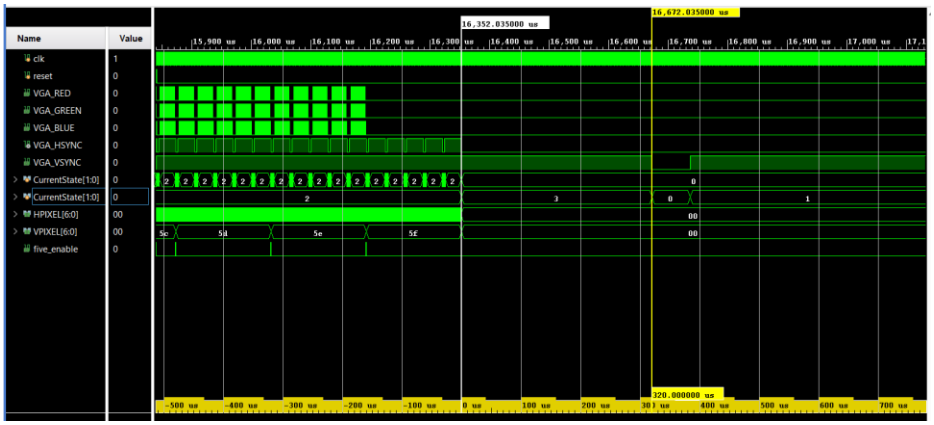


Στο συγκεκριμένο στιγμιότυπο παρατηρούμε πως η 1η κατάσταση του VSYNC , η κατάσταση P διαρκεί 64μsec όπως περιμέναμε από τον πίνακα χρονισμού .

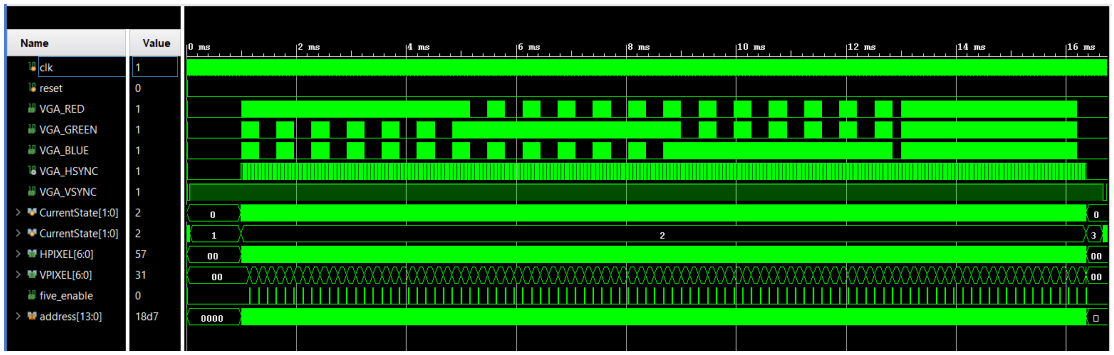


Στο συγκεκριμένο στιγμιότυπο παρατηρούμε πως η 2η κατάσταση του VSYNC ,η κατάσταση Q διαρκεί 928μsec όπως περιμέναμε από τον πίνακα χρονισμού .

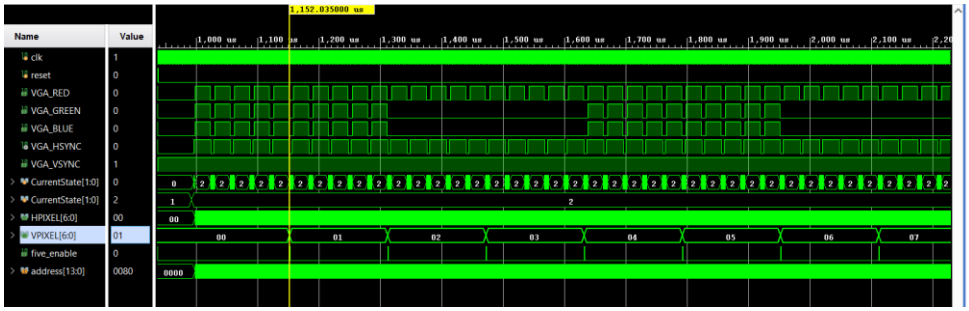
Στο συγκεκριμένο στιγμιότυπο παρατηρούμε πως η 3η κατάσταση του VSYNC , η κατάσταση R διαρκεί 15.36msec όπως περιμέναμε από τον πίνακα χρονισμού .



Στο συγκεκριμένο στιγμιότυπο παρατηρούμε πως η 4η κατάσταση του HSYNC , η κατάσταση S διαρκεί 320μsec όπως περιμέναμε από τον πίνακα χρονισμού .



Στο συγκεκριμένο στιγμιότυπο παρατηρούμε πως το VPIXEL αυξάνεται όταν έρχεται το σήμα five_enable(έχουν περάσει 16000 κύκλοι ρολογιού) και αυξάνεται μόνο στην κατάσταση R. Επίσης παρατηρούμε πως τα χρώματα είναι σωστά καθώς όπως φαίνεται από το στιγμιότυπο υπάρχουν εναλλαγές μεταξύ των κόκκινων/πράσινων/μπλε και λευκών γραμμών αλλά και στο τέλος υπάρχουν πολλές ανάλλαγες χρωμάτων που υποδηλώνουν τις εναλλαγές στο κατακόρυφο τμήμα της εικόνας απεικόνισης. Όλα αυτά φαίνονται καλυτέρα σε μεγέθυνση που θα παραθέσω παρακάτω.

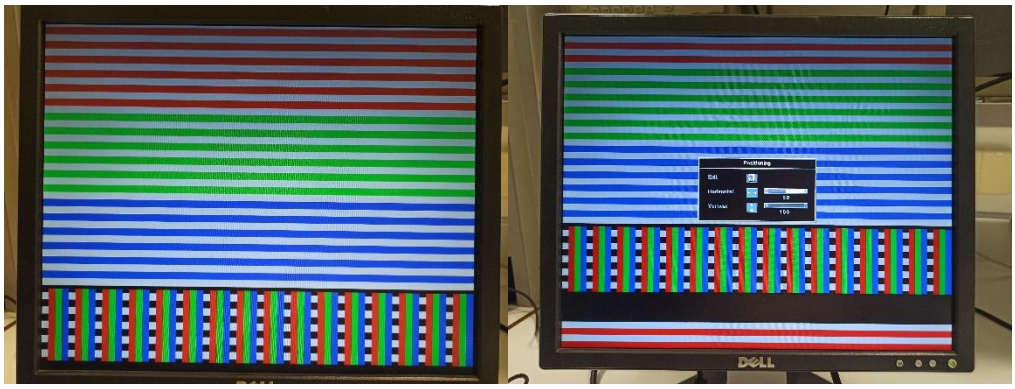


*Στο μεγεθυμένο αυτό στιγμιότυπο φαίνεται ότι ανά 5 γραμμές το VPIXEL αυξάνεται και πως το λευκό χρώμα μένει για 10 γραμμές γιατί είναι 2 λευκές(από την υλοποίηση)*5μεγεθυνσης = 10 όπως αντίστοιχα μετά και για το κόκκινο . Επίσης παρατηρούμε πως το VGA_RED,VGA_GREEN,VGA_BLUE γίνεται 0 για διαστήματα εκτός των D και R .*

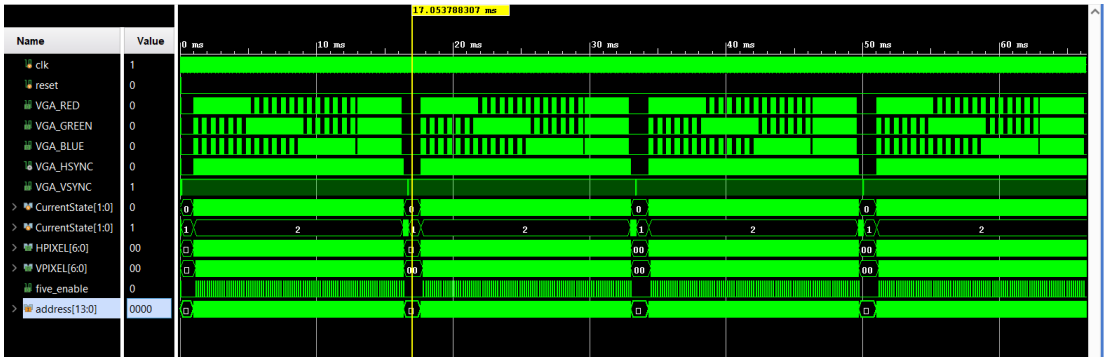
Δυσκολίες που αντιμετώπισα

Κατά τον προγραμματισμό της FPGA χρειάστηκε μόνο μια επανάληψη ώστε να εμφανιστεί η εικόνα στην οθόνη όμως αντιμετώπισα ένα πρόβλημα .Το συγκεκριμένο πρόβλημα ήταν ότι κατά την προβολή της εικόνας δεν εμφανιζόταν η πρώτη από τις 2 λευκές γραμμές και ότι η τελευταία ήταν μαύρη δηλαδή ήταν σβηστή. Στην υλοποίηση μου έχω τις πρώτες 2 γραμμές άσπρες και εμφανιζόταν μόνο η μια. Ενώ δοκιμάσαμε με τους βοηθούς να κάνουμε την οθόνη Auto adjust το πρόβλημα δεν λύθηκε αλλά αν μετακινούσα την εικόνα προς τα κάνω η πρώτη γραμμή μου εμφανιζόταν με 2 άσπρες .

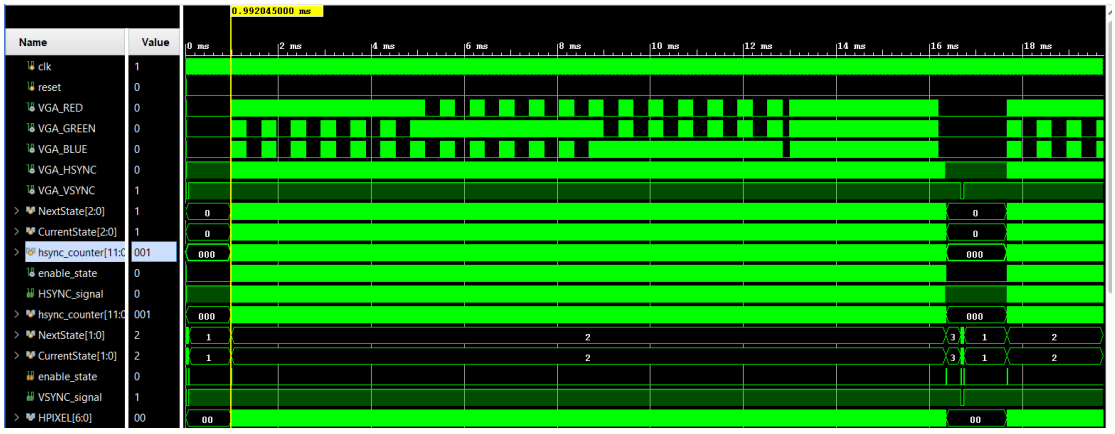
<u>Εικόνες από τα αποτελέσματα</u>



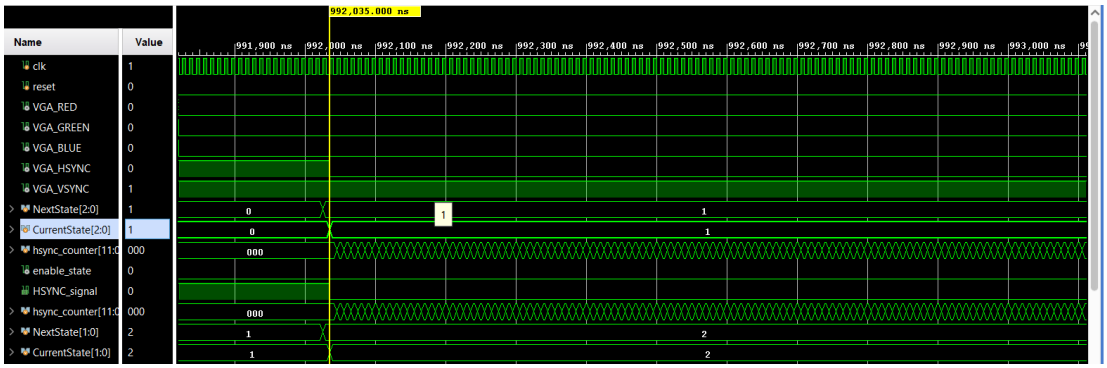
Αρχικά πιστεύαμε ήταν πρόβλημα συγχρονισμού αλλά όπως θα δείτε και στα στιγμιότυπα που θα παραθέσω παρακάτω όλοι οι counters μηδενίζονταν στην αλλαγή κάθε frame . Επίσης δοκίμασα μετά από προτροπή των βοηθών να κάνω και μια άλλη υλοποίηση όπου μέχρι να έρθει το σήμα R του VSYNC το HSYNC να ήταν στο 1 και όχι στο 0 αλλά τότε η οθόνη δεν άναβε καθόλου .



Στο συγκεκριμένο στιγμιότυπο για 4 frames το HPIXEL , VPIXEL και το address όταν έρχεται νέο frame ξεκινάνε πάλι από το 0 .



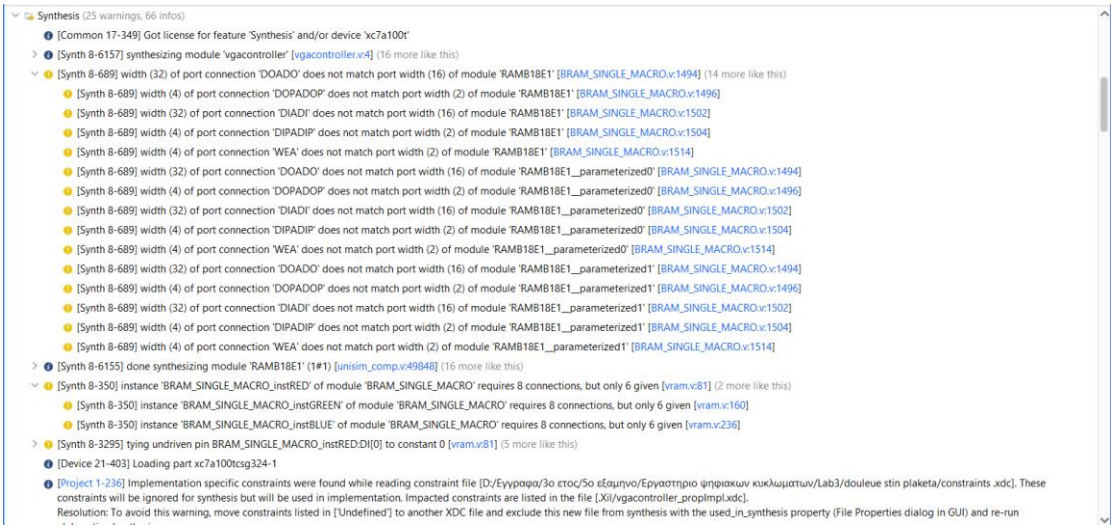
Το συγκεκριμένο στιγμιότυπο είναι από το αρχείο με το HSYNC να βρίσκεται στο 1 μέχρι να έρθει το R . Όπως θα δείτε όλα φαίνονται όπως και στην άλλη υλοποίηση



Εδώ μπορείτε να δείτε σε μεγέθυνση ότι για την υλοποίηση με το HSYNC να είναι στο 1 η κατάσταση R ξεκινάει μαζί με την κατάσταση B του HSYNC άρα είναι και πάλι συγχρονισμένα όμως η οθόνη σε αυτή την περίπτωση δεν άνοιγε καθόλου .

Μετά από όλες αυτές τις προσπάθειες και επικοινωνία με τους βοηθούς δεν μπορέσαμε να καταλάβουμε τι ήταν αυτό που δημιουργούσε το συγκεκριμένο πρόβλημα και υποθέσαμε μήπως ήταν ζήτημα της ιδίας της οθόνης και όχι του κώδικα .

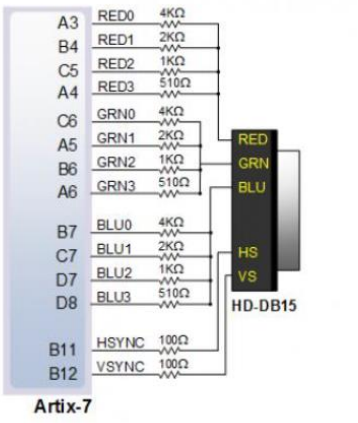
Warnings



Όλα τα warnings που μου εμφανιζόταν είχαν να κάνουν με την μνήμη. Επειδή δεν τα θεώρησα σημαντικά καθώς το instantiation της μνήμης που χρησιμοποίησα δεν είχε αυτές τις μεταβλητές DOADO,DIADI κτλπ τα αγνόησα γιατί δεν επηρέαζαν την λειτουργία του κυκλώματος μου .

Επιλογή constraints

Για την επιλογή των constraints για το .xdc file διάβασα το manual της Nexys nexys-a7_rm.pdf και με βάση την παρακάτω εικόνα συνέδεσα τις εξόδους του vgacontroller (VGA_RED,VGA_GREEN,VGA_BLUE,VGA_HSYNC,VGA_VSYNC) με τα αντίστοιχα πινάκια. Συγκεκριμένα τα πινάκια A3,B4,C5,A4 τα συνέδεσα στο VGA_RED , τα πινάκια C6,A5,B6,A6 τα συνέδεσα στο VGA_GREEN , τα πινάκια B7,C7,D7,D8 τα συνέδεσα στο VGA_BLUE και τέλος τα VGA_HSYNC και VGA_VSYNC τα συνέδεσα στα πινάκια B11 και B12 αντίστοιχα .



8. Συμπεράσματα

Η 3η εργασία ήταν μια αρκετά απαιτητική εργασία αλλά και πολύ ενδιαφέρουσα καθώς έμαθα πως λειτουργεί η VGA και ο συγχρονισμός της με τα σήματα HSYNC , VSYNC. Ο στόχος επιτεύχθηκε και εμφάνισα εικόνα στην οθόνη όμως με το πρόβλημα που προανέφερα. Ήταν μια καλή εξάσκηση με τις μηχανές πεπερασμένων καταστάσεων και με τον συγχρονισμό σημάτων .