


Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Πανεπιστήμιο Θεσσαλίας

3η Εργαστηριακή Εργασία - Υλοποίηση Ελεγκτή Video Graphics Array (VGA) Driver

ECE333 - Εργαστήριο Ψηφιακών Συστημάτων

Διδάσκων: Χ. Σωτηρίου Ημερομηνία: 03/12/2023

Συγγραφέας Εργασίας: Γεώργιος Καπάκος

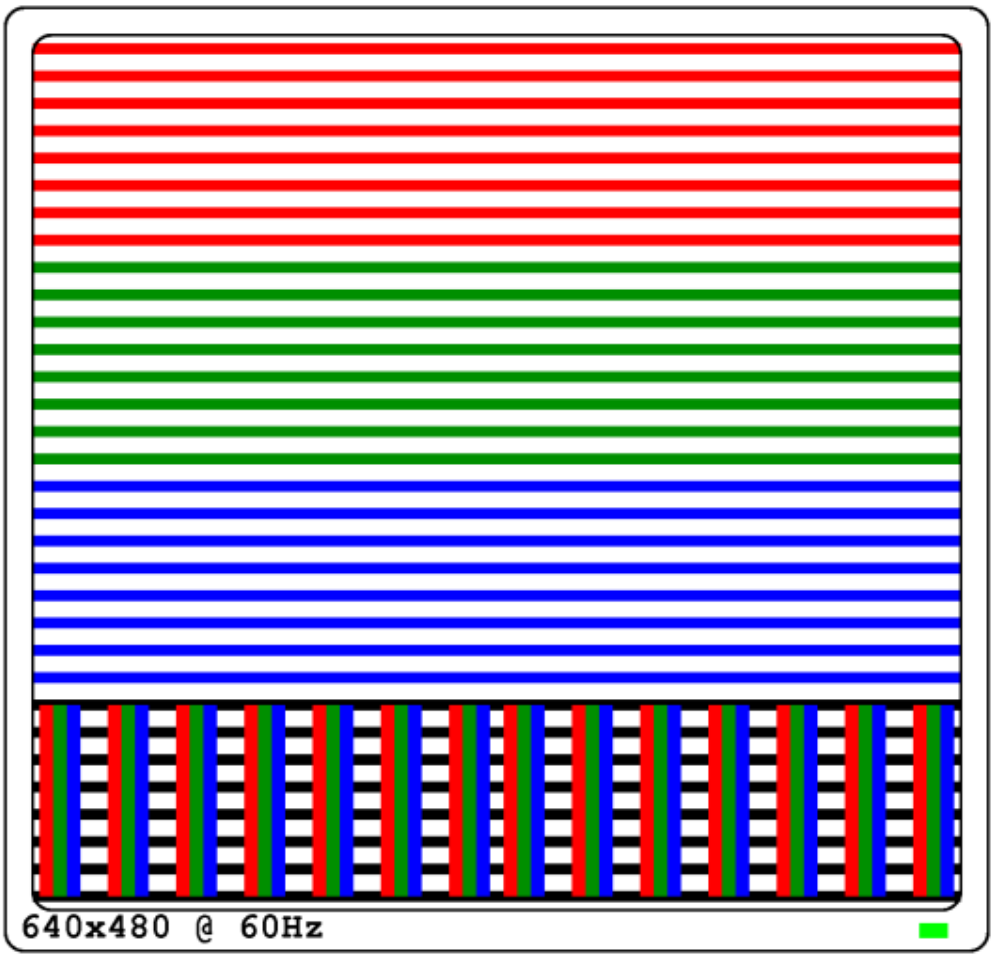
ΑΕΜ: 03165

Περιεχόμενα

ΠΕΡΙΛΗΨΗ:

Στόχος της 3ης εργαστηριακής εργασίας είναι η υλοποίηση ενός ελεγκτή της θύρας οθόνης VGA (Video Graphics Array). Με τον προγραμματισμό της συγκεκριμένης θύρας επιτυγχάνουμε την οδήγηση μίας εικόνας σε μία οθόνη. Σε αυτήν την εργαστηριακή αναφορά περιγράφεται ο τρόπος, με τον οποίο έχουμε ορίσει την VRAM(Video RAM), για να μπορούμε να απεικονίζουμε αδιάκοπα στην οθόνη, με την συνεχή οδήγηση της VGA. Επιθυμούμε να αναπαραστήσουμε ένα παρόμοιο pattern, με αυτό της παρακάτω εικόνας Σχήμα 1. Η συγκεκριμένη εικόνα έχει προδιαγραφές:

- Ανάλυση 640 × 480 pixel (picture element — στοιχείων οθόνης - κουκίδων)
- Ρυθμός Ανανέωσης (refresh rate) 60Hz.



Σχήμα 1: Εικόνα Ελέγχου για την Επαλήθευση VGA Ελεγκτή

ΕΙΣΑΓΩΓΗ:

Οι στόχοι της εργασίας, είναι η υλοποίηση ενός VGA port. Δηλαδή, μία θύρα, η οποία οδηγεί μία εικόνα συνεχώς σε μία οθόνη. Ο χρήστης αποθηκεύει την συγκεκριμένη εικόνα σε

μία μονάδα VRAM, η οποία αποτελείται από 3 επιμέρους BRAM(Block RAM), για να μπορέσει να υποστηρίξει 8 χρώματα. Μία για κάθε χρώμα (RGB = {Red Green Blue}).

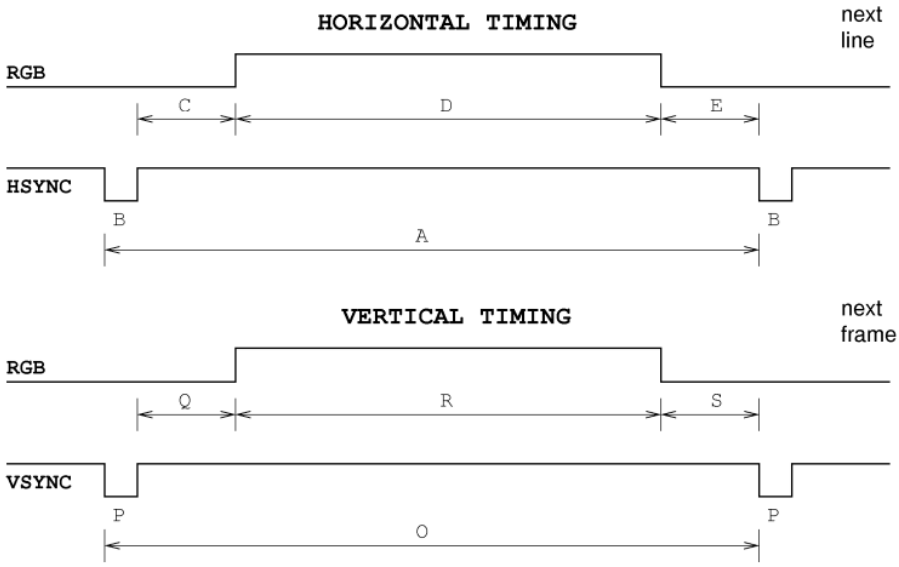
Η θύρα VGA, αποτελείται από 15 ακίδες, απ' τις οποίες εμείς χρησιμοποιούμε τις 5. Χρησιμοποιούμε 3, για την απεικόνιση των 3^{ων} βασικών χρωμάτων (RGB) και τις άλλες 2, για να οδηγήσουμε τα σήματα HSYNC & VSYNC, τα οποία χρησιμοποιούνται για τον έλεγχο του οριζοντίου χρονισμού και τον έλεγχο του καθέτου χρονισμού. Τα σήματα αν και ορίζονται ως αναλογικά με ύψος παλμού 0.7V, τα ορίζουμε ως ψηφιακά.

Αντιστοιχίζουμε τα χρώματα σύμφωνα με το παρακάτω πίνακα Σχήμα 2, όπου αναπαριστούμε 8 χρώματα σύμφωνα με τις τιμές των σημάτων Red, Green, Blue:

Κόκκινο (Red)	Πράσινο (Green)	Μπλε (Blue)	Συνισταμένη Χρώματος
0	0	0	Μαύρο
0	0	1	Μπλε
0	1	0	Πράσινο
0	1	1	Κυανό
1	0	0	Κόκκινο
1	0	1	Μοβ
1	1	0	Κίτρινο
1	1	1	Άσπρο

Σχήμα 2: Απεικόνιση της ψηφιακής αντιστοίχισης χρωμάτων

Τα δύο αυτά σήματα χρονισμού (HSYNC, VSYNC), στις οθόνες CRT, ελέγχουν τη δέσμη, ενώ σε τύπου Υγρών Κρυστάλλων (LCD) οθόνες, ελέγχουν την εγγραφή νέων pixel. Εφόσον η εικόνα είναι ενεργή, οι τιμές των σημάτων Red, Green, Blue καθορίζουν το χρώμα εξόδου της εικόνας. Όπως φαίνεται από την παρακάτω εικόνα Σχήμα 3, οι κυματομορφές για το HSYNC & VSYNC, είναι παρόμοιες. Περιγράφουμε την οριζόντια σάρωση, για χρόνο E μετά το πέρας της ενεργοποίησης της εικόνας υπάρχει το μπροστινό μέρος (front porch), έπειτα το διαδέχεται ο παλμός B, το πλάτος και η συχνότητα του οποίου καταδεικνύουν στην οθόνη την οριζόντια ανάλυση. Ύστερα για χρονικό διάστημα C, την πίσω όψη (back porch), όπου η εικόνα παραμένει ανενεργή. Τέλος εμφανίζεται η επόμενη γραμμή, για χρονικό διάστημα D, όπου η εικόνα είναι ενεργή. Σύμφωνα με την άνωθεν περιγραφή της οριζοντίας σάρωσης λειτουργεί και η κατακόρυφη.



Σχήμα 3: Οριζόντιος και Κατακόρυφος Συγχρονισμός του VGA Οδηγού

0 χρονισμός των διαστημάτων εξαρτάται από την ανάλυση (640x480) και τον ρυθμό ανανέωσης (60Hz). Έτσι ο παρακάτω πίνακας Σχήμα 4, απεικονίζει τις χρονικές τιμές που έχουν τα διανύσματα στο άνωθεν Σχήμα 3.

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

Σχήμα 4: Τιμές Διαστημάτων των Κυματομορφών

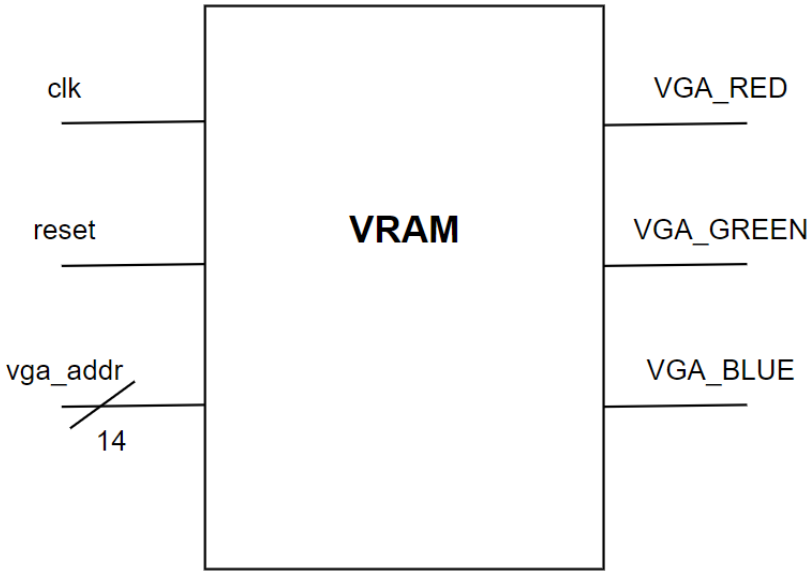
Η ενεργοποίηση της εικόνας εξαρτάται από τα σήματα HSYNC & VSYNC. Εάν τα σήματα αυτά είναι ανενεργά, τότε η εικόνα απενεργοποιείται. Επίσης, εάν είναι ένα από αυτά τα δύο ανενεργά, τότε η οθόνη απενεργοποιείται. Αυτό συμβαίνει, για να εξοικονομήσει ενέργεια το VGA.

Η VRAM, είναι μία μνήμη, που απεικονίζεται συνεχώς στην οθόνη. Το μέγεθος της VRAM, έχει απευθείας σχέση με την ανάλυση της οθόνης μας. Οπότε εφόσον έχουμε ανάλυση 640 x 480 pixels, θέλουμε 640 × 480 = 307200 bits μνήμης, για μονόχρωμα bits. Άρα, για την απεικόνιση έγχρωμων bits, θέλουμε 3 x 307200 = 115.2K Bytes μνήμης.

Εμείς για την αποθήκευση όλων αυτών των bits, χρησιμοποιούμε την BRAM, η οποία είναι εσωτερική μνήμη και χρησιμοποιείται για την δημιουργία της VRAM, αντιστοιχίζοντας 3 BRAM των 16K x 1, σε μία VRAM 128 x 96.

ΜΕΡΟΣ Α -ΥΛΟΠΟΙΗΣΗ VRAM

Υλοποίηση:



Σε αυτό το Μέρος θα υλοποιήσουμε μία VRAM, η οποία αποτελεί κεντρικό τμήμα του ελεγκτή VGA. Η VRAM, θα χωρέσει εικόνα μεγέθους 128 x 96 pixel και 8 χρωμάτων. Η VRAM, αποτελείται από 3 επιμέρους BRAM, οι οποίες αρχικοποιούνται ξεχωριστά για κάθε ένα από τα βασικά χρώματα Red Green Blue.

Παίρνω τις BRAM, από το Vivado, ακολουθώντας τις οδηγίες και τις αρχικοποιώ κατάλληλα. Δηλαδή όταν ένα χρώμα εμφανίζεται στην οθόνη τότε θέλω σε εκείνο το σημείο το bit, που αναπαριστά το χρώμα να είναι ενεργό. Οπότε αν θέλω να αναπαραστήσω το κόκκινο θα πρέπει η BRAM του κόκκινου χρώματος να έχει το bit της στο 1 και οι δύο άλλες BRAM, του πράσινου και μπλε να είναι 0, στο συγκεκριμένο bit. Αρχικοποιώ τις BRAM, μέσω των παραμέτρων .INIT_xx. Με το τρόπο που περιγράψαμε πάνω. Περιγράφω τους χρωματικούς συνδυασμούς από το τέλος προς την αρχή, της κάθε γραμμής των .INIT_xx, επειδή έτσι αναπαριστούνται με την σωστή σειρά στην οθόνη, δηλαδή αντίστροφα.

Περιγράφουμε το σχήμα της εικόνας επαλήθευσης, κάνοντας χρήση 48 γραμμών που έχουν η κάθε μία 256 bits. Οπότε περιγράφουμε μέχρι τα μέσα της κάθε σειράς των INIT_xx, την κάθε γραμμή της εικόνας. Μετά την 48^{η} γραμμή, δεν μας ενδιαφέρει η τιμή που λαμβάνουν τα INIT_xx, οπότε τα αφήνουμε μηδέν. Αυτό συμβαίνει, επειδή ο πίνακάς μας είναι 128 x 96 = 12.288 bits, που σημαίνει ότι θα χρησιμοποιήσουμε για να τον περιγράψουμε 48 σειρές των INIT_xx, αφού 48 x 256 = 12.288 bits.

Για την υλοποίηση της VRAM δημιουργήσαμε το παρακάτω module:

Αποτελείται από τις εισόδους:

- **reset**: Ασύγχρονο reset του κυκλώματός μας
- **clk**: Ρολόι, με fclk = 100MHz συχνότητα
- **vga_addr**: Η διεύθυνση του στοιχείου μέσα στην VRAM. Αποθηκευμένο σαν στοιχείο συνδυασμού των χρωμάτων των διευθύνσεων των 3^{ων} BRAM.

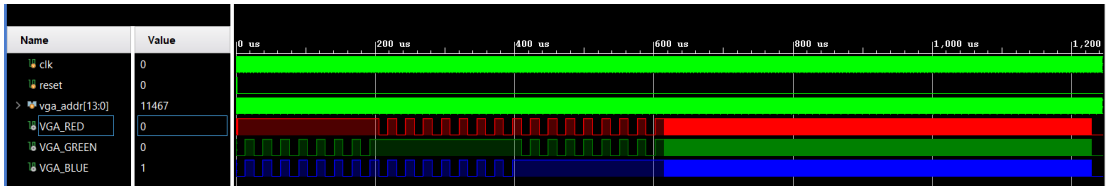
Αποτελείται από τις εξόδους:

- **VGA_RED**: Η τιμή του στοιχείου, όσον αφορά το κόκκινο χρώμα.
- **VGA_GREEN**: Η τιμή του στοιχείου, όσον αφορά το πράσινο χρώμα.

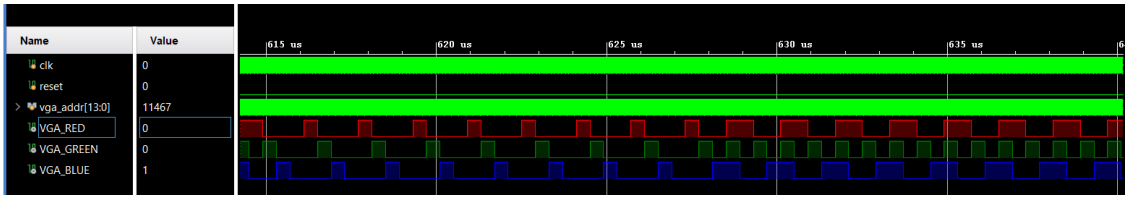
- **VGA_BLUE**: Η τιμή του στοιχείου, όσον αφορά το μπλε χρώμα.

Επαλήθευση:

Για την επαλήθευση του κυκλώματός μας έχουμε δημιουργήσει ένα testbench, στο οποίο εξετάζουμε τις τιμές των σημάτων εξόδου VGA_RED, VGA_GREEN, VGA_BLUE, για διαφορετικές τιμές του vga_addr.



Στο παραπάνω σχήμα βλέπουμε, ότι τα χρώματα παίρνουν μία μορφή όσο το δυνατόν κοντύτερα σε αυτή της εικόνας που πρέπει να περιγράψουμε. Αρχίζουμε με ανά γραμμή εναλλαγές του κόκκινου και του άσπρου χρώματος, ύστερα με ανά γραμμή εναλλαγές του πράσινου και άσπρου, μετά με ανά γραμμή εναλλαγές του μπλέ και άσπρο. Τέλος έχουμε ανά στήλη εναλλαγές του κόκκινου πράσινου μπλε μαύρου, με ανά γραμμή εναλλαγές του κόκκινου πράσινου μπλε άσπρου.



Το δεύτερο μέρος της περιγραφής μας φαίνεται στο σχήμα παραπάνω, αρχικά για την εναλλαγή με κόκκινο-πράσινο-μπλε-μαύρο και ύστερα με την εναλλαγή με κόκκινο-πράσινο-μπλε-άσπρο.

ΜΕΡΟΣ Β -ΥΛΟΠΟΙΗΣΗ HSYNC ΚΑΙ ΟΡΙΖΟΝΤΙΟΥ ΜΕΤΡΗΤΗ PIXEL

Υλοποίηση:

Σε αυτό το Μέρος θα υλοποιήσουμε το σήμα HSYNC και τον οριζόντιο μετρητή Pixel, για ανάλυση 640 x 480 και ρυθμό ανανέωσης 60Hz. Παράλληλα με το HSYNC, χρησιμοποιούμε και τον οριζόντιο μετρητή pixel HPIXEL, ο οποίος επιλέγει κατάλληλα τα pixel που έχουμε αποθηκευμένα στην VRAM. Από την διεύθυνση 0, του 1^{ου} pixel, έως την τελική διεύθυνση 12268 (128 x 96 = 12268), όπου 128 x 96, είναι οι διαστάσεις των BRAM, από τις οποίες αποτελείται η VRAM.

Για τον οριζόντιο μετρητή έχουμε υπολογίσει τους κύκλους ρολογιού, για τους αντίστοιχους χρόνους και τις αντίστοιχες καταστάσεις σύμφωνα με το παρακάτω πινακάκι:

State	Symbol	Parameter	Time	Clocks
Sum of all states	Ts	Sync pulse	32 us	800
StateD	Tdisp	Display time	25.6 us	640
StateB	Tpw	Pulse width	3.84 us	96
StateE + StateA	Tfp	Front porch	640 ns	16
StateC	Tbp	Back porch	1.92 us	48

Σχήμα 5: Χρονισμοί σημάτων για ανάλυση οθόνης 640-Pixel by 480-Row με 25 MHz Pixel ρολόι και 60 Hz ρυθμό ανανέωσης, για τον οριζόντιο μετρητή

Για την υλοποίηση του Hcounter, οριζόντιου μετρητή pixel, δημιουργούμε το παρακάτω module:

Το οποίο έχει ως εισόδους:

- **reset**: Ασύγχρονο σήμα αναίρεσης του κυκλώματος.
- **clk**: Ρολόι του κυκλώματος μας.

Έχει ως εξόδους:

- **VGA_HSYNC**: Σήμα οριζόντιου συγχρονισμού για τη θύρα VGA.
- **HPIXEL**: Αποτελεί την διεύθυνση των οριζόντιων bit (6 bits), τα οποία θα χρησιμοποιήσουμε αργότερα για να συνθέσουμε το vga_addr.

- **hcount**: Μετρητής (10 bit), ο οποίος μετράει για το χρονικό διάστημα το οποίο έχει δωθεί στο πίνακα των χρόνων παραπάνω και μας βοηθάει στην μετάβαση από τη μία κατάσταση στην άλλη ανάλογα με την τιμή που έχει.
- **HRGB_EN**: Σήμα το οποίο δηλώνει, πότε το RGB μπορεί να πάρει διάφορες τιμές σύμφωνα με αυτό που θέλουμε να μεταδώσουμε.

Υλοποιούμε μία FSM, μέσα στο module στην οποία κάνουμε τις κατάλληλες ενέργειες, για τη μετάδοση του σήματος:

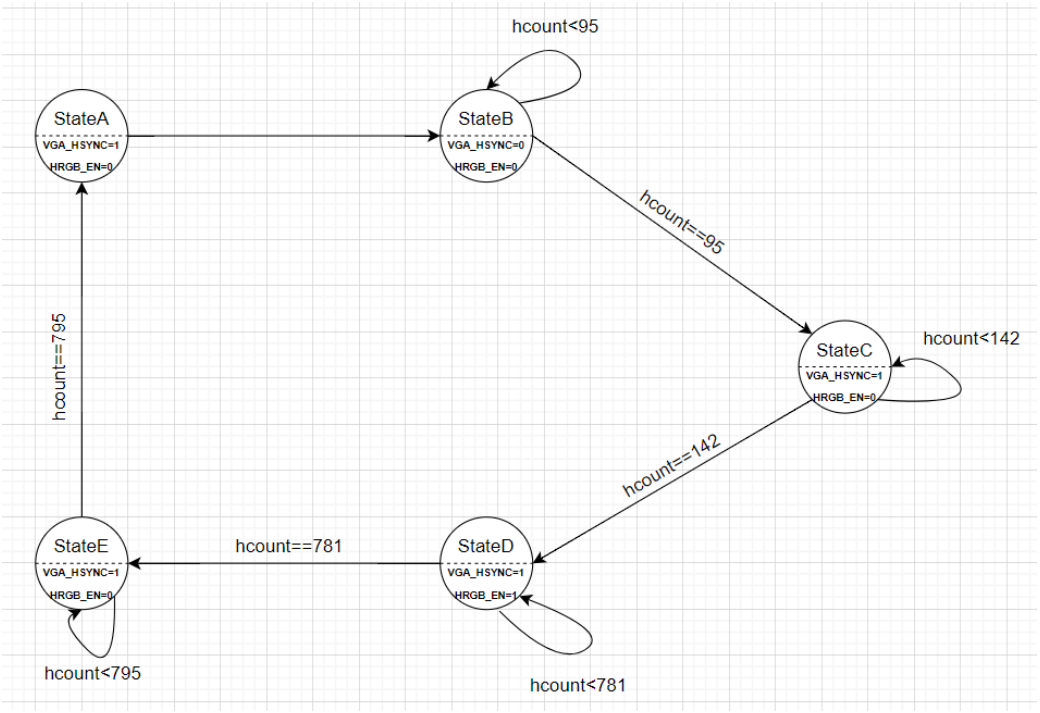
1) **stateA(0):** Σε αυτή την κατάσταση μετράμε έναν κύκλο και ο counter αρχικοποιεί το σήμα hcount σε 0 και θέτει τα σήματα εξόδου HRGB_EN = 0 και VGA_HSYNC = 1 και μεταβαίνει απευθείας στην επόμενη κατάσταση.
2) **stateB(1):** Σε αυτήν την κατάσταση ο μετρητής hcount αρχίζει και μετράει και όταν μετρήσει συνολικά 96 κύκλους, τότε μεταβαίνει στην επόμενη κατάσταση, συνάμα το σήμα VGA_HSYNC, γίνεται 0 ενώ το σήμα HRGB_EN, παραμένει 0.
3) **stateC(2):** Σε αυτή την κατάσταση ο μετρητής hcount αρχίζει και μετράει και όταν μετρήσει άλλους 48 κύκλους, τότε μεταβαίνει στην επόμενη κατάσταση, συνάμα το σήμα VGA_HSYNC, γίνεται 1 ενώ το σήμα HRGB_EN, παραμένει 0.
4) **stateD(3):** Σε αυτήν την κατάσταση μετράμε άλλους 640 κύκλους για το hcount και ανά πέντε bits, που μετράμε αυξάνουμε το μετρητή hcount_ins, αυτό επιτυχγάνεται χρησιμοποιώντας μία μεταβλητή την οποία ονομάζουμε flag, κάθε φορά που ισούται με 4. Ύστερα μηδενίζεται και επαναλαμβάνεται η διαδικασία μέχρι τη λήξη της κατάστασης. Το hcount_ins, είναι σημαντικό επειδή είναι αυτό από το οποίο παίρνει τιμή το HPIXEL. Ουσιαστικά το HPIXEL, ανά 5 bit ανανεώνει την τιμή του λόγω του μεγέθους που μετράμε συνολικά, σε αυτήν την κατάσταση. Δειγματολειπτώντας, ανά 5 bit είναι σαν να διαιρείς με 5 το συνολικό 640, 640/5 = 128, το οποίο είναι το μέγεθος της κάθε σειράς στην VRAM. Τέλος το σήμα VGA_HSYNC, παραμένει 0 ενώ το σήμα HRGB_EN, ενεργοποιείται γίνεται 1.
5) **stateE(4):** Σε αυτήν την κατάσταση μετράμε άλλους 15 κύκλους με τον hcount και επιστρέφουμε ξανά στην αρχική κατάσταση και θέτει τα σήματα εξόδου HRGB_EN = 0 και VGA_HSYNC = 1.

Για την υλοποίηση της FSM χρησιμοποιούμε δύο always blocks:

Το πρώτο είναι sequential always block και είναι υπεύθυνο για να δίνει τιμές στην στην τωρινή κατάσταση εξισώνοντάς τη με την επόμενη κατάσταση. Κάθε φορά που αλλάζει μία μεταβλητή αυτή ανατίθεται στην τωρινή κατάσταση. Αλλιώς εάν πατηθεί το reset, έχουμε αρχικοποίηση όλων των τιμών σε 0.

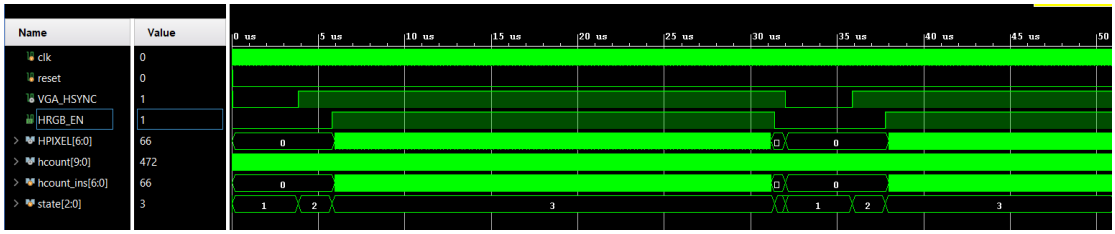
Το δεύτερο είναι combinational always block και τροποποιεί τις επόμενες καταστάσεις.

Παρακάτω παραθέτουμε το FSM μοντέλο σχηματικά, ως μοντέλο Moore:



Επαλήθευση:

Στο testbench, ελέγχουμε αν οι καταστάσεις και οι χρόνοι συμπίπτουν όπως περιγράφουμε στο πινακάκι πάνω (Σχήμα 5):



Οι χρόνοι φαίνεται να είναι σωστοί και το σήμα HSYNC, όπως και το σήμα HRGB_EN, να ενεργοποιούνται στις κατάλληλες στιγμές.

ΜΕΡΟΣ Γ -ΥΛΟΠΟΙΗΣΗ VSYNC ΚΑΙ ΚΑΤΑΚΟΡΥΦΟΥ ΜΕΤΡΗΤΗ PIXEL-ΟΛΟΚΛΗΡΩΣΗ ΤΟΥ ΕΛΕΓΚΤΗ/ΟΔΗΓΟΥ VGA.

Υλοποίηση:

Υλοποίηση VSYNC και του κατακόρυφος μετρητής pixel:

Σε αυτό το Μέρος θα υλοποιήσουμε το σήμα VSYNC και τον κατακόρυφο μετρητή Pixel, για ανάλυση 640 x 480 και ρυθμό ανανέωσης 60Hz. Παράλληλα με το VSYNC, χρησιμοποιούμε και τον οριζόντιο μετρητή pixel VPIXEL, ο οποίος επιλέγει κατάλληλα τα pixel που έχουμε αποθηκευμένα στην VRAM. Από την διεύθυνση 0, του 1^{ου} pixel, έως την τελική διεύθυνση 12268 (128 x 96 = 12268), όπου 128 x 96, είναι οι διαστάσεις των BRAM, από τις οποίες αποτελείται η VRAM. Ο μετρητής VPIXEL, αυξάνεται κάθε φορά που αλλάζουμε σειρά και παίρνει τιμές από το 0 έως το 95, δηλαδή 96 συνολικά σειρές της VRAM, που έχει δεδομένα τις διασχίζουμε με τον κατακόρυφο μετρητή, ο οποίος σε συνδυασμό με τον οριζόντιο μετρητή μας δίνει την vga_addr.

Για τον κατακόρυφο μετρητή έχουμε υπολογίσει τους κύκλους ρολογιού, για τους αντίστοιχους χρόνους και τις αντίστοιχες καταστάσεις σύμφωνα με το παρακάτω πινακάκι:

State	Symbol	Parameter	Time	Clocks	Lines
Sum of all states	Ts	Sync pulse	16.7 ms	416800	521
StateR	Tdisp	Display time	15.36 ms	384000	480
StateP	Tpw	Pulse width	64 us	1600	2
StateS + StateO	Tfp	Front porch	320 ns	8000	10
StateQ	Tbp	Back porch	928 us	23200	29

Σχήμα 6: Χρονισμοί σημάτων για ανάλυση οθόνης 640-Pixel by 480-Row με 25 MHz Pixel ρολόι και 60 Hz ρυθμό ανανέωσης, για τον κατακόρυφο μετρητή

Για την υλοποίηση του Vcounter, κατακόρυφου μετρητή pixel, δημιουργούμε το παρακάτω module:

Το οποίο έχει ως εισόδους:

- **reset**: Ασύγχρονο σήμα αναίρεσης του κυκλώματος.
- **clk**: Ρολόι του κυκλώματος μας.

- **hcount**: Μετρητής (10 bit), ο οποίος μετράει για το χρονικό διάστημα, που διαρκεί ολόκληρος ο παλμός του Hcounter, από την αρχή μέχρι το τέλος του. Το χρησιμοποιούμε αυτό ως σήμα ελέγχου, για να αυξάνουμε το vcount, το οποίο είναι ο μετρητής που χρησιμοποιούμε καθ' όλες τις καταστάσεις του Vcounter.

Έχει ως εξόδους:

- **VGA_VSYNC**: Σήμα κατακόρυφου συγχρονισμού για τη θύρα VGA.
- **VPIXEL**: Αποτελεί την διεύθυνση των κατακόρυφων bit (6 bits), τα οποία θα χρησιμοποιήσουμε αργότερα για να συνθέσουμε το vga_addr.
- **VRGB_EN**: Σήμα το οποίο δηλώνει, πότε το RGB μπορεί να πάρει διάφορες τιμές σύμφωνα με αυτό που έχουμε αποθηκευμένο στη VRAM.

Υλοποιούμε μία FSM, μέσα στο module στην οποία κάνουμε τις κατάλληλες ενέργειες, για τη μετάδοση του σήματος:

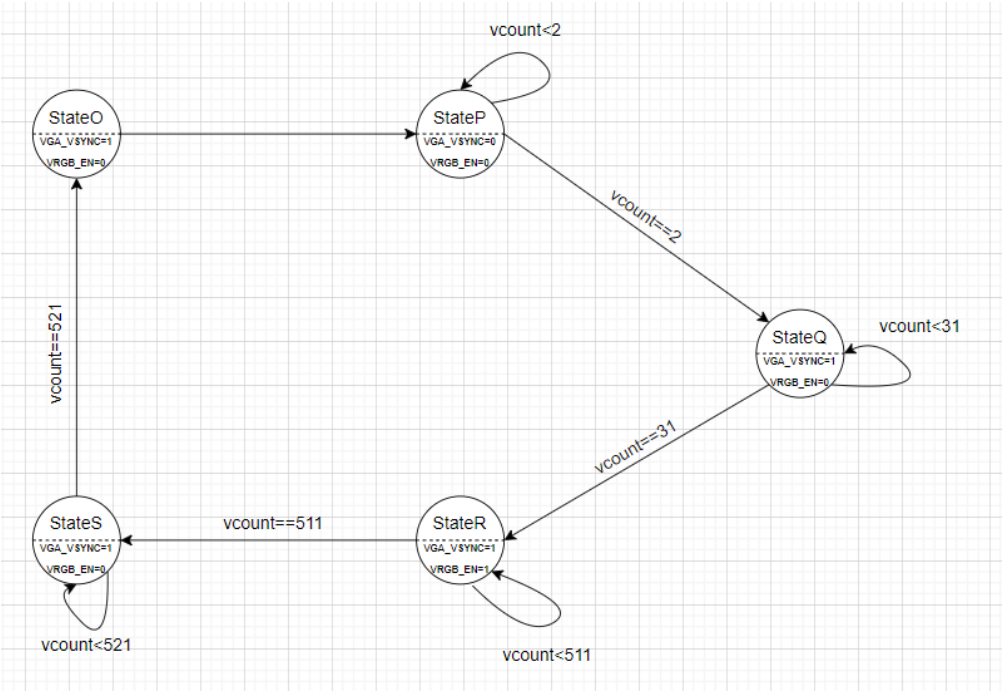
1) **stateO(0):** Σε αυτή την κατάσταση μετράμε έναν κύκλο και ο counter αρχικοποιεί το σήμα vcount σε 0 και θέτει τα σήματα εξόδου VRGB_EN = 0 και VGA_VSYNC = 1 και μεταβαίνει απευθείας στην επόμενη κατάσταση.
2) **stateP(1):** Σε αυτήν την κατάσταση ο μετρητής vcount αρχίζει και μετράει και όταν μετρήσει συνολικά 2 γραμμές, τότε μεταβαίνει στην επόμενη κατάσταση, συνάμα το σήμα VGA_VSYNC, γίνεται 0 ενώ το σήμα VRGB_EN, παραμένει 0.
3) **stateQ(2):** Σε αυτή την κατάσταση ο μετρητής vcount αρχίζει και μετράει και όταν μετρήσει άλλες 29 γραμμές, τότε μεταβαίνει στην επόμενη κατάσταση, συνάμα το σήμα VGA_VSYNC, γίνεται 1 ενώ το σήμα VRGB_EN, παραμένει 0.
4) **stateR(3):** Σε αυτήν την κατάσταση μετράμε άλλες 480 γραμμές για το vcount και ανά πέντε bits, που μετράμε αυξάνουμε το μετρητή vcount_ins, αυτό επιτυχγάνεται χρησιμοποιώντας μία μεταβλητή την οποία ονομάζουμε flag, κάθε φορά που ισούται με 4. Ύστερα μηδενίζεται και επαναλαμβάνεται η διαδικασία μέχρι τη λήξη της κατάστασης. Το vcount_ins, είναι σημαντικό επειδή είναι αυτό από το οποίο παίρνει τιμή το VPIXEL. Ουσιαστικά το VPIXEL, ανά 5 bit ανανεώνει την τιμή του λόγω του μεγέθους που μετράμε συνολικά, σε αυτήν την κατάσταση. Δειγματολειπτώντας, ανά 5 bit είναι σαν να διαιρείς με 5 το συνολικό 480, 480/5 = 96, το οποίο είναι το μέγεθος της κάθε σήλης στην VRAM. Τέλος το σήμα VGA_VSYNC, παραμένει 0 ενώ το σήμα VRGB_EN, ενεργοποιείται γίνεται 1.
5) **stateS(4):** Σε αυτήν την κατάσταση μετράμε άλλες 10 γραμμές, μείον ένα κύκλο με τον vcount και επιστρέφουμε ξανά στην αρχική κατάσταση και θέτει τα σήματα εξόδου VRGB_EN = 0 και VGA_VSYNC = 1.

Για την υλοποίηση της FSM χρησιμοποιούμε δύο always blocks:

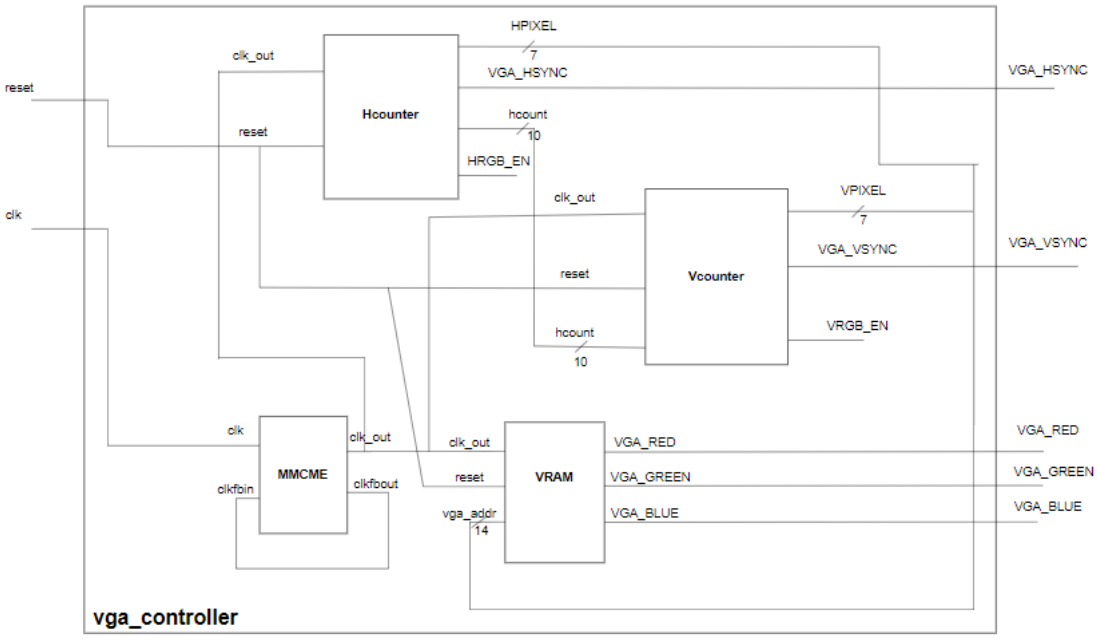
Το πρώτο είναι sequential always block και είναι υπεύθυνο για να δίνει τιμές στην στην τωρινή κατάσταση εξισώνοντάς τη με την επόμενη κατάσταση. Κάθε φορά που αλλάζει μία μεταβλητή αυτή ανατίθεται στην τωρινή κατάσταση. Αλλιώς εάν πατηθεί το reset, έχουμε αρχικοποίηση όλων των τιμών σε 0.

Το δεύτερο είναι combinational always block και τροποποιεί τις επόμενες καταστάσεις.

Παρακάτω παραθέτουμε το FSM μοντέλο σχηματικά, ως μοντέλο Moore:



Υλοποίηση ελεγκτή/οδηγού vga (vga_controller):



Συνδέουμε τα modules, που έχουμε δημιουργήσει και αναφέρονται παραπάνω μεταξύ τους, για την δημιουργία του οδηγού vga. Επιπλέον δημιουργήσαμε και μία μονάδα MMCME, για

την μετατροπή του γρήγορου ρολογιού σε αργό έτσι ώστε ο χρόνος να αντιστοιχεί ακριβώς στις περιόδους ρολογιού που αναγράφονται στους πίνακές μας.

Υλοποιήσαμε μία μονάδα MMCME, η οποία μετατρέπει το γρήγορο ρολόι σε αργό. Θέσαμε την τιμή CLFBOUT_MULT_F, σε 12, την τιμή CLKIN1_PERIOD σε 10, την τιμή CLKOUT1_DIVIDE σε 48 και την τιμή DIVCLK_DIVIDE σε 1. Σύμφωνα με την σχέση που έχουμε στο manual μας, ισχύει ότι: Fout = Fclkin x M/(D x O), όπου Fout, η νέα συχνότητα με το αργό ρολόι, το οποίο έχει περίοδο 40ns, άρα η Fout = 25MHz. Fin, η συχνότητα με την οποία οδηγείται το γρήγορο ρολόι, το οποίο έχει περίοδο 10ns, άρα Fclkin = 100Mhz. Το M αντιστοιχίζεται στο CLFBOUT_MULT_F, το οποίο καθορίζει την ποσότητα με την οποία θα πολλαπλασιάσουμε όλα τα CLKOUT, δηλαδή τα output clocks, με τη συχνότητα που επιθυμούμε. Σε συνδυασμό με το DIVCLK_DIVIDE, καθορίζει τη συχνότητα της εξόδου. Το D, αντιστοιχίζεται στο DIVCLK_DIVIDE, το οποίο καθορίζει το μέγεθος για όλα τα ρολόγια των εξόδων, σε συνδυαστολή με το ρολόι της εισόδου, διαιρώντας αποτελεσματικά το CLKIN. Το O αντιστοιχίζεται στο CLKOUT_DIVIDE, και στην δικιά μας περίπτωση στο CLKOUT1_DIVIDE, το οποίο κάνει την ίδια δουλεία με το DIVCLK_DIVIDE, αλλά μόνο για το συγκεκριμένο ρολόι. Το CLKIN1_PERIOD, είναι η περίοδος του ρολογιού της εισόδου μας, το οποίο είναι 10ns.

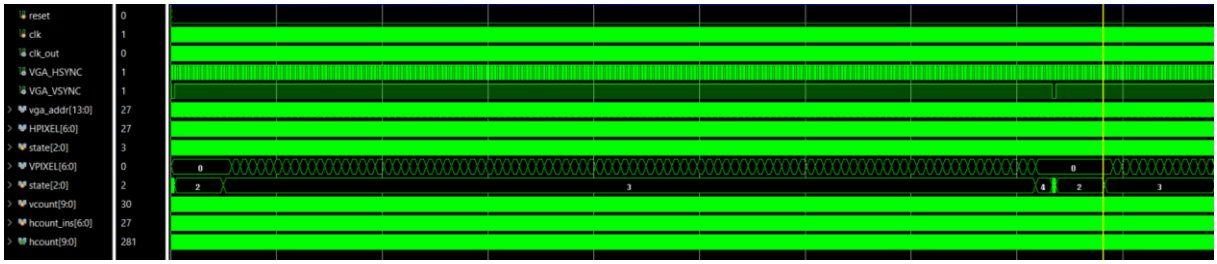
Επίσης για την σύνδεση με την πλακέτα χρησιμοποιήσαμε ένα αρχείο με constraints, το οποίο συνδέει τις μεταβλητές μας στην βέριλογκ με τα pins της πλακέτας μας:

```
## This file is a general .xdc for the Nexys A7-100T
### Clock Signal
set_property -dict { PACKAGE_PIN E3 IOSTANDARD LVCMOS33 } [get_ports { clk }];
create_clock -add -name clk -period 10.00 -waveform {0 5} [get_ports { clk }];
### RGB Display
##RED
set_property -dict { PACKAGE_PIN A3 IOSTANDARD LVCMOS33 } [get_ports { VGA_RED }];
set_property -dict { PACKAGE_PIN B4 IOSTANDARD LVCMOS33 } [get_ports { VGA_RED }];
set_property -dict { PACKAGE_PIN C5 IOSTANDARD LVCMOS33 } [get_ports { VGA_RED }];
set_property -dict { PACKAGE_PIN A4 IOSTANDARD LVCMOS33 } [get_ports { VGA_RED }];

##GREEN
set_property -dict { PACKAGE_PIN C6 IOSTANDARD LVCMOS33 } [get_ports { VGA_GREEN }];
set_property -dict { PACKAGE_PIN A5 IOSTANDARD LVCMOS33 } [get_ports { VGA_GREEN }];
set_property -dict { PACKAGE_PIN B6 IOSTANDARD LVCMOS33 } [get_ports { VGA_GREEN }];
set_property -dict { PACKAGE_PIN A6 IOSTANDARD LVCMOS33 } [get_ports { VGA_GREEN }];

##BLUE
set_property -dict { PACKAGE_PIN B7 IOSTANDARD LVCMOS33 } [get_ports { VGA_BLUE }];
set_property -dict { PACKAGE_PIN C7 IOSTANDARD LVCMOS33 } [get_ports { VGA_BLUE }];
set_property -dict { PACKAGE_PIN D7 IOSTANDARD LVCMOS33 } [get_ports { VGA_BLUE }];
set_property -dict { PACKAGE_PIN D8 IOSTANDARD LVCMOS33 } [get_ports { VGA_BLUE }];

set_property -dict { PACKAGE_PIN B11 IOSTANDARD LVCMOS33 } [get_ports { VGA_HSYNC }];
set_property -dict { PACKAGE_PIN B12 IOSTANDARD LVCMOS33 } [get_ports { VGA_VSYNC }];
### Button(s)
set_property -dict { PACKAGE_PIN N17 IOSTANDARD LVCMOS33 } [get_ports { reset }];
```

Επαλήθευση:

Επαλήθευση VSYNC και του κατακόρυφος μετρητής pixel:

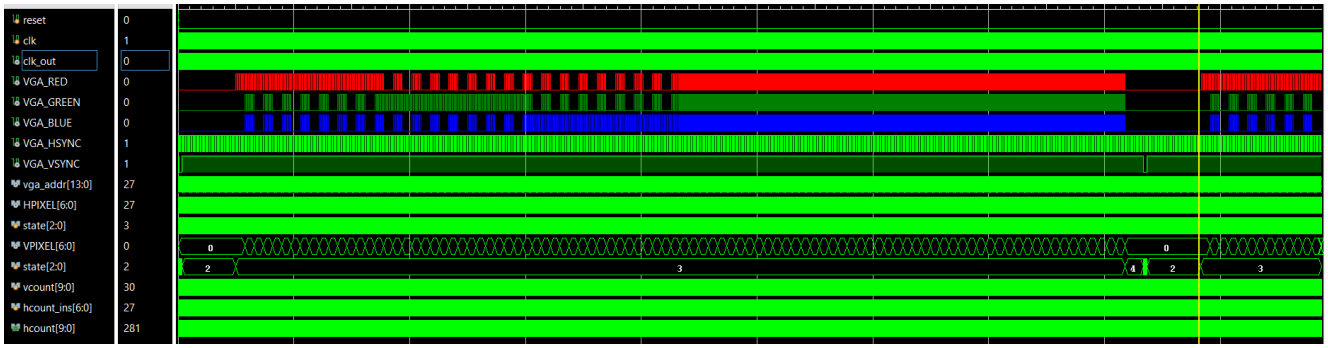
Στο testbench, ελέγχουμε αν οι καταστάσεις και οι χρόνοι συμπίπτουν όπως περιγράφουμε στο πινακάκι πάνω (Σχήμα 6):



Οι χρόνοι φαίνεται να είναι σωστοί και το σήμα VSYNC, όπως και το σήμα VRGB_EN, να ενεργοποιούνται στις κατάλληλες στιγμές. Ο μετρητής VPIXEL λαμβάνει σωστές τιμές από το 0->95(96 κύκλους). Τα σήματα συγχρονίζονται επιτυχώς και οι καταστάσεις, όπως τις περιγράφουμε στον πίνακα απεικονίζονται και στις κυματομορφές.

Επαλήθευση ελεγκτή/οδηγού vga (vga_controller):

Στο testbench, ελέγχουμε αν ενεργοποιούνται τα χρώματα (RGB), στην κατάλληλη κατάσταση και αν απεικονίζονται με την σωστή σειρά, την οποία έχουμε ορίσει μέσω της VRAM:



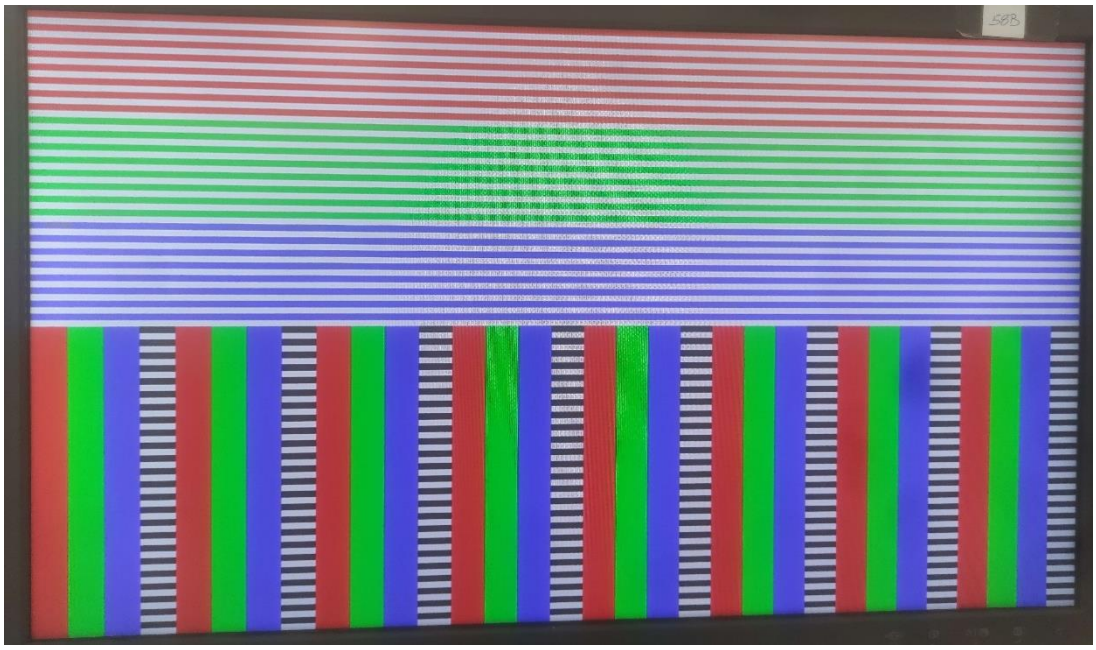
Τα χρώματα απεικονίζονται κατά τις σωστές ενεργοποιήσεις των καταστάσεων και στην επιθυμητή σειρά, άρα οι διευθύνσεις των χρωμάτων έχουν τις επιθυμητές τιμές. Φαίνεται επίσης από την εικόνα όταν το HRGB_EN είναι μηδέν με διακεκομμένες γραμμές σε μερικά σημεία ότι απενεργοποιούνται τα χρώματα, όπως όταν το VRGB_EN είναι μηδέν οι τιμές των χρωμάτων είναι όλες μηδέν.

Πείραμα/Τελική Υλοποίηση:

Κάνοντας generate το bitstream, το εφαρμόσαμε στην πλακέτα αφού συνδέσαμε και το καλώδιο για το vga στην οθόνη. Στην 1^{η} προσπάθεια παρατηρήσαμε πως το σχήμα ήταν σωστό αλλά τα χρώματα ήταν λανθασμένα. Αφού όμως θέσαμε τα χρώματα να είναι μηδέν

στα διαστήματα όπου το HRGB_EN & το VRGB_EN είναι μηδέν τότε στη 2η προσπάθειά μας, λειτούργησε η οθόνη βγάζοντας της σωστή σειρά και τα κατάλληλα χρώματα.

Παρουσιάζουμε τις κυματομορφές, οι οποίες δημιουργήθηκαν στην οθόνη, μετά την υλοποίηση του κυκλώματος στην πλακέτα, στην 2η και σωστή υλοποίηση του κυκλώματός μας:



Παρατηρούμε επιπλέον από τις οθόνες ότι το άνω μισό την εικόνας έχει εναλλαγές ανάμεσα σε λευκές και κόκκινες/πράσινες/μπλε γραμμές και το δεύτερο μισό έχει στήλες με κόκκινο/πράσινο/μπλε και εναλλαγές γραμμών ανάμεσα σε μαύρο και λευκό, το οποίο είναι ακριβώς το κύκλωμα, που έχουμε αποθηκευμένο στην VRAM.