Report III

Τσόγκας Παναγιώτης Νικόλαος - 3672

1/12/2024

Περιεχόμενα

1 Εισαγωγή **2**

2 Περίληψη **2**

3 Υλοποίηση VRAM PAPUnit **3**
3.1 Επεξήγηση μονάδας PAPU: . . . 3
3.2 Υλοποίηση: . . . 3
3.3 Επαλήθευση: . . . 3
3.4 Πείραμα/Τελική Υλοποίηση: . . . 3
3.5 Συμπεράσματα-Παρατηρήσεις: . . . 3

4 Υλοποίηση σημάτων Συγχρονισμού **4**
4.1 Επεξήγηση μονάδας SSyncFSM: . . . 4
4.2 Υλοποίηση: . . . 5
4.3 Επαλήθευση HSYNC: . . . 6
4.4 Επαλήθευση VSYNC: . . . 6
4.5 Συμπεράσματα-Παρατηρήσεις: . . . 6

5 Εικόνα και Βίντεο **7**
5.1 Εικόνα Design Flow: . . . 7
5.2 Υλοποίηση: . . . 7
5.3 Επαλήθευση: . . . 8
5.4 Βίντεο Υλοποίηση: . . . 9
5.5 Sprite Υλοποίηση: . . . 10
5.6 Σημειώσεις: . . . 11

1 Εισαγωγή

Η παρούσα εργασία ασχολείται με την υλοποίηση ενός συστήματος VGA στην πλατφόρμα FPGA, το οποίο περιλαμβάνει τη δημιουργία και εμφάνιση εικόνας και βίντεο, καθώς και τη διαχείριση της κίνησης των sprite στην οθόνη. Το έργο αναπτύσσεται με τη χρήση μονάδων Verilog και επικεντρώνεται στη σχεδίαση απλών και ευέλικτων μονάδων που διασφαλίζουν τη σωστή εμφάνιση δεδομένων στην οθόνη VGA. Οι μονάδες που δημιουργήθηκαν καλύπτουν διάφορες λειτουργίες, όπως η διαχείριση εικόνας, η επεξεργασία βίντεο, η κίνηση των sprite, καθώς και η συγχρονισμένη παραγωγή σημάτων VGA. Στόχος είναι η επίτευξη εύχρηστων και αποδοτικών λύσεων για την επεξεργασία και την εμφάνιση γραφικών στην οθόνη μέσω της FPGA.

2 Περίληψη

Η υλοποίηση του συστήματος ξεκινά με τη δημιουργία της μονάδας PAPUnit (Πιξελ Αςχυισιτιον Προςεσσινγ Υνιτ), η οποία αναλαμβάνει την ανάγνωση και επεξεργασία των πιξελ για την οθόνη VGA. Ενσωματώθηκαν μονάδες VRAM για την αποθήκευση των δεδομένων των πιξελ, χρησιμοποιώντας μνήμη BRAM και την εντολή initial για την αρχικοποίησή τους, παρέχοντας μια απλή αλλά αποτελεσματική λύση. Στη συνέχεια, υλοποιήθηκε η μονάδα συγχρονισμού SSyncFSM, η οποία διαχειρίζεται τα σήματα χρονισμού HSYNC και VSYNC με τις κατάλληλες καθυστερήσεις, εξασφαλίζοντας την ομαλή αναπαραγωγή εικόνας και βίντεο στην οθόνη. Η μονάδα PixelAddrGen δημιουργεί τις διευθύνσεις μνήμης για τα δεδομένα των πιξελ, βασισμένες στα σήματα VDISP και HDISP, εξασφαλίζοντας την εμφάνιση των σωστών δεδομένων στην κατάλληλη θέση στην οθόνη. Η μονάδα ClockGenerator παρέχει το ρολόι των 25 MHζ, το οποίο χρησιμοποιείται σε όλα τα χρονισμένα σήματα του συστήματος, διασφαλίζοντας τη σωστή χρονική ευθυγράμμιση των σημάτων και των δεδομένων.

Η μονάδα ελέγχου VGA χρησιμοποιεί τα σήματα VDISP και HDISP για την εμφάνιση των δεδομένων πιξελ και συνδυάζει τη στατική εικόνα με την αναπαραγωγή βίντεο. Το βίντεο περιλαμβάνει ένα GIF από το γνωστό βίντεο “Bad Apple”, το οποίο προβάλλεται με ρυθμό 4 καρέ ανά δευτερόλεπτο, με τα καρέ να αποθηκεύονται σε μνήμη BRAM, εξασφαλίζοντας αρκετό χώρο για την αποθήκευση περισσότερων από 60 καρέ. Η υλοποίηση των Sprite επιτρέπει την εμφάνιση και κίνηση γραφικών στοιχείων στην οθόνη, με κάθε sprite να αποθηκεύεται σε μνήμη ROM και να μετακινείται δυναμικά μέσω της μονάδας MovementController, η οποία αναλαμβάνει τη διαχείριση της κίνησης και την ανίχνευση κρούσεων με τα όρια της οθόνης.

3 Υλοποίηση VRAM PAPUnit

3.1 Επεξήγηση μονάδας PAPU:

Η μονάδα PAPUnit (Pixel Acquisition Processing Unit) αποτελεί τη γέφυρα επικοινωνίας μεταξύ του χρήστη και των δεδομένων των pixel προς εκτύπωση στην VGA. Η συγκεκριμένη μνήμη γράφεται μία φορά κατά την υλοποίηση της synthesis και δεν ξαναγράφεται, καθιστώντας την πρακτικά ROM, παρά τη χρήση ενός ακολουθιακού τύπου μνήμης BRAM.

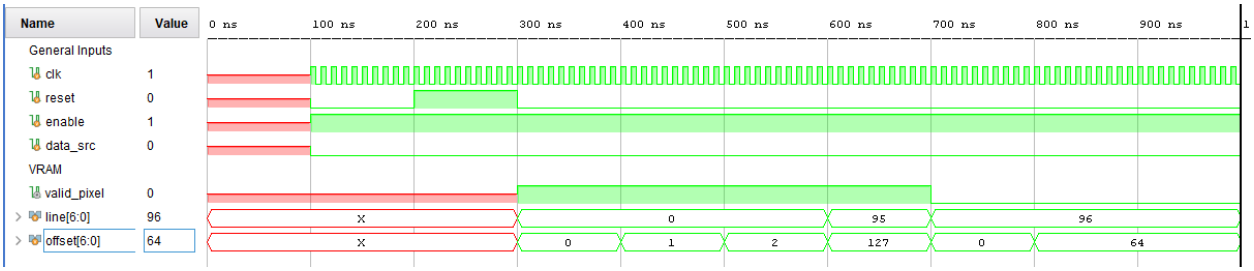
Αξίζει να σημειωθεί ότι οι μονάδες VRAM που χρησιμοποιούνται εντός της μονάδας PAPUnit είναι wrappers για τις πρωτότυπες μνήμες BRAM που παρέχει το Vivado.

3.2 Υλοποίηση:

Στα αρχικά στάδια της εργασίας, και συγκεκριμένα για το Part A, η μονάδα ήταν πολύ απλή, καθώς η έξοδός της ήταν η τιμή RGB ενός pixel σε ένα δεδομένο line και offset. Επιπλέον, η μονάδα πραγματοποιούσε έναν έλεγχο για το αν η διεύθυνση ήταν εντός των πλαισίων της οθόνης, δηλαδή εντός των διαστάσεων $128 \cdot 96$.

3.3 Επαλήθευση:

Για την επαλήθευση της μονάδας υπάρχει μια μονάδα ελέγχου testbench με όνομα "VRAM_tb", η οποία ελέγχει τη λειτουργικότητα και την ορθότητα των εξόδων της μονάδας PAPU. Επιπλέον, εκτυπώνονται κάποια μηνύματα σχετικά με τα αποτελέσματα και τις δοκιμές που εφαρμόζει το testbench.



Σχήμα 1: Κυματομορφές Part A

3.4 Πείραμα/Τελική Υλοποίηση:

Ένα καλό πείραμα για τη μονάδα αυτή είναι να δοκιμαστεί η λειτουργία της με τη χρήση ασύγχρονου reset για την επαναφορά των σημάτων line και offset. Στην πρώτη υλοποίηση της, παρουσιάστηκε ένα πρόβλημα, καθώς παρατηρούνταν timing violation όταν το line ή το offset άλλαζαν ασύγχρονα κοντά στη θετική ακμή του ρολογιού.

Για την αποφυγή αυτού του προβλήματος, χρησιμοποιήθηκε ένας synchronizer, ο οποίος περιορίζει οποιαδήποτε αλλαγή της address που δέχονται οι VRAM μόνο στη θετική ακμή του ρολογιού.

3.5 Συμπεράσματα-Παρατηρήσεις:

Η σχεδίαση και η δοκιμασία της λειτουργικότητας της μονάδας ήταν επιτυχής και φαίνεται να λειτουργεί όπως πρέπει.

Στην τελική έκδοση υπήρχε μόνο το παρακάτω Vivado Warning, το οποίο αγνοήθηκε:

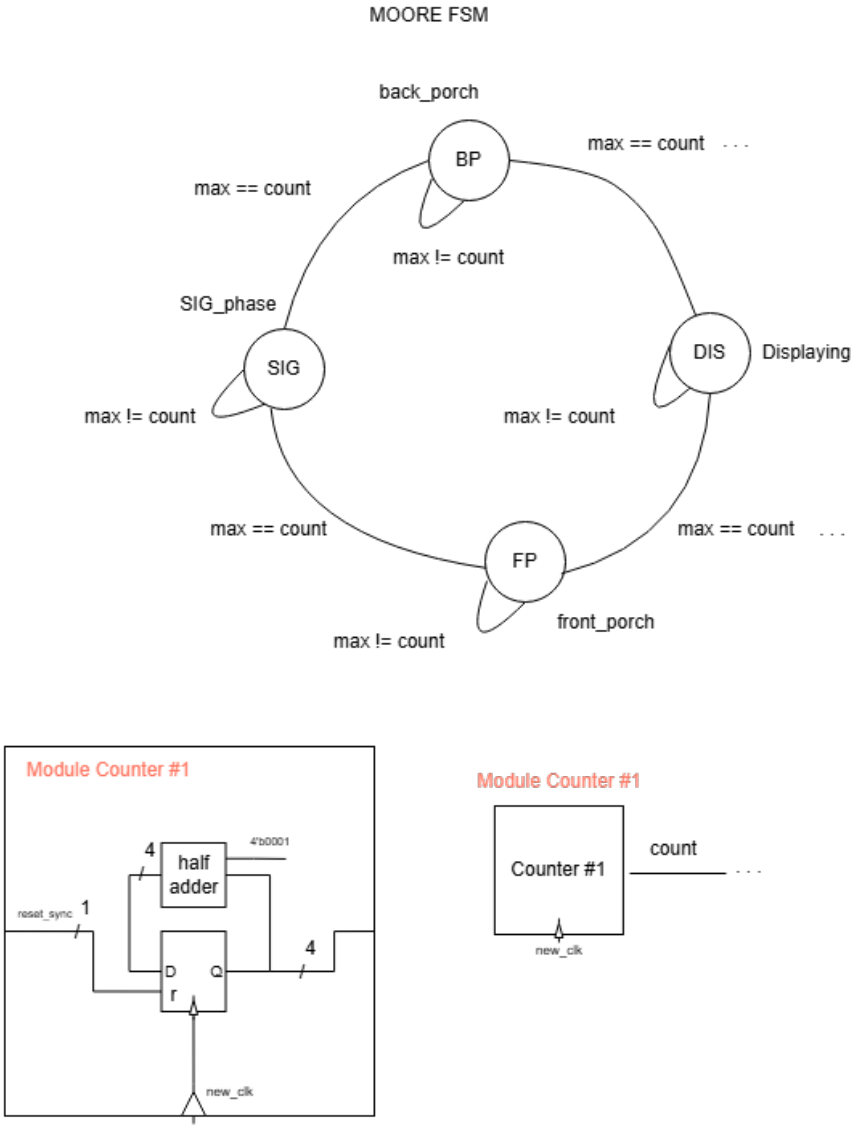
[Constraints 18-5210] No constraints selected for write. Το παραπάνω αγνοήθηκε, καθώς φαίνεται να είναι κάποιο bug του Vivado, δεδομένου ότι έχει οριστεί constraints file στην τελική έκδοση.

4 Υλοποίηση σημάτων Συγχρονισμού

4.1 Επεξήγηση μονάδας SSyncFSM:

Για την υλοποίηση των σημάτων συγχρονισμού χρησιμοποιείται η μονάδα SSyncFSM (Signal Synchronizing Finite State Machine), της οποίας η δουλειά είναι η παραγωγή των σημάτων συγχρονισμού με συγκεκριμένα διαστήματα Back Porch, Front Porch, Displaying και Signaling.

Τα παραπάνω διαστήματα οριοθετούνται από τον χρήστη με τη βοήθεια των παραμέτρων SBP, SFP, DISPLAY_TIME και SHOLD, αντίστοιχα, και αναφέρονται στον αριθμό των pixel που χρειάζεται να περάσουν για να προχωρήσει η FSM στην επόμενη κατάσταση.



Σχήμα 2: FSM inside SSyncFSM

4.2 Υλοποίηση:

Για την υλοποίηση των σημάτων συγχρονισμού της VGA στη συγκεκριμένη υλοποίηση, έχει χρησιμοποιηθεί η μονάδα ClockGenerator για τη δημιουργία ενός νέου ρολογιού στα 25MHz. Το ρολόι αυτό χρησιμοποιείται με σκοπό την αντιστοίχιση ενός pixel της VGA σε έναν θετικό παλμό ρολογιού. Συνεπώς, οι χρόνοι που χρησιμοποιούνται στον VGAController αναφέρονται σε διαστήματα μεγέθους 40ns, δηλαδή στο διάστημα εκπομπής ενός pixel.

	Label	**Time (s)**	**Clock Cycles (25 MHz)**
A	Scanline Time	32 µs	800 cycles
B	HSYNC Pulse Width	3.84 µs	96 cycles
C	Back Porch	1.92 µs	48 cycles
D	Display Time	25.6 µs	640 cycles
E	Front Porch	0.640 µs	16 cycles
	Total Frame Time	**16.67 ms**	**416,750 cycles**
P	VSYNC Pulse Width	64 µs	1,600 cycles
Q	Back Porch	928 µs	23,200 cycles
R	Active Video Time	15.36 ms	384,000 cycles
S	Front Porch	320 µs	8,000 cycles

Table 1: Timing Parameters and Corresponding Clock Cycles for 25 MHz Clock

Για την υλοποίηση του HSYNC, χρησιμοποιείται το instantiation της μονάδας SSyncFSM με το όνομα HSyncFSM, το οποίο βρίσκεται στο toplevel module (VGAController).

Για τον χρονισμό του σήματος, σύμφωνα με τις απαιτήσεις της VGA και της εργασίας, έχουν δοθεί οι εξής χρόνοι:

Parameter	**Value**
SHOLD	96
SBP	48
DISPLAY_TIME	640
SFP	16

Table 2: Table of Parameters and Values

Για την υλοποίηση του VSYNC, χρησιμοποιείται το instantiation της μονάδας SSyncFSM με το όνομα VSyncFSM, το οποίο βρίσκεται στο toplevel module (VGAController).

Για τον χρονισμό του σήματος, σύμφωνα με τις απαιτήσεις της VGA και της εργασίας, έχουν δοθεί οι εξής χρόνοι:

Parameter	**Value**
SHOLD	1600
SBP	23200
DISPLAY_TIME	384000
SFP	8000

Table 3: Table of Parameters and Values for VSYNC

Επιπλέον, η μονάδες ειδοποιούν τον χρήστη όταν η FSM βρίσκεται σε κατάσταση displaying με τα σήματα VDISP και HDISP, ώστε να μπορεί να χρησιμοποιηθεί από διαφορετικές μονάδες που είναι υπεύθυνες για την εξαγωγή δεδομένων των pixel από κάποια μνήμη.

4.3 Επαλήθευση HSYNC:

Για την επαλήθευση του σήματος, παρατηρούμε τις κυματομορφές της αντίστοιχης FSM και την τιμή του counter σε κάθε κατάσταση. Η ανάλυση που πραγματοποιήθηκε στις κυματομορφές έδειξε ότι η λειτουργία της μονάδας είναι σωστή, όπως επίσης και ο χρονισμός του σήματος HSYNC. Υπάρχει, επίσης, μια μονάδα testbench για τον έλεγχο της λειτουργίας της.

Επιπλέον, πραγματοποιήθηκε Error Analysis στις μετατροπές των χρόνων που δόθηκαν για την VGA σε χρόνους ρολογιού 25MHz. Από την ανάλυση αυτή φαίνεται ότι δεν υπάρχει κανένα σφάλμα που πρέπει να ληφθεί υπόψη.

	Time (s)	**Calculated Cycles**	**Rounded Cycles**	**Comment**
A	32 μs	800	800	Exact
B	3.84 μs	96	96	Exact
C	1.92 μs	48	48	Exact
D	25.6 μs	640	640	Exact
E	0.640 μs	16	16	Exact
P	64 μs	1600	1600	Exact
Q	928 μs	23,200	23,200	Exact
R	15.36 ms	384,000	384,000	Exact
S	320 μs	8,000	8,000	Exact

Table 4: Verification of Clock Cycles for a 25 MHz Clock (Cycle Time = 40 ns)

4.4 Επαλήθευση VSYNC:

Για την επαλήθευση του σήματος VSYNC, ακολουθήθηκε παρόμοια διαδικασία με αυτή για το HSYNC. Παρατηρήθηκαν οι κυματομορφές της αντίστοιχης FSM, καθώς και οι τιμές του counter σε κάθε κατάσταση. Η ανάλυση έδειξε ότι η λειτουργία της μονάδας είναι ορθή και ο χρονισμός του σήματος VSYNC ανταποκρίνεται στις απαιτήσεις της προδιαγραφής.

Για τον έλεγχο της λειτουργικότητας της μονάδας, δημιουργήθηκε και χρησιμοποιήθηκε ειδική testbench που επαληθεύει τη σωστή διαδοχή των καταστάσεων, καθώς και τις αντίστοιχες τιμές του counter.

Τέλος, πραγματοποιήθηκε Error Analysis για τις μετατροπές των χρόνων που δόθηκαν για την VGA σε χρόνους ρολογιού 25MHz. Η ανάλυση έδειξε ότι οι υπολογισμοί για τις τιμές του VSYNC (βλ. Πίνακα 3) είναι ακριβείς (βλ. Πίνακα 4) και δεν υπάρχει κανένα σφάλμα που να απαιτεί διόρθωση ή περαιτέρω ανάλυση.

4.5 Συμπεράσματα-Παρατηρήσεις:

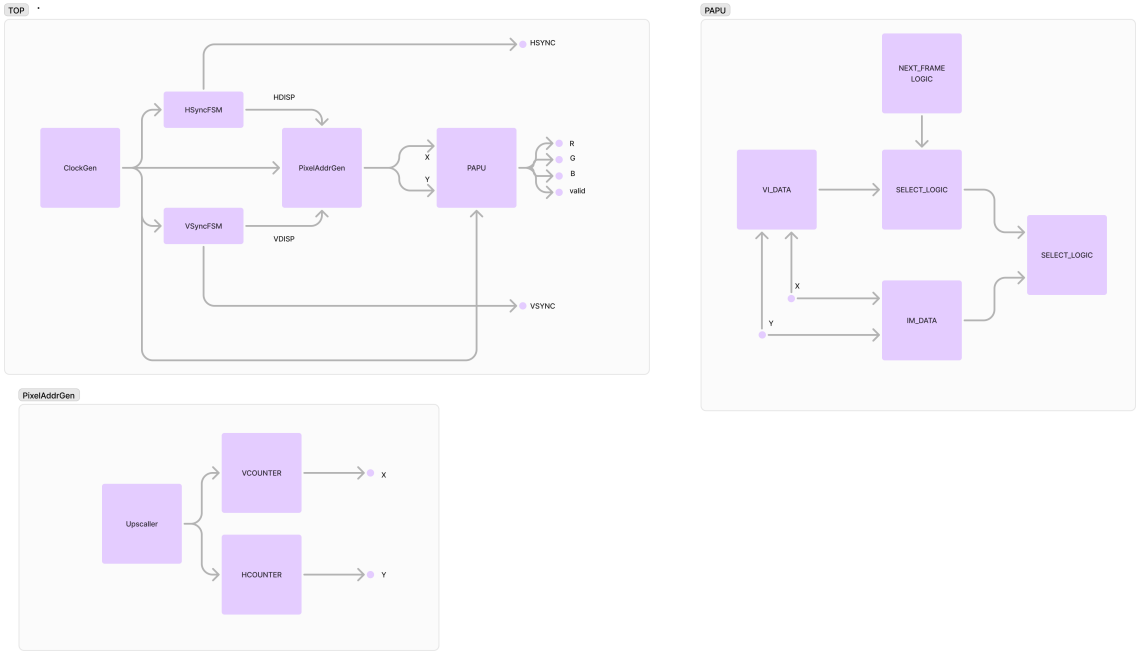
Η σχεδίαση και η δοκιμασία της λειτουργικότητας της μονάδας ήταν επιτυχής και φαίνεται να λειτουργεί όπως πρέπει αφού το Bitstream εξετάστηκε και στην πλακετα FPGA. Στην τελική έκδοση υπήρχε μόνο το παρακάτω Vivado Warning, το οποίο αγνοήθηκε:

[Constraints 18-5210] No constraints selected for write. Το παραπάνω αγνοήθηκε, καθώς φαίνεται να είναι κάποιο bug του Vivado, δεδομένου ότι έχει οριστεί constraints file στην τελική έκδοση.

5 Εικόνα και Βίντεο

5.1 Εικόνα Design Flow:

Το παρακάτω design υλοποιήθηκε με κύριο στόχο την αποφυγή πολύπλοκων μονάδων που εκτελούν πολλαπλά και διαφορετικά tasks για την εκτύπωση της εικόνας. Για αυτόν τον λόγο, οι μονάδες είναι περισσότερες από όσες ίσως απαιτούνται. Ωστόσο, αυτές έχουν πολύ συγκεκριμένες λειτουργίες, γεγονός που καθιστά ξεκάθαρο τον ρόλο τους στο κύκλωμα και διευκολύνει τόσο τη συντήρηση όσο και την ανάπτυξη επιπλέον κώδικα για την υποστήριξη sprite και video. Ένα επιπλέον πλεονέκτημα αυτής της προσέγγισης είναι ότι διευκολύνει την κατανόηση του κώδικα Verilog.

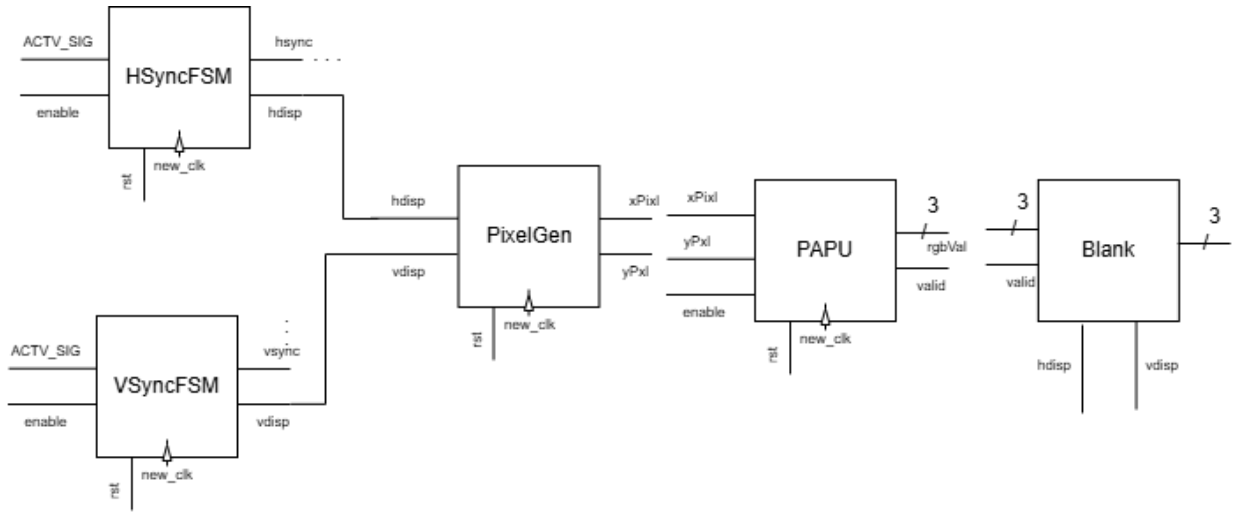


Σχήμα 3: Dataflow for Video and Image design

5.2 Υλοποίηση:

Για την υλοποίηση της εικόνας θα χρειαστεί ένα toplevel module που θα ελέγχει τις εισόδους της VGA, το οποίο υλοποιείται όπως φαίνεται στο Dataflow. Χρησιμοποιούνται οι FSM που αναλύθηκαν παραπάνω, καθώς και μια μονάδα PixelAddrGen, η οποία δημιουργεί, ανάλογα με τα σήματα VDISP και HDISP, τις συντεταγμένες από τις οποίες πρέπει να διαβάσει η μονάδα PAPU. Επιπλέον, στο σχήμα παρουσιάζονται αφαιρετικά οι τρόποι λειτουργίας των μονάδων PixelAddrGen και PAPU.

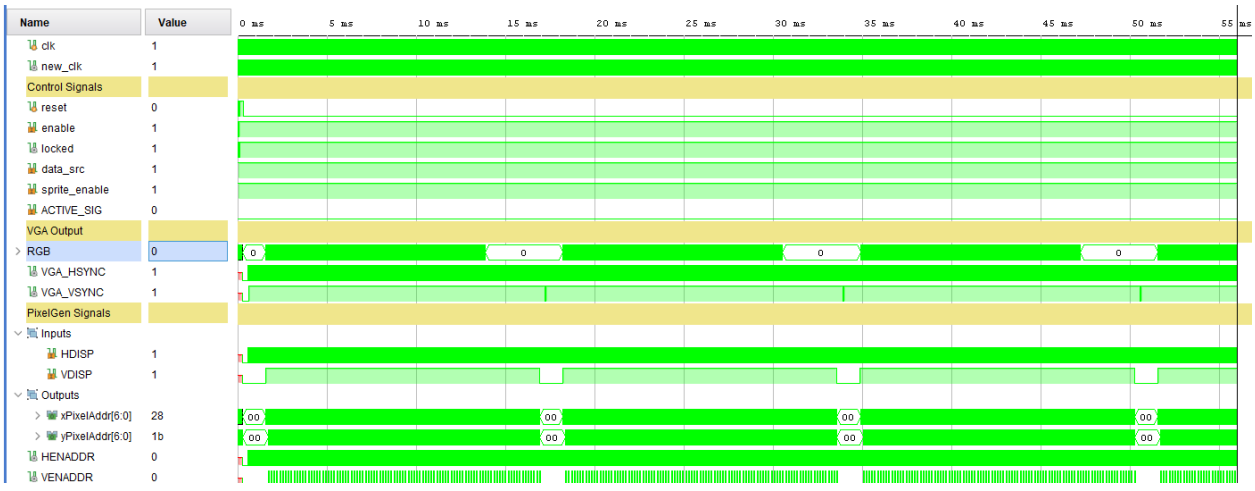
Η μονάδα δέχεται ένα εξωτερικό σήμα enable, το οποίο ενεργοποιεί οποιοδήποτε συνδυαστικό κύκλωμα σε όλες τις μονάδες. Εδώ αξίζει να σημειωθεί ότι το enable δεν πραγματοποιεί reset στο κύκλωμα· αυτή είναι ευθύνη του χρήστη. Περισσότερα σχετικά με αυτή τη συμπεριφορά αναλύονται στην ενότητα επαλήθευσης.



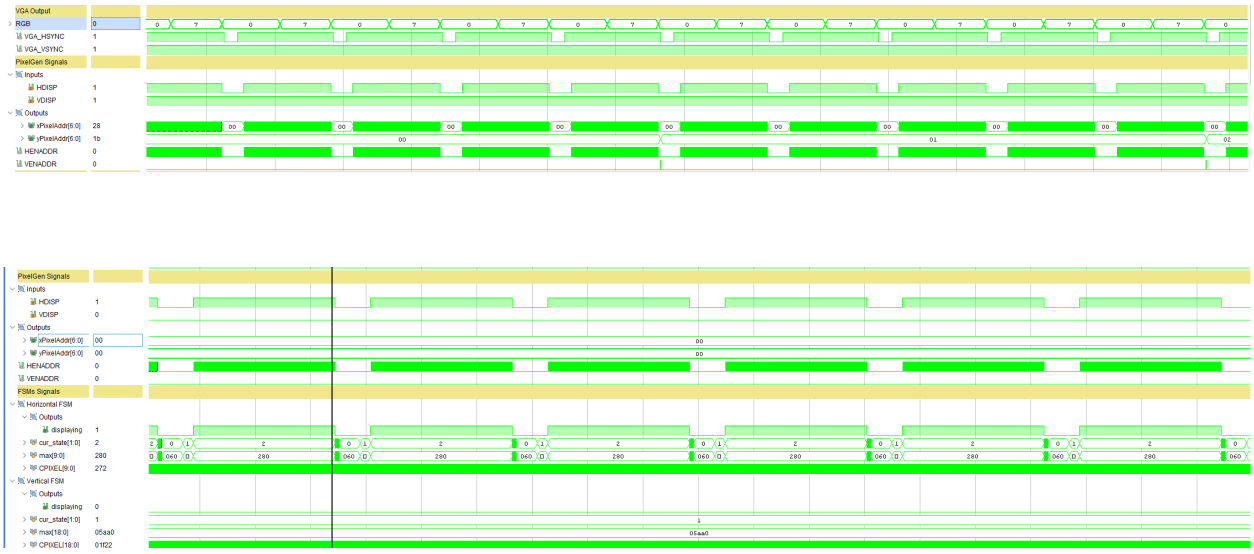
Σχήμα 4: Top level module

5.3 Επαλήθευση:

Για την επαλήθευση της σωστής απεικόνισης στη VGA, χρησιμοποιήθηκε η εικόνα που μας παραχωρήθηκε στην εργασία. Η εικόνα αυτή διευκολύνει την ανίχνευση φαινομένων όπως το stuttering και το τρεμόπαιγμα. Επιπλέον, χρησιμοποιήθηκε και η πρώτη ασπρόμαυρη εικόνα από το video που παραχωρήθηκε μαζί με τον κώδικα (οι κυματομορφές που παρουσιάζονται παρακάτω έχουν παραχθεί με την ασπρόμαυρη εικόνα).



Σχήμα 5: Κυματομορφές Top level VGAController module



Σχήμα 6: Zoomed

5.4 Βίντεο Υλοποίηση:

Για την υλοποίηση της αναπαραγωγής video, απαιτήθηκαν αλλαγές τόσο στη μονάδα PAPU όσο και στον VGAController. Συγκεκριμένα, προστέθηκε ένα επιπλέον σήμα, το οποίο καθορίζει τη λειτουργία του controller, επιτρέποντάς του να εναλλάσσεται μεταξύ των καταστάσεων Image (για την απεικόνιση στατικής εικόνας) και Video (για την απεικόνιση δυναμικού περιεχομένου).

Επιπλέον, λόγω του συγκεκριμένου σχεδιασμού, ο οποίος δεν αξιοποιεί την DDR2 μνήμη, ήταν απαραίτητο να γίνουν νέα instantiations στο PAPU. Αυτά τα instantiations περιλαμβάνουν τις πληροφορίες που σχετίζονται με κάθε εικόνα του video, διασφαλίζοντας έτσι ότι η σωστή εικόνα φορτώνεται και απεικονίζεται στη σωστή χρονική στιγμή.

Για τα πλαίσια της εργασίας έχει δημιουργηθεί ένα GIF με 13 εικόνες από το γνωστό video "Bad Apple", το οποίο πρέπει να προβάλλεται με ρυθμό περίπου 4 καρέ ανά δευτερόλεπτο ή 1 καρέ ανά 15 VSYNC. Παρόλα αυτά, σύμφωνα με υπολογισμούς από τη χωρητικότητα μνήμης BRAM από το εγχειρίδιο της FPGA, υπάρχει διαθέσιμος χώρος για περισσότερα από 60 καρέ.

Η παραπάνω προσέγγιση επιλέχθηκε για να παρακαμφθεί η πολυπλοκότητα χρήσης της DDR2, αν και απαιτεί επιπλέον μονάδες και περισσότερη μνήμη BRAM για την αποθήκευση των δεδομένων κάθε εικόνας. Παρά τις αυξημένες απαιτήσεις πόρων, η απλοποίηση της διαδικασίας καθιστά τη σχεδίαση πιο ευέλικτη και κατανοητή. (+ Δεν έβγαλα άκρη με το MiG, IP Config :))

5.5 Sprite Υλοποίηση:

Η υλοποίηση των Sprite έγινε με στόχο την απλότητα και την ευκολία χρήσης, προκειμένου να διευκολυνθεί η ενσωμάτωσή τους στο γενικότερο σύστημα της VGA. Κάθε sprite είναι μια αυτόνομη μονάδα, η οποία μπορεί να δεχτεί εισόδους για τη μετακίνηση της στην οθόνη, με κάποια λογική που καθορίζει την κίνησή της.

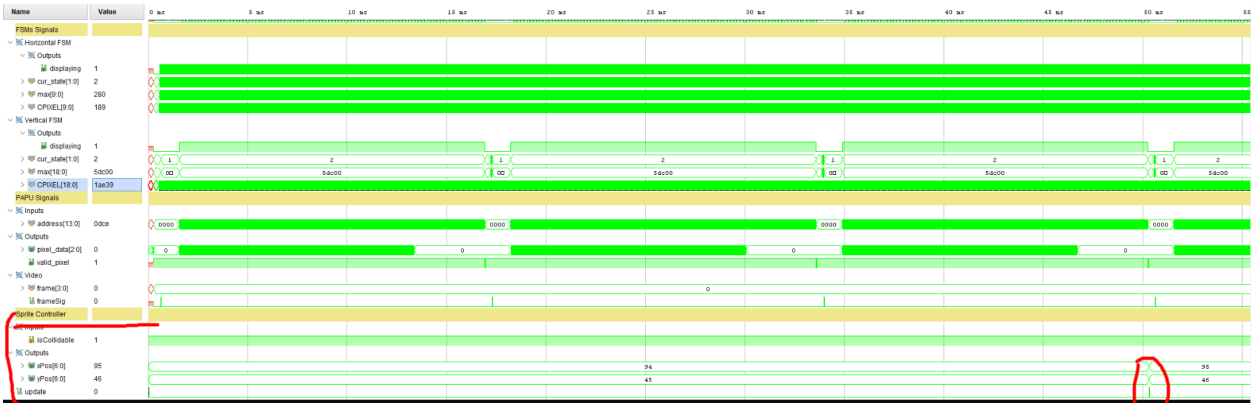
Τα sprites επιπλέον έχουν έναν συγκεκριμένο μέγεθος που μπορούν να πάρουν, και η εικόνα που εμφανίζουν είναι προκαθορισμένη σε μια μνήμη ROM256x5. Είναι σημαντικό να αναφερθεί ότι για την υλοποίηση των μνημών ROM, χρησιμοποιήθηκε η εντολή initial, η οποία είναι συνθεσίμη σε αυτήν την περίπτωση και αποτελεί τον σωστό τρόπο δημιουργίας και αρχικοποίησης μιας μνήμης ROM.

Η χρήση της εντολής initial επιτρέπει την αποτελεσματική φόρτωση των προκαθορισμένων δεδομένων στην ROM, εξασφαλίζοντας τη σωστή και έγκαιρη αρχικοποίηση της μνήμης κατά τη διάρκεια της σύνθεσης του κυκλώματος.

Για τους σκοπούς της εργασίας, παρέχεται μια απλή μονάδα διαχείρισης των διαφόρων sprite, το MovementController, το οποίο υλοποιεί την κίνηση στην οθόνη με μεταβλητή ταχύτητα και αρχική τοποθεσία. Το MovementController επιτρέπει την ευέλικτη και δυναμική κίνηση των sprite, ενώ ταυτόχρονα υποστηρίζει τον περιορισμό της κίνησης εντός της οθόνης.

Επιπλέον, η μονάδα υλοποιεί λογική για την ανίχνευση και διαχείριση κρούσης με τους τοίχους της VGA, αποτρέποντας την έξοδο του sprite από τα όρια της οθόνης. Αν το sprite φτάσει στα όρια της οθόνης, το MovementController αναλαμβάνει να αντιστρέψει την κατεύθυνση της κίνησης ή να εκτελέσει κάποια άλλη προγραμματισμένη ενέργεια, διασφαλίζοντας τη σωστή εμφάνιση και κίνηση του sprite κατά τη διάρκεια της αναπαραγωγής.

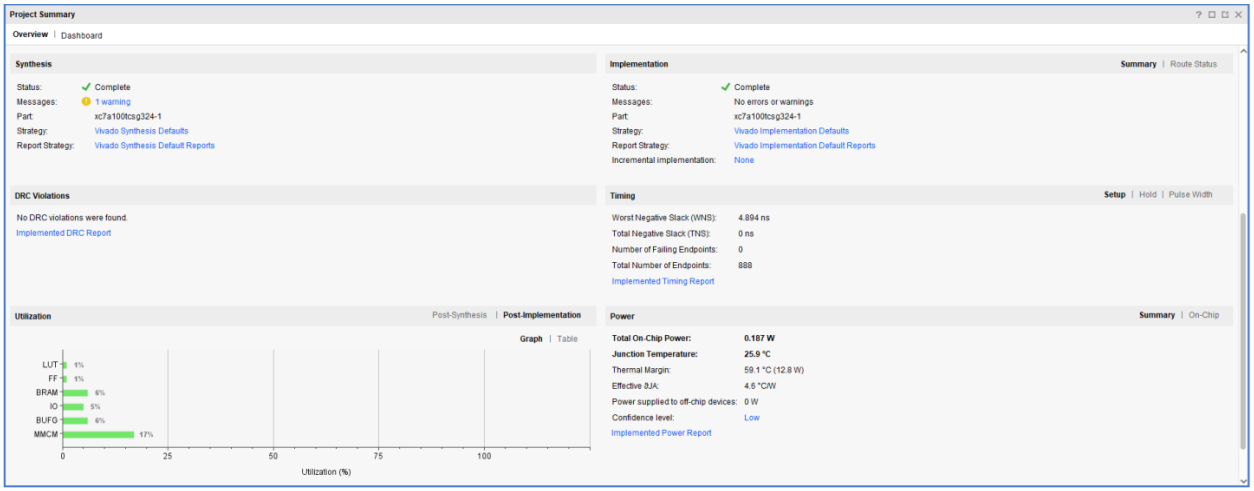
Η χρήση μιας τέτοιας μονάδας παρέχει μεγάλη ευχέρεια στο σχεδιασμό και τη διαχείριση πολλαπλών sprite, ενώ επιτρέπει τη διατήρηση της απλότητας και της ευκολίας στην εφαρμογή της λογικής κίνησης και κρούσης.



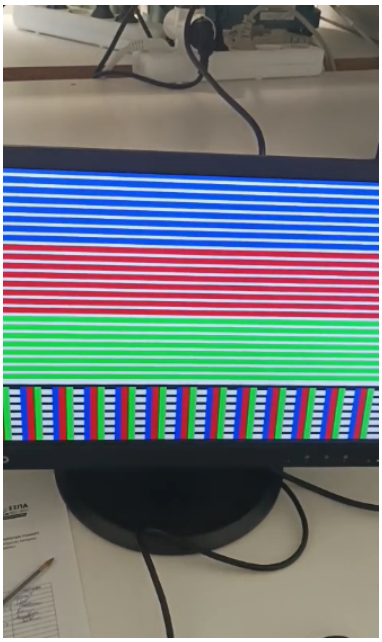
5.6 Σημειώσεις:

Στο Design χρησιμοποιείται συγχρονισμένο reset, καθώς στο Vivado εμφανιζόταν warning ότι η διεύθυνση (ADDRESS) από τις BRAM μηδενιζόταν ασύγχρονα. Στη συνέχεια, προστέθηκε ένας synchronizer στο reset για τις BRAM, αλλά η αλλαγή αυτή δεν έγινε αντιληπτή από το Vivado, με αποτέλεσμα να συνεχίζει να εμφανίζεται το ίδιο warning. Για τον λόγο αυτό, ο σχεδιασμός (design) τροποποιήθηκε ώστε να χρησιμοποιείται συγχρονισμένο reset, προκειμένου να αποφεύγονται τα warnings του Vivado.

Υπάρχουν τα source files από τις εικόνες και τα sprites που χρησιμοποιήθηκαν, καθώς και το video που δείχνει το video και την εικόνα να τρέχουν στην FPGA. Η κίνηση του sprite προστέθηκε αργότερα και, ενώ δεν είναι παρόν στο video, εκτελείται κανονικά κατά την εκτέλεση του συστήματος.



Σχήμα 7: Screenshot Post-Implementation Timing



Σχήμα 8: Image Showcase