Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων
Εργαστηριο Μικροϋπολογιστων και Ψηφιακων Συστηματων

Ψηφιακά Συστήματα VLSI

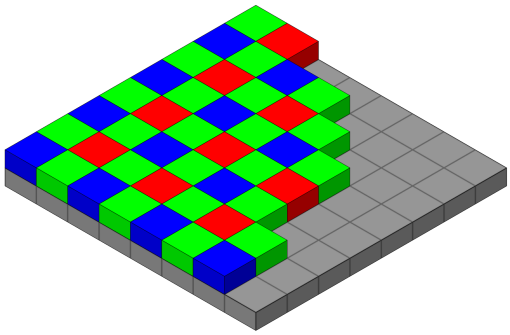
5η Εργαστηριακή Άσκηση

"Υλοποίηση Debayering Φίλτρου με AXI διεπαφή σε Zynq SoC FPGA"

Μέρος Α

Εισαγωγή

Μία συστοιχία χρωματικών φίλτρων (Color Filter Array) αποτελεί ένα μωσαϊκό από μικροσκοπικά χρωματικά φίλτρα, το οποίο τοποθετείται πάνω από τα pixels των αισθητήρων κάμερας ώστε να συλλάβει τις πληροφορίες χρώματος. Ένα από τα πιο ευρέως χρησιμοποιούμενα μωσαϊκά είναι το Bayer [1], το οποίο παρουσιάζεται στο Σχήμα 1. Η κατανομή των χρωματικών φίλτρων στο μωσαϊκό Bayer είναι ως εξης: 50% πράσινα, 25% κόκκινα, και 25% μπλε.



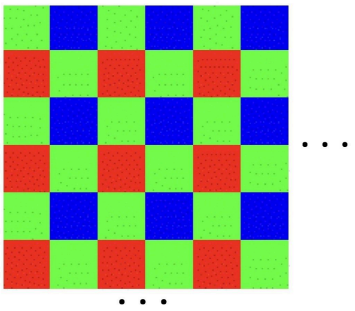
Σχήμα 1: Η συστοιχία χρωματικών φίλτρων Bayer [1].

Για κάθε pixel αποθηκεύεται μόνο μία από τις τρεις χρωματικές συνιστώσες, η οποία προσδιορίζεται από το αντίστοιχο χρωματικό φίλτρο. Για παράδειγμα, αν το χρωματικό φίλτρο που έχει τοποθετηθεί πάνω από το pixel είναι το μπλε, τότε για το συγκεκριμενο pixel θα αποθηκευτεί μόνο η μπλε συνιστώσα. Υπάρχουν τέσσερα διαφορετικά πρότυπα Bayer, τα οποία προσδιορίζονται από το χρώμα των τεσσάρων pixels (2x2 γειτονιά) στην πάνω αριστερή γωνία της εικόνας [2]: GBRG, GRBG, BGGR, RGGB.
Για να μετατραπεί η εικόνα που προκύπτει από το μωσαϊκό Bayer σε RGB, θα πρέπει να υπολογιστούν οι δύο χρωματικές συνιστώσες που λείπουν για κάθε pixel. Η διαδικασία αυτή ονομάζεται Debayering ή Demosaicing [3].

[1] https://en.wikipedia.org/wiki/Bayer_filter
[2] https://www.mathworks.com/help/images/ref/demosaic.html
[3] https://en.wikipedia.org/wiki/Demosaicing

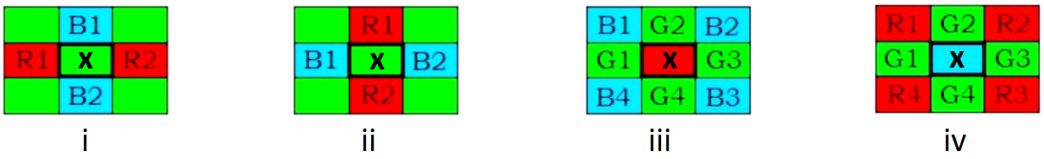
Ζητούμενο Εργαστηριακής Άσκησης

Στα πλαίσια της παρούσας Εργαστηριακής Άσκησης, καλείστε να υλοποιήσετε το φίλτρο Debayering, το οποίο θα μετατρέπει την Bayer εικόνα σε RGB υπολογίζοντας τους μέσους όρους των γειτονικών pixels σε 3x3 γειτονιές. Θεωρούμε ότι το μωσαικό Bayer ακολουθεί το πρότυπο GBRG, δηλαδή η Bayer εικόνα ακολουθεί το μοτίβο που παρουσιάζεται στο Σχήμα 2.



Σχήμα 2: Το πρότυπο GBRG της συστοιχίας χρωματικών φίλτρων Bayer [1].

Με βάση την κατανομή των χρωματικών συνιστωσών, υπάρχουν συνολικά τέσσερις διαφορετικές περιπτώσεις όπου γνωρίζουμε την μία χρωματική συνιστώσα και θέλουμε να βρούμε τις άλλες δύο. Οι περιπτώσεις αυτές παρουσιάζονται στο Σχήμα 3.



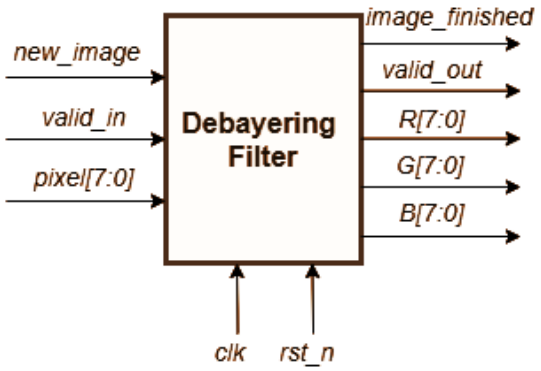
Σχημα 3: Οι πιθανές 3x3 γειτονιές της συστοιχίας χρωματικών φίλτρων Bayer.

Με βάση το Σχήμα 3, θα πρέπει να πραγματοποιηθούν οι ακόλουθοι υπολογισμοί για το κεντρικό pixel X:

- Αν αυτό είναι πράσινο (περ. i, ii), η κόκκινη (μπλε) χρωματική του συνιστώσα υπολογίζεται ως ο μέσος όρος των δύο κόκκινων (μπλε) γειτονικών pixels.
- Αν αυτό είναι κόκκινο (περ. iii), η πράσινη (μπλε) χρωματική του συνιστώσα υπολογίζεται ως ο μέσος όρος των τεσσάρων πράσινων (μπλε) γειτονικών pixels.
- Αν αυτό είναι μπλε (περ. iv), η πράσινη (κόκκινη) χρωματική του συνιστώσα υπολογίζεται ως ο μέσος όρος των τεσσάρων πράσινων (κόκκινων) γειτονικών pixels.

Προτεινόμενη Αρχιτεκτονική & Σχεδίαση

Η υλοποίηση θα πραγματοποιηθεί σε FPGA με τη γλώσσα περιγραφής υλικού VHDL. Ως εργαλείο θα χρησιμοποιήσετε το Xilinx Vivado, στο οποίο θα επιλέξετε ως board το Zybo. Οι είσοδοι/έξοδοι (I/O) του κορυφαίου σε ιεραρχία module της στοχευμένης αρχιτεκτονικής παρουσιάζεται στο Σχήμα 4.



Σχήμα 4: Το I/O της στοχευμένης αρχιτεκτονικής του Debayering φίλτρου.

<u>Επεξήγηση Σημάτων</u>:

- ***clk***: το σήμα ρολογιού, με βάση το οποίο συγχρονίζεται όλο το κύκλωμα.
- ***rst_n***: το ασύγχρονο σήμα μηδενισμού, με βάση το οποίο μηδενίζονται οι καταχωρητές και οι έξοδοι του κυκλώματος. Ενεργοποιείται στο λογικό '0'. Κατά τη λειτουργία του κυκλώματος, είναι απενεργοποιημένο (βρίσκεται μόνιμα στο λογικό '1').
- ***pixel[7:0]***: σήμα εισόδου των 8 bits, το οποίο τροφοδοτεί το κύκλωμα με τα 8-bit pixels της Bayer εικόνας.
- ***valid_in***: σήμα εισόδου, το οποίο ενεργοποιείται (στο λογικό '1') για να υποδείξει ότι η τιμή του σήματος *pixel* είναι έγκυρη. Ενεργοποιείται για αριθμό κύκλων ρολογιού ίσο με το μέγεθος της εικόνας. Για παράδειγμα, για *N*x*N* εικόνα, θα ενεργοποιηθεί για N^2 κύκλους ρολογιού.
- ***new_image***: σήμα εισόδου, το οποίο ενεργοποιείται (στο λογικό '1') για 1 κύκλο ρολογιού ταυτόχρονα με το πρώτο *valid_in* σήμα, ώστε να υποδείξει ότι θα αρχίσει η επεξεργασίας μιας νέας εικόνας.
- ***R[7:0], G[7:0], B[7:0]***: σήματα εξόδου των 8 bits, τα οποία υποδεικνύουν τις χρωματικές συνιστώσες. Για κάθε pixel, εμφανίζονται ταυτόχρονα στην έξοδο και οι τρεις χρωματικές συνιστώσες. Επίσης, με βάση την λειτουργία του Debayering φιλτρου, για κάθε pixel, κάποιο από τα *R, G*, και *B* θα έχει την τιμή του σήματος *pixel*, ενώ τα άλλα δύο θα υπολογίζονται με τους μέσους όρους των γειτονικών pixels.
- ***valid_out***: σήμα εξόδου, το οποίο ενεργοποιείται (στο λογικό '1') για να υποδείξει ότι οι τιμές των σημάτων *R, G, B* είναι έγκυρες. Ενεργοποιείται για αριθμό κύκλων ρολογιού ίσο με το μέγεθος της εικόνας. Για παράδειγμα, για *N*x*N* εικόνα, θα ενεργοποιηθεί για N^2 κύκλους ρολογιού.

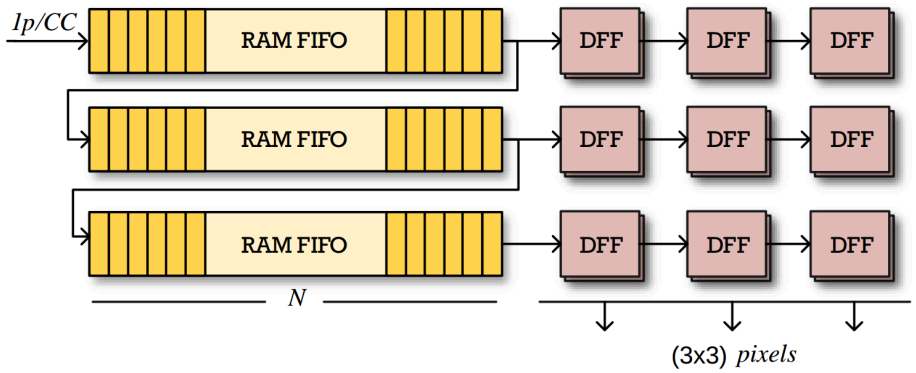
- ***image_finished***: σήμα εξόδου, το οποίο ενεργοποιείται (στο λογικό ‘1’) για 1 κύκλο ρολογιού ταυτόχρονα με το τελευταίο *valid_out* σήμα, ώστε να υποδείξει ότι η επεξεργασία της εικόνας τελείωσε.

Λειτουργίες Κυκλώματος:

1. Η εικόνα σκανάρεται και δίνεται ως είσοδο στο κύκλωμα (μέσω του σήματος *pixel*) από τα αριστερά προς τα δεξιά και από πάνω προς τα κάτω (raster scan ordering). Το κύκλωμα μας παίρνει ως είσοδο ένα pixel ανα κύκλο ρολογιού σε μορφή streaming. Δηλαδή, σε N^2 συνεχόμενους κύκλους ρολογιού, έχει λάβει όλα τα pixels μιας *N*x*N* εικόνας.
2. Για τη δημιουργία γειτονιών για τα ακριανά pixels (πρώτη και τελευταία γραμμή/στήλη της εικόνας), θεωρούμε μηδενικά pixels για τους γείτονες που βγαίνουν εκτός ορίων της εικόνας.
3. Σε περίπτωση που ενεργοποιηθεί το σήμα *rst_n*, θα πρέπει, όταν αυτό απενεργοποιηθεί, να ξεκινήσει από την αρχή η επεξεργασίας μιας νέας εικόνας. Δηλαδή, το κύκλωμα θα περιμένει την ενεργοποίηση του σήματος *new_image*, να λάβει τα pixels από την αρχή, κ.ο.κ.

Λεπτομέρειες Σχεδίασης:

1. Δεδομένου ότι το Debayering φίλτρο λαμβάνει ένα pixel ανα κύκλο ρολογιου (*1p/CC*) με raster scan ordering και ότι η επεξεργασία (υπολογισμός μέσων όρων) πραγματοποιείται σε 3x3 γειτονιές, θα πρέπει να χρησιμοποιήσετε έναν μετατροπέα σειριακού-σε-παράλληλο (serial-to-parallel converter). Το κύκλωμα του μετατροπέα θα περιλαμβάνει προσωρινή μνήμη (buffers) και μία συστοιχία από καταχωρητές (registers), ώστε να αποθηκεύει τα pixels εισόδου και να τροφοδοτεί την είσοδο του κυκλώματος υπολογισμού των μέσων όρων με όλα τα pixels της 3x3 γειτονιάς παράλληλα. Η υλοποίηση του μετατροπέα σειριακού-σε-παράλληλο μπορεί να γίνει με τη χρήση FIFO μνημών και Flip-Flops (FFs), όπως φαίνεται στο Σχήμα 5. Για τις FIFO μνήμες, οι οποίες θα πρέπει να έχουν βάθος *N*, μπορείτε να χρησιμοποιήσετε το έτοιμο IP της Xilinx από το Vivado.



Σχήμα 5: Η προτεινόμενη αρχιτεκτονική του μετατροπέα σειριακού-σε-παράλληλο.

2. Θεωρώντας το πρότυπο Bayer GBRG (Σχήμα 2), γνωρίζουμε εκ των προτέρων ότι το 1ο έγκυρο pixel που θα λάβουμε ως είσοδο είναι G, το 2ο είναι B, το 3ο είναι G, κ.ο.κ. Αντίστοιχα, για την επόμενη γραμμή της εικόνας, το 1ο είναι R, το 2ο είναι G, το 3ο είναι R, κ.ο.κ. Το κύκλωμα χρειάζεται να γνωρίζει τι χρώμα είναι το κάθε pixel, ώστε να πραγματοποιήσει τους αντίστοιχους υπολογισμούς για τις χρωματικές συνιστώσες που λείπουν (Σχήμα 3). Αυτό επιτυγχάνεται με την χρήση μετρητών (counters), τόσο για τις γραμμές όσο και για τα pixels της κάθε γραμμής. Ο μετρητής γραμμών υποδεικνύει το μοτίβο (GBG ή RGR) που θα έχουν τα επόμενα *N* pixels, ενώ ο μετρητής pixels υποδεικνύει το χρώμα του pixel.
3. Προτείνεται να υλοποιήσετε μία μηχανή πεπερασμένων καταστάσεων (finite-state machine), ώστε να ελέγχετε και να συγχρονίζετε το κύκλωμα σας. Η μηχανή αυτή θα παράγει σήματα ελέγχου, τα οποία θα συντονίζουν την λειτουργία του κυκλώματος.
4. Η υλοποίηση πρέπει να είναι παραμετρική ως προς το μέγεθος της εικόνας (*N*x*N*), όπου *N* = 32, 64, 128, κτλ, μέσω της χρήσης των generics της VHDL [4].

Επαλήθευση Κυκλώματος:

1. Για να ελέγξετε την ορθή λειτουργία του κυκλώματός σας, μπορείτε να υλοποιήσετε σε λογισμικό (MATLAB/C/Python) ένα απλό πρόγραμμα που θα διαβάζει ένα .txt αρχείο με *N*x*N* τυχαία 8-bit pixels, θα υπολογίζει τους μέσους όρους των γειτονιών όπως το Debayering φίλτρο, και θα αποθηκεύει σε ένα .txt αρχείο τις R, G, B τιμές για τα *N*x*N* pixels. Στη συνέχεια, θα συγκρίνετε το αρχείο αυτό με τις έγκυρες τιμές των σημάτων *R, G, B* που παράγει το κύκλωμα κατά την προσομοίωση με το ίδιο αρχείο εισόδου.
2. Στο testbench, μπορείτε να τροφοδοτείτε το κύκλωμα σας με pixels που διαβάζονται από αρχείο με χρήση της βιβλιοθήκης TextIO [5]. Αντίστοιχα, μπορείτε να γράψετε σε ένα αρχείο τις έγκυρες τιμές των σημάτων *R, G, B*, έτσι ώστε να το συγκρίνετε απευθείας με το αρχείο που παράγει το λογισμικό.

[4] https://www.ics.uci.edu/~jmoorkan/vhdlref/generics.html
[5] https://surf-vhdl.com/read-from-file-in-vhdl-using-textio-library/

Μέρος Β

Προγραμματισμός SoC FPGA

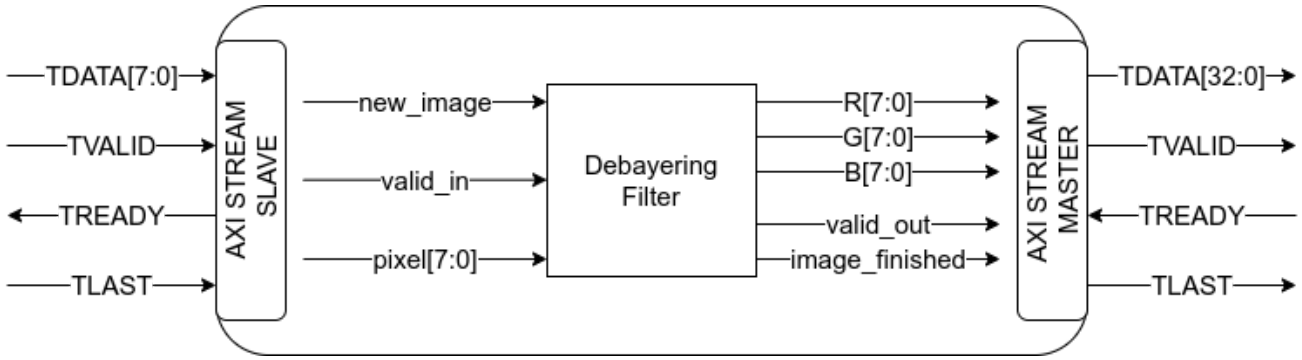
Στο Μέρος Β αυτής της εργαστηριακής άσκησης καλείστε να προγραμματίσετε την αναπτυξιακή πλακέτα ZYBO, ώστε να υλοποιεί ένα Debayering φίλτρο, στο οποίο τα δεδομένα εισόδου θα αποστέλλονται από τον ενσωματωμένο επεξεργαστή (ARM) προς το FPGA για επεξεργασία, και αντίστροφα για τα αντίστοιχα αποτελέσματα. Η επικοινωνία επεξεργαστή-FPGA θα βασίζεται στο πρωτόκολλο AXI. Η υλοποίηση του συστήματος χωρίζεται στα παρακάτω βήματα:

1) Εισαγωγή του ZYNQ Processing System (PS).
2) Εισαγωγή της **AXI4-Stream** διεπαφής στο FPGA (PL) για την πραγματοποίηση της επικοινωνίας ARM-FPGA.
3) Διασύνδεση του PS με PL.
4) Σύνθεση και υλοποίηση του συστήματος και παραγωγή του bitstream αρχείου.
5) Εξαγωγή της περιγραφής του συστήματος και δημιουργία της εφαρμογής.
6) Προγραμματισμός του ZYNQ SoC FPGA και εκτέλεση της εφαρμογής.

Εφαρμογή

Σε αυτή την άσκηση καλείστε να συνδέσετε το Debayering φίλτρο που έχετε ήδη υλοποιήσει στο Μέρος Α της άσκησης με AXI4-Stream διεπαφές (AXI4-Stream Slave/Master Interface). Η διασύνδεση των σημάτων εισόδου και εξόδου του φίλτρου με την AXI διεπαφή να υλοποιηθεί όπως φαίνεται στο Σχήμα 6. Συγκεκριμένα :

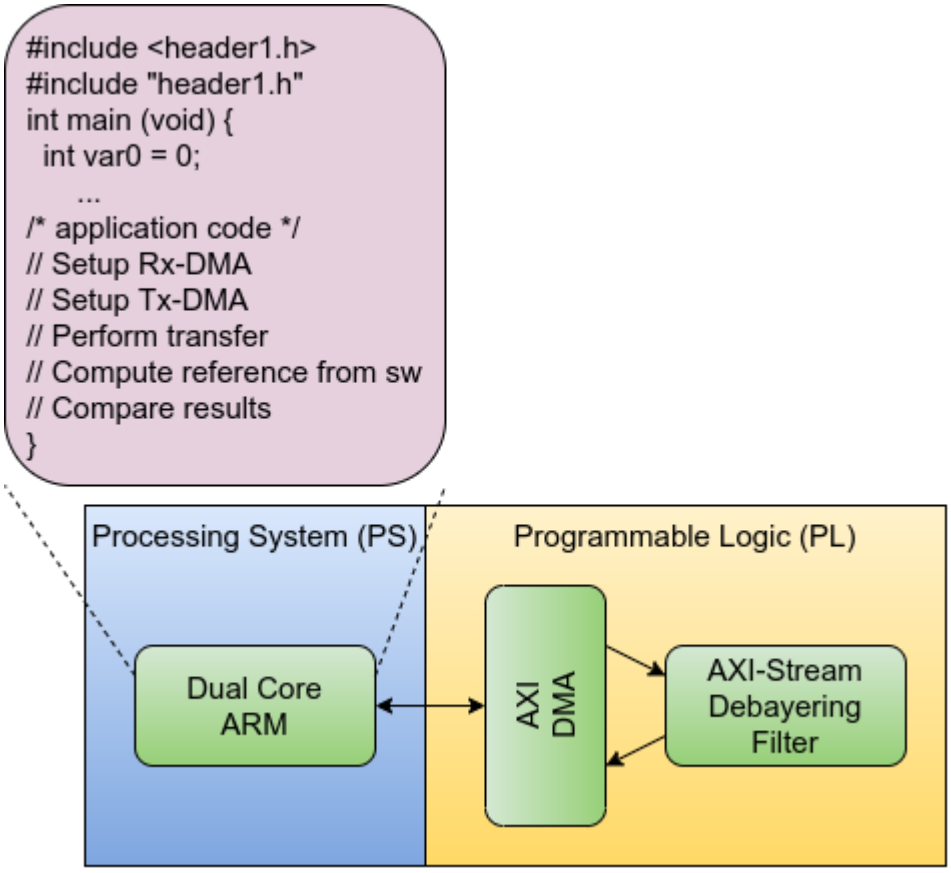
- `Slave_TDATA[7:0]  = pixel[7:0]`
- `Slave_TVALID  = valid_in`
- `Master_TDATA[32:0] = “00000000” & R[7:0] & G[7:0] & B[7:0]`
- `Master_TVALID  = valid_out`
- `Master_TLAST  = image_finished`



Σχήμα 6: Αρχιτεκτονική Debayering φίλτρου με διεπαφή AXI4-Stream.

Επιπλέον να αναπτύξετε την ανάλογη εφαρμογή λογισμικού για την αποστολή των σημάτων εισόδου και την λήψη των σημάτων εξόδου του φίλτρου από τον ενσωματωμένο επεξεργαστή. Η Η επικοινωνία μεταξύ PS-PL θα υλοποιείται μέσω Direct-Memory Access (DMA) λογικής. Η εφαρμογή λογισμικού θα είναι υπεύθυνη για την προετοιμασία του DMA ώστε να στέλνει τα pixel προς το φίλτρο και να δέχεται τα αποτελέσματα. Επίσης, η εφαρμογή λογισμικού θα υλοποιεί και reference software το οποίο θα υπολογίζει και αυτό τα αποτελέσματα του Debayering φίλτρο και

θα τα συγκρίνει με τα αποτελέσματα που θα λαμβάνονται από το FPGA. Η τελική αρχιτεκτονική του συνολικού συστήματος που καλείστε να παραδώσετε παρουσιάζεται στο Σχήμα 7.



Σχήμα 7: Συνολική αρχιτεκτονική συστήματος.

Εξέταση Εργαστηριακής Άσκησης - Μέρος Α

Θα κληθείτε να απαντήσετε σε ερωτήματα σχετικά με την σχεδίαση του κυκλώματος, το εργαλείο Vivado, καθώς και να επιδείξετε την ορθή λειτουργία του κυκλώματος μέσω προσομοίωσης για εικόνα εισόδου που θα σας δοθεί την ημέρα της εξέτασης.

Εξέταση Εργαστηριακής Άσκησης - Μέρος Β

Να υλοποιήσετε και να επιδείξετε στο ZYBO την παραπάνω εφαρμογή λαμβάνοντας υπόψη όλες τις λειτουργίες όπως αυτές είχαν καθοριστεί στο Μέρος Α της εργαστηριακής άσκησης και για **εικόνα 1024x1024 pixels**.

Παραδοτέα Εργαστηριακής Άσκησης

Καλείστε να παραδώσετε στο HELIOS ένα zip αρχείο που να περιέχει τα ακόλουθα:

- τον VHDL κώδικα του κυκλώματος
- τον VHDL κώδικα του testbench του κυκλώματος
- ένα PDF αρχείο (έως 2 σελίδες), το οποίο θα περιλαμβάνει:
    - το μπλοκ διάγραμμα του κυκλώματος
    - μία σύντομη επεξήγηση των βασικών δομικών μονάδων
    - το διάγραμμα της μηχανής πεπερασμένων καταστάσεων
    - την καταγραφή και ανάλυση των πόρων του FPGA (LUT, DFF, DSP, RAMB) που χρησιμοποιήθηκαν για N = 64, 128 (μετά το στάδιο Place & Route)
    - τον υπολογισμό της καθυστέρησης (latency) του κυκλώματος σε κύκλους ρολογιού

- ○ τον υπολογισμό της απόδοσης (throughput) σε έγκυρες εξόδους ανα κύκλο ρολογιού

<u>Παρατηρήσεις</u>:

1) **Η Εργαστηριακή Άσκηση θα παρουσιαστεί την Τετάρτη 11 Μαΐου 2022**. Παρακαλείστε να την έχετε μελετήσει μέχρι τότε, καθώς και να έχετε συγκεντρώσει τυχόν απορίες.
2) Για διευκόλυνση των φοιτητών στην προετοιμασία της εργαστηριακής άσκησης παρέχετε αυτοματοποιημένος τρόπος δημιουργίας project στο Vivado 2018.2.1. Για να το χρησιμοποιήσετε ακολουθήστε τα παρακάτω βήματα:
    a) Κατεβάζετε και κάνετε export τα αρχεία του dvlsi2021_lab5.zip
    b) Εκτέλεση του Vivado.
    c) Στο αρχικό παράθυρο επιλέγετε: Tools > Run Tcl Script …
    d) Στο παράθυρο που θα ανοίξει πηγαίνετε στο φάκελο που έχετε κάνει export και μέσα στο φάκελο scripts (πχ. /Desktop/dvlsi2021_lab5/scripts) επιλέγετε το αρχείο dvlsi2021_lab5_prj.tcl. Πατάτε *OK*. Το Vivado αυτόματα θα δημιουργήσει ένα καινούργιο project για να υλοποιήσετε την 5η εργαστηριακή άσκηση.
    e) Από αυτό το σημείο και μετά μπορείτε να προσθέσετε τα δικά σας αρχεία.