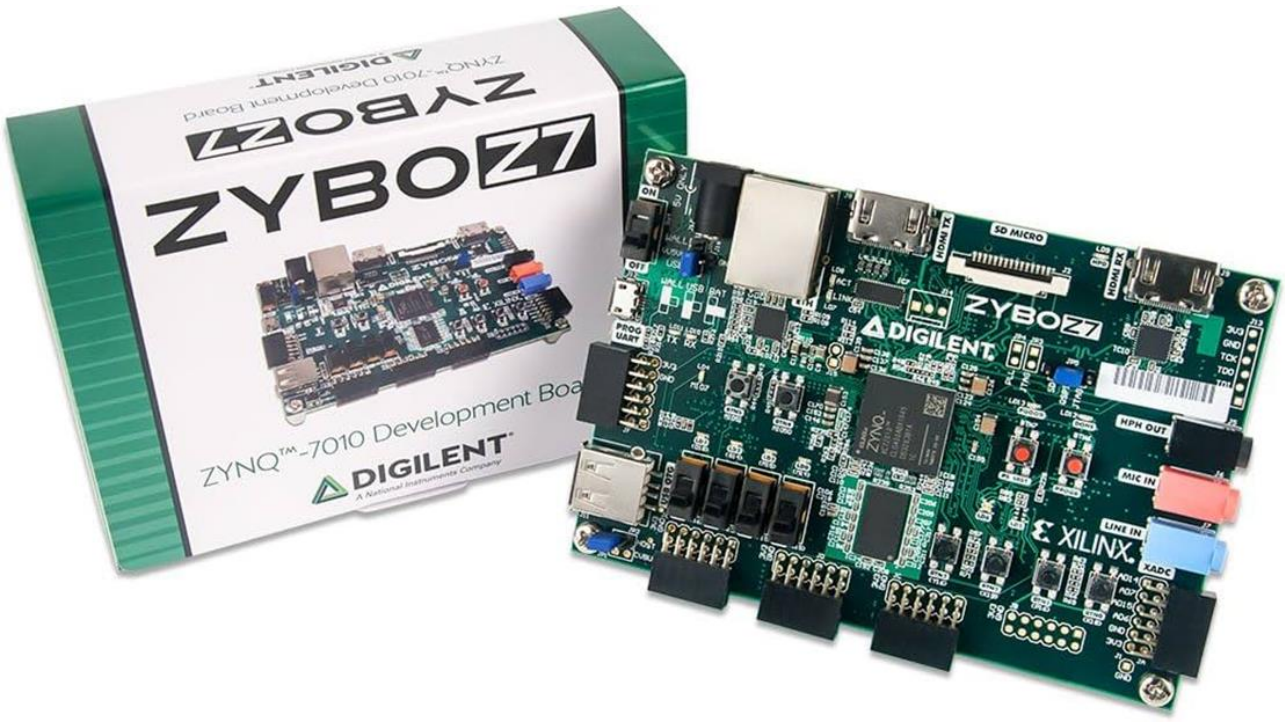
Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών



Μάθημα: Ψηφιακά Συστήματα VLSI

Θέμα: 6η Εργαστηριακή Άσκηση – Υλοποίηση Debayering Φίλτρου με AXI διεπαφή σε Zynq SoC FPGA

Εξάμηνο: 8ο

Ομάδα: 6

Συνεργάτες:

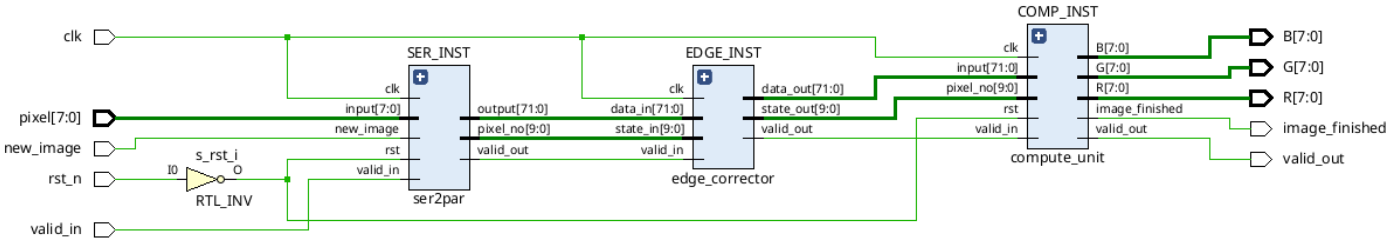
- Ακύλας Αντώνιος 03121152
- Κουμπιάς Ιωάννης-Χρυσοβαλάντης 03121053

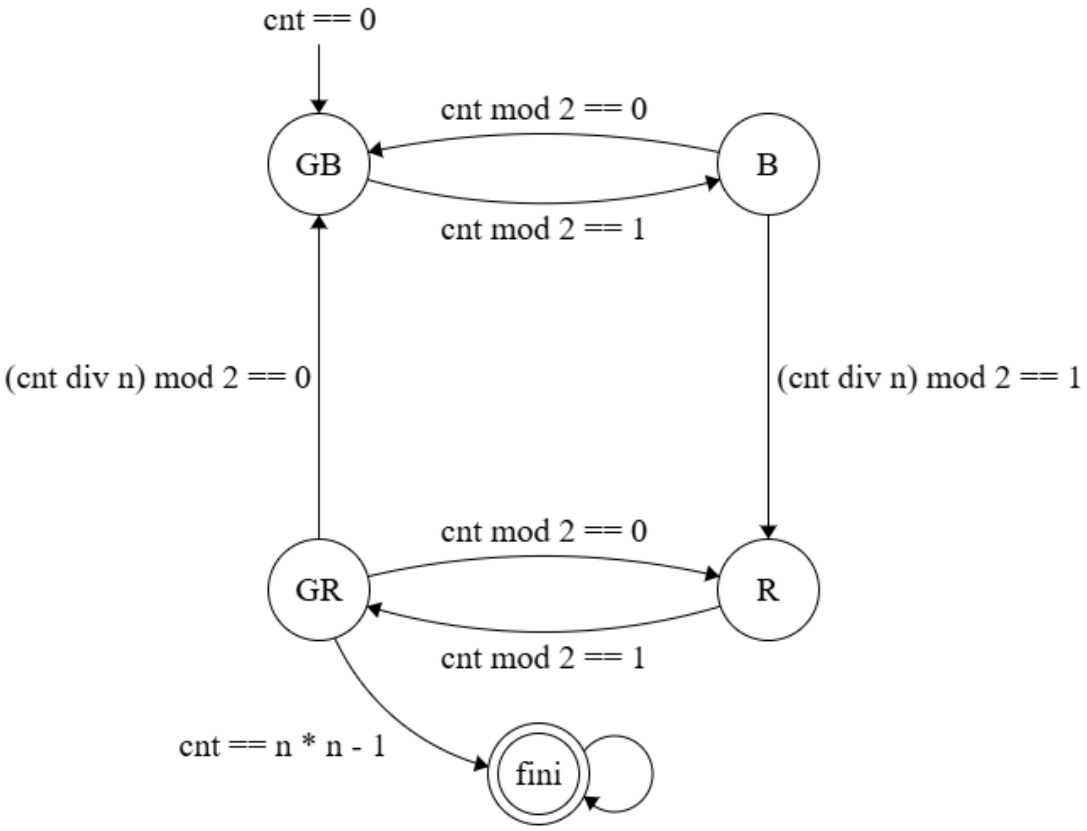
Εισαγωγή

Στην συγκεκριμένη εργαστηριακή άσκηση κληθήκαμε να σχεδιάσουμε με VHDL ένα GBRG Debayering Filter για την μετατροπή μιας Bayer εικόνας σε RGB υπολογίζοντας τους μέσους όρους των γειτονικών pixels σε παράθυρα 3x3. Στην συνέχεια αξιοποίωντας το design μας, προγραμματίσαμε την πλακέτα ZYBO ώστε να υλοποιεί ένα Debayering Filter όπου τα δεδομένα αποστέλλονται από τον ARM προς το FPGA για επεξεργασία και από τον επιταχυντή πίσω στον επεξεργαστή για την επίδειξή τους μέσω κατάλληλου software που επίσης φτιάξαμε. Η επικοινωνία αυτή επιτεύχθηκε μέσω της AXI-Stream διεπαφής.

Design - Block Diagram

Σε επίπεδο λογικής το φίλτρο αποτελείται από τέσσερις συνεργαζόμενες μονάδες. Αρχικά η μονάδα **ser2par** (Serial to Parallel), μετατρέπει το συνεχόμενο input stream από pixels σε παράθυρα 3x3. Η διαδικασία αυτή επιτυγχάνεται με την αξιοποίηση τριών FIFO οι οποίες ελέγχονται εξ' ολοκλήρου από έναν counter. Οι FIFO υλοποιήθηκαν μέσω του έτοιμου IP που μας παρέχει η Xilinx στο Vivado, έχουν οριστεί ως generic με την ικανότητα να διαχειριστούν εικόνες διαφορετικών διαστάσεων, ενώ η έξοδός τους λαμβάνεται από 9 registers που αποτελούν το window εξόδου. Ωστόσο το output αυτό αναλόγως το στιγμιότυπο στην συνολική διαδικασία ενδέχεται να περιέχει και λανθασμένες τιμές. Η διαχείριση του φαινομένου αυτού ανατίθεται στο **edge_corrector** (Edge Corrector) module το οποίο εφαρμόζει κατάλληλη μάσκα με μηδενικά στο παράθυρο αναλόγως το σημείο επεξεργασίας της εικόνας, το οποίο κάνει track μέσω ενός ακόμη counter. Το τελικό αποτέλεσμα περνάει στο **compute_unit** (Compute Unit). Το module αυτό πραγματοποιεί τους υπολογισμούς της διαδικασίας του debayering υπολογίζοντας την RGB τιμή του κάθε pixel. Βασίσαμε την υλοποίησή του σε έναν counter σύμφωνα με την τιμή του οποίου διαχειριζόμαστε μια μηχανή καταστάσεων. Κάθε state αναφέρεται στο κεντρικό pixel του παραθύρου, για το οποίο γίνεται και ο αντίστοιχος υπολογισμός, πριν εξάγουμε το αποτέλεσμα στην έξοδο. Ακολουθούν τόσο το RTL block diagram όσο και η μηχανή καταστάσεων που περιγράψαμε.

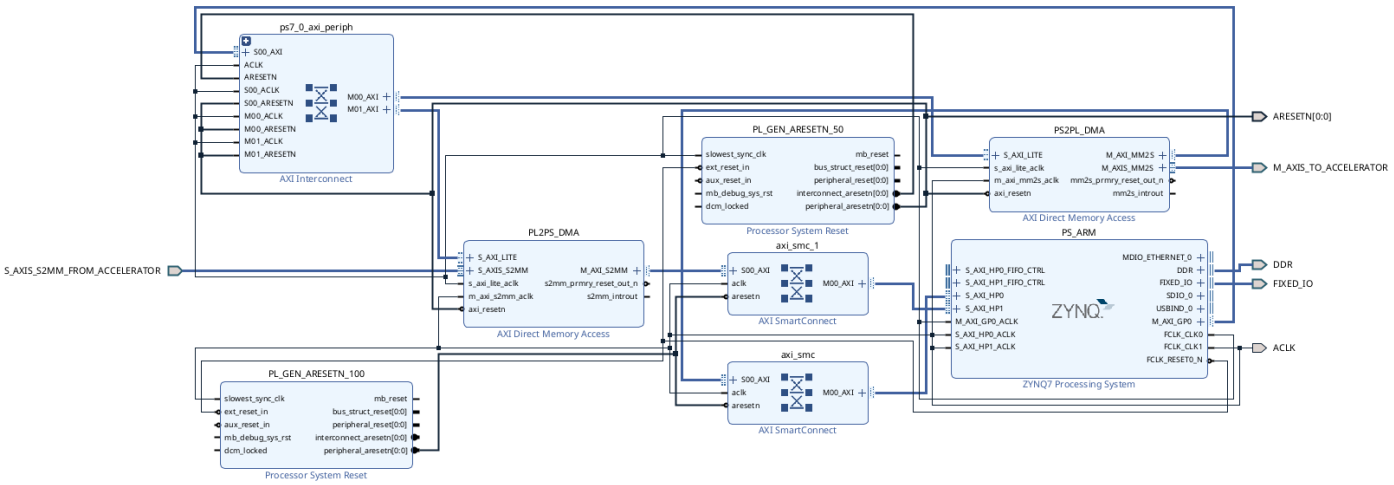




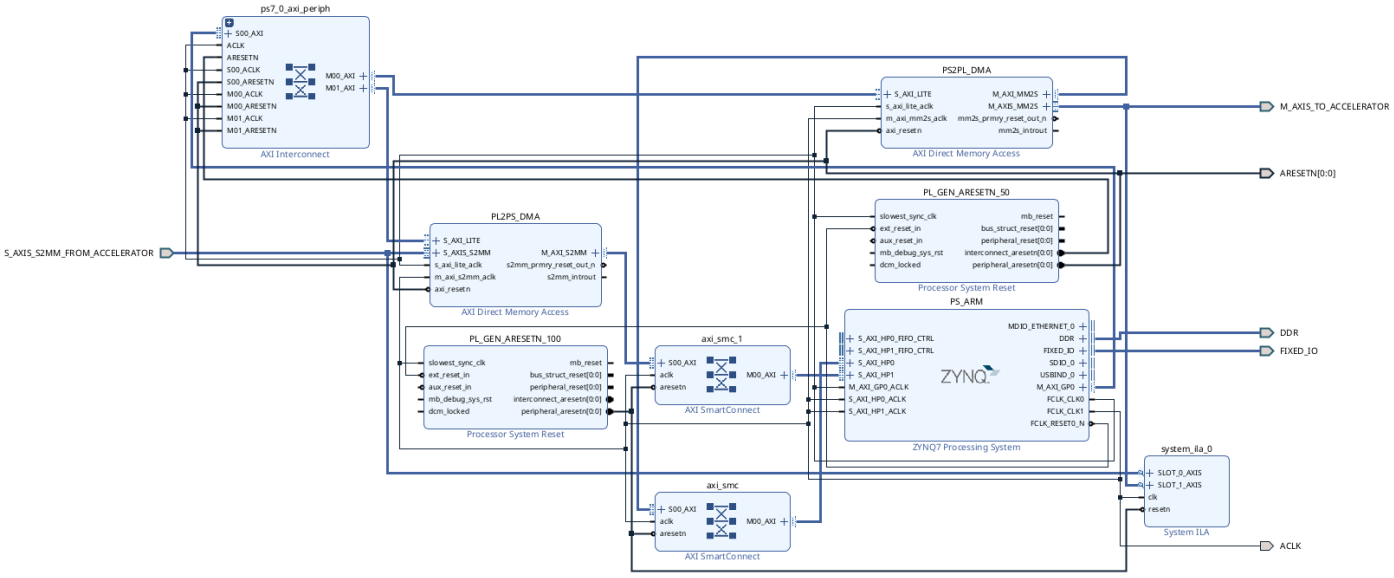
Synthesis – Utilization

Έχοντας έτοιμη την λογική του φίλτρου και έχοντας τρέξει κατάλληλα simulations συνολικά αλλά και για κάθε module ξεχωριστά μέσω testbenches, συνδέουμε το φίλτρο με την διεπαφή AXI-Stream κάνοντας κατάλληλη διασύνδεση των σημάτων. Η γενική αρχιτεκτονική και το signal mapping είναι αυτά που περιγράφηκαν στην εκφώνηση, ενώ για τα υπόλοιπα σήματα του πρωτοκόλλου, είτε τα κάναμε hardwire στην τιμή που ορίζει η λειτουργικότητα της εφαρμογής μας είτε φτιάξαμε κάποια επιπλέον λογική για την ενεργοποίησή τους. Τέλος η επικοινωνία μεταξύ PS και PL επιτυγχάνεται μέσω DMA το οποίο αρχικοποιείται από την εφαρμογή μας.

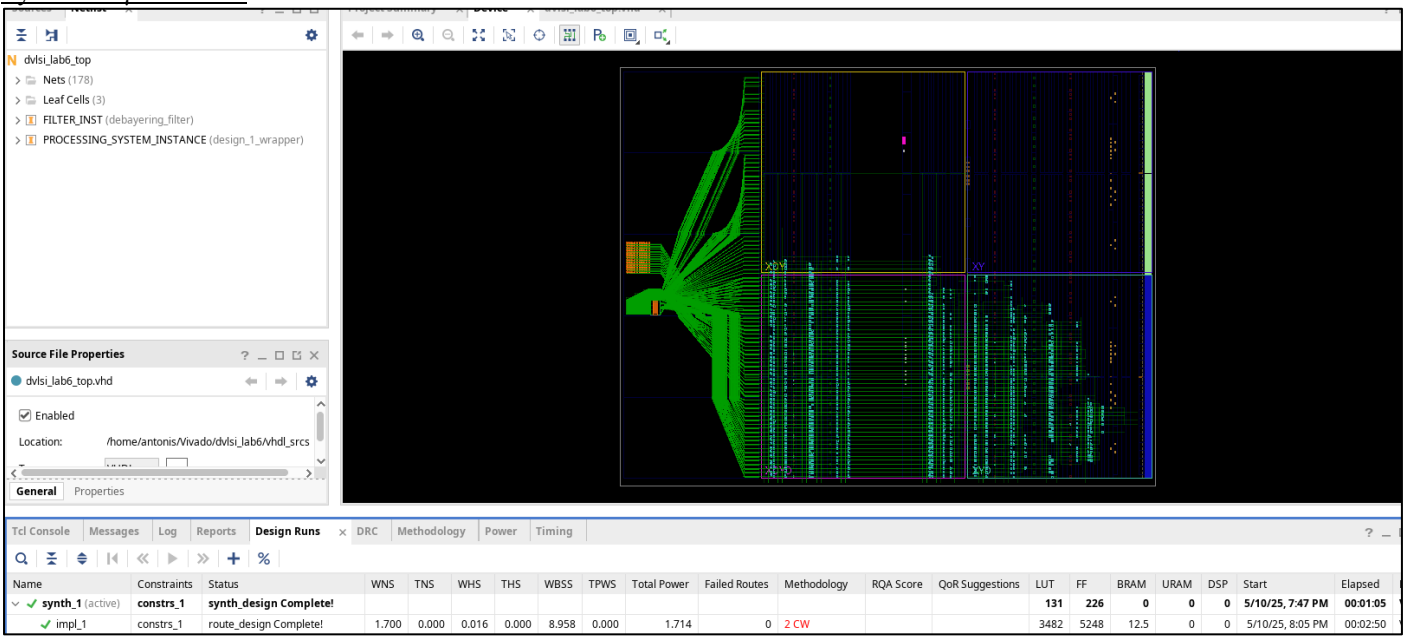
Συνολικό Block Diagram PS – PL



Συνολικό Block Diagram PS – PL – ILA

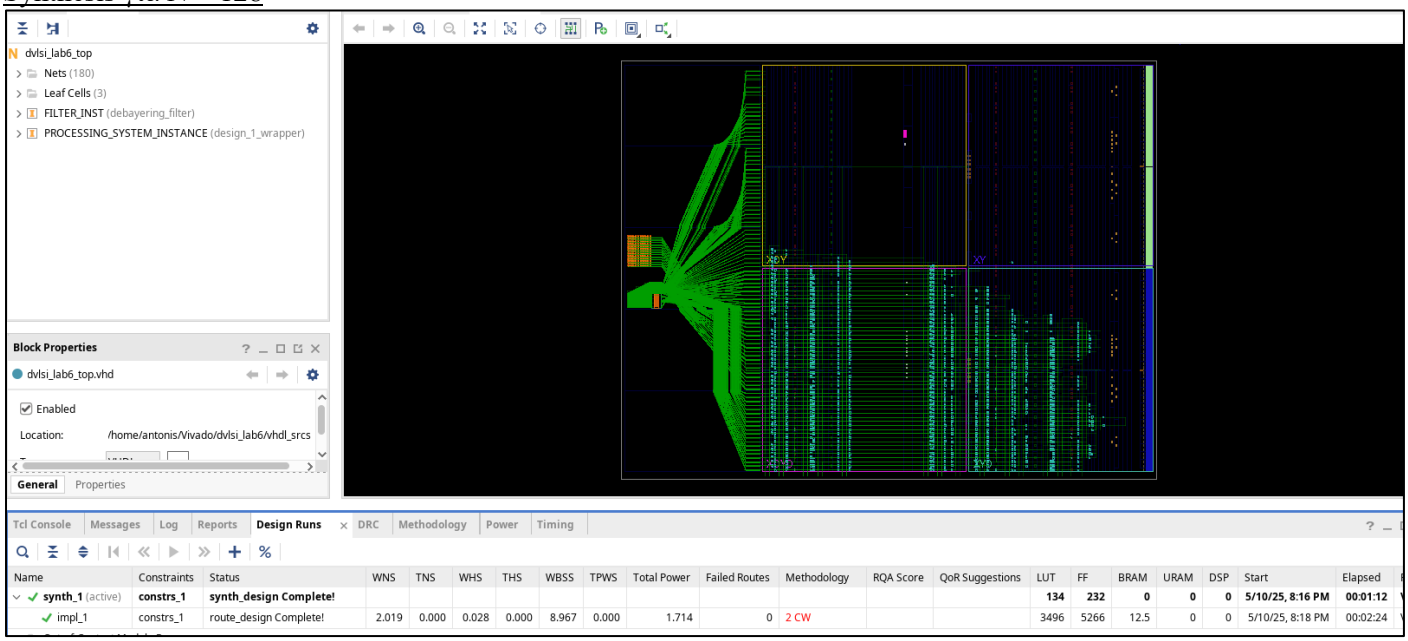


Synthesis για N = 64



Από τα αποτελέσματα της σύνθεσης βλέπουμε πως το τελικό design δεσμεύει 131 LUTs, 226 FFs και καθόλου BRAM, URAM και DSP. Αξίζει να σημειώσουμε πως ενώ το FIFO IP χρησιμοποιεί την Block RAM υλοποίηση (κάτι που φαίνεται και στις λεπτομέρειες του implementation με 12.5 BRAM) εν τέλει μετά από optimizations στην διαδικασία του synthesis υπάρχει μηδενικό BRAM utilization.

Synthesis για N = 128



Από τα αποτελέσματα της σύνθεσης βλέπουμε πως το τελικό design δεσμεύει 134 LUTs, 232 FFs και καθόλου BRAM, URAM και DSP. Η προηγούμενη παρατήρηση για την FIFO ισχύει και σε αυτή την περίπτωση.

Latency – Throughput – Results

Όσον αφορά την απόδοση του τελικού φίλτρου, με συνεχή έγκυρη είσοδο και έξοδο έχουμε latency N + 4 κύκλους ρολογιού και throughout ίσο με $\frac{N^2}{N^2+N+4}$ δηλαδή 0.984 για N = 64, 0.992 για N = 128 και 0.999 για N = 1024. Συνεπώς όσο αυξάνεται η διάσταση της εικόνας έχουμε μεγαλύτερο latency μέχρι να ξεκινήσει το φίλτρο να παράγει έγκυρες εξόδους, ωστόσο το throughput τείνει στο 1, δηλαδή μία έγκυρη έξοδο ανά κύκλο ρολογιού. Το φίλτρο δοκιμάστηκε μέσω εφαρμογής γραμμένης σε C της οποίας τον κώδικα έχουμε επισυνάψει. Μέσω αυτής ελέγξαμε την ορθότητα των αποτελεσμάτων του FPGA με μία software υλοποίηση του φίλτρου που έτρεξε στον ARM, καθώς και την (αισθητή όπως αποδείχθηκε!) διαφορά στην ταχύτητά τους.

```
HELLO 1
DMA transfer finished.
FPGA data saved.
SW calculation finished.
Total errors: 0.
FPGA time: 3416022 cycles.
SW time: 3377205939 cycles.
Speedup:  988.
```