2024

Αναφορά 6ης εργαστηριακής Άσκησης

Εργαστήριο VLSI

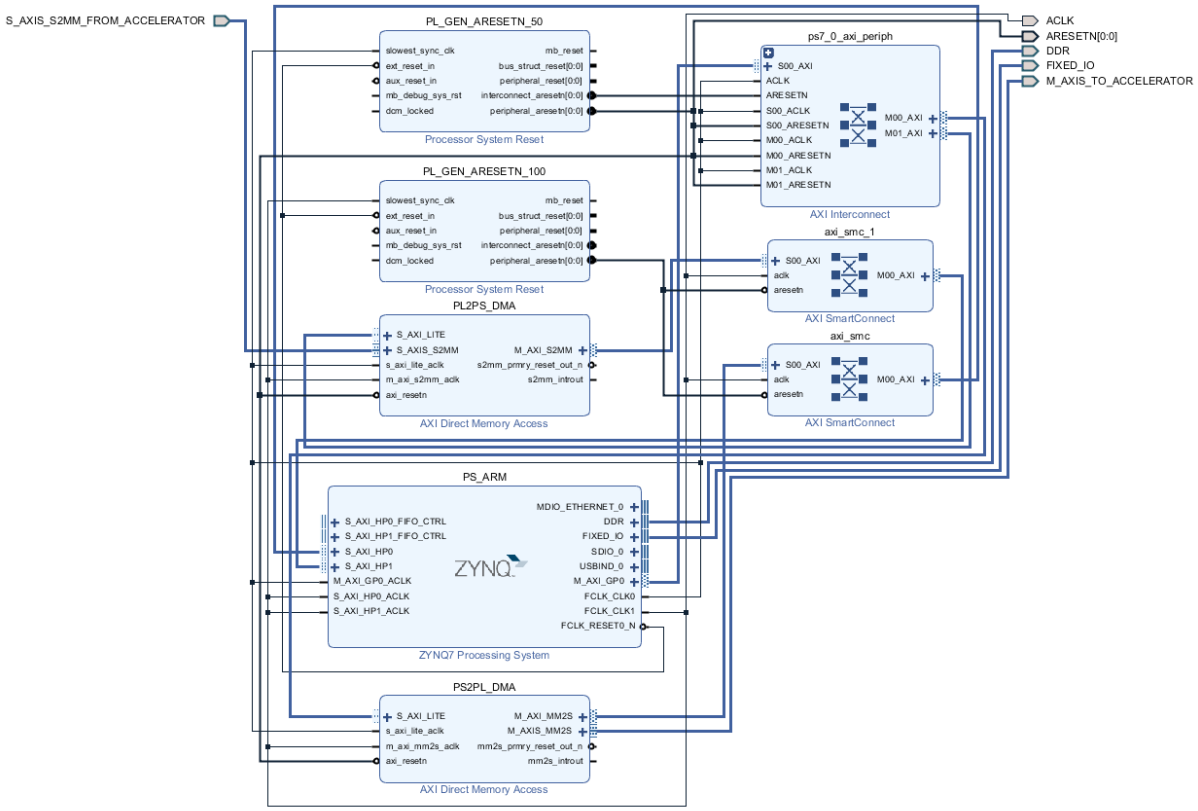
ΠΑΠΑΔΟΠΟΥΛΟΣ ΣΠΥΡΙΔΩΝ (ΑΜ):03120033
ΕΜΜΑΝΟΥΗΛ ΞΕΝΟΣ (ΑΜ):03120850

Εισαγωγή

Σε αυτή την άσκηση μας ζητήθηκε η κατασκευή ενός deybayer φίλτρου το οποίο θα λαμβάνει τα δεδομένα του από τον επεξεργαστή του Zybo θα κάνει την επεξεργασία στο FPGA και θα επιστρέφει την έξοδο στον επεξεργαστή.

Block Diagram

Αρχικά θα παραθέσουμε το Μπλοκ Διάγραμμα του κυκλώματος.



Από το παραπάνω Μπλοκ Διάγραμμα παρατηρούμε τις ακόλουθες βασικές δομικές μονάδες:

- PS_ARM: Αποτελεί τον επεξεργαστή ARM που έχει το Zybo
- PS2PL_DMA: Αποτελεί το DMA που μεταφέρει δεδομένα από τον επεξεργαστή ARM στο Debayer φίλτρο
- PL2PS_DMA: Αποτελεί το DMA που μεταφέρει δεδομένα από το FPGA στον ARM επεξεργαστή
- Ps7_0_axi_periph: Αποτελεί στο φίλτρο Debayer που έχουμε υλοποιήσει.

Ανάλυση Πόρων

Η ακόλουθη ανάλυση των πόρων του fpga γίνεται για ολόκληρη την υλοποίηση μαζί με το wrapper ,καθώς στο τέλος με βάση αυτό θα γίνει program το fpga. Οπότε αν πρέπει να αξιολογήσουμε την υλοποίηση του debayer μας με axi stream πρέπει να αξιολογήσουμε το σύνολο του συστήματος, που υλοποιήθηκε στην άσκηση αυτή.

Η ανάλυση πόρων του FPGA μετά το Implementation για N=64 είναι:

Resource	Utilization	Available	Utilization %
LUT	4156	17600	23.61
LUTRAM	443	6000	7.38
FF	6458	35200	18.35
BRAM	12.50	60	20.83
BUFG	3	32	9.38

Name	Slice LUTs (17600)	Slice Registers (35200)	F7 Muxes (8800)	Slice (4400)	LUT as Logic (17600)	LUT as Memory (6000)	LUT Flip Flop Pairs (17600)	Block RAM Tile (60)	Bonded IOPADs (130)	BUFGCTRL (32)
dvlsi2021_lab5_top	4156	6458	24	1858	3713	443	2531	12.5	130	3
Debayering_instance (...	371	240	10	136	371	0	108	1.5	0	0
dff (D_flip_flop_1_bit)	1	1	0	2	1	0	0	0	0	0
PROCESSING_SYSTE...	3784	6217	14	1734	3341	443	2422	11	0	2

Η ανάλυση πόρων του FPGA μετά το Implementation για N=128 είναι:

Resource	Utilization	Available	Utilization %
LUT	4158	17600	23.63
LUTRAM	442	6000	7.37
FF	6470	35200	18.38
BRAM	12.50	60	20.83
BUFG	3	32	9.38

Name	Slice LUTs (17600)	Slice Registers (35200)	F7 Muxes (8800)	Slice (4400)	LUT as Logic (17600)	LUT as Memory (6000)	LUT Flip Flop Pairs (17600)	Block RAM Tile (60)	Bonded IOPADs (130)	BUFGCTRL (32)
dvlsi2021_lab5_top	4158	6470	27	1869	3716	442	2531	12.5	130	3
Debayering_instance (...	381	252	13	134	381	0	109	1.5	0	0
dff (D_flip_flop_1_bit)	1	1	0	2	1	0	0	0	0	0
PROCESSING_SYSTE...	3776	6217	14	1747	3334	442	2422	11	0	2

Παρατηρούμε λοιπόν ότι και οι δύο υλοποιήσεις παρόλο που το μέγεθος των FIFO που χρησιμοποιούμε είναι διαφορετικό οι πόροι που χρησιμοποιούνται είναι ίσοι.

Latency

Η καθυστέρηση που έχουμε από την ώρα που θα βάλουμε την έγκυρη είσοδο μέχρι να λάβουμε την έγκυρη έξοδο είναι η καθυστέρηση που θέλουμε να έχουμε ώστε τα να κάνουμε pop τα στοιχεία της δεύτερης FIFO έτσι ώστε να είναι ευθυγραμμισμένα με τα στοιχεία της πρώτης FIFO και να γίνεται σωστά ο υπολογισμός των pixel. Αυτό θα γίνει N-1 κύκλους μετά από την είσοδο του πρώτου pixel και μετά από 6 κύκλους από αυτό το σημείο έχουμε την πρώτη έγκυρη έξοδο (η καθυστέρηση αυτή οφείλεται εκτός από το shifting στους registers και σε κάποιους επιπλέον registers που χρησιμοποιούνται ώστε να υπολογίσουμε το σωστό αποτέλεσμα). Επομένως έχουμε καθυστέρηση N+5 κύκλους.

Throughput

Συνολικά θα έχουμε N+6 κύκλους μέχρι να δούμε την πρώτη έξοδο (1 κύκλος για να εισάγουμε το πρώτο δεδομένο + την καθυστέρηση που χρειάζεται ώστε να δούμε την έξοδο) και (N*N-1) κύκλους για να δούμε το αποτέλεσμα των υπόλοιπων pixels. Άρα θα χρειαστούμε συνολικά:

$$(N+6)+(N*N-1)=N^2+N\ +5\ \textit{κύκλους}$$

Έτσι το throughput θα είναι:

$$throughput=\frac{N*N}{N^2+N+5}$$

FSM

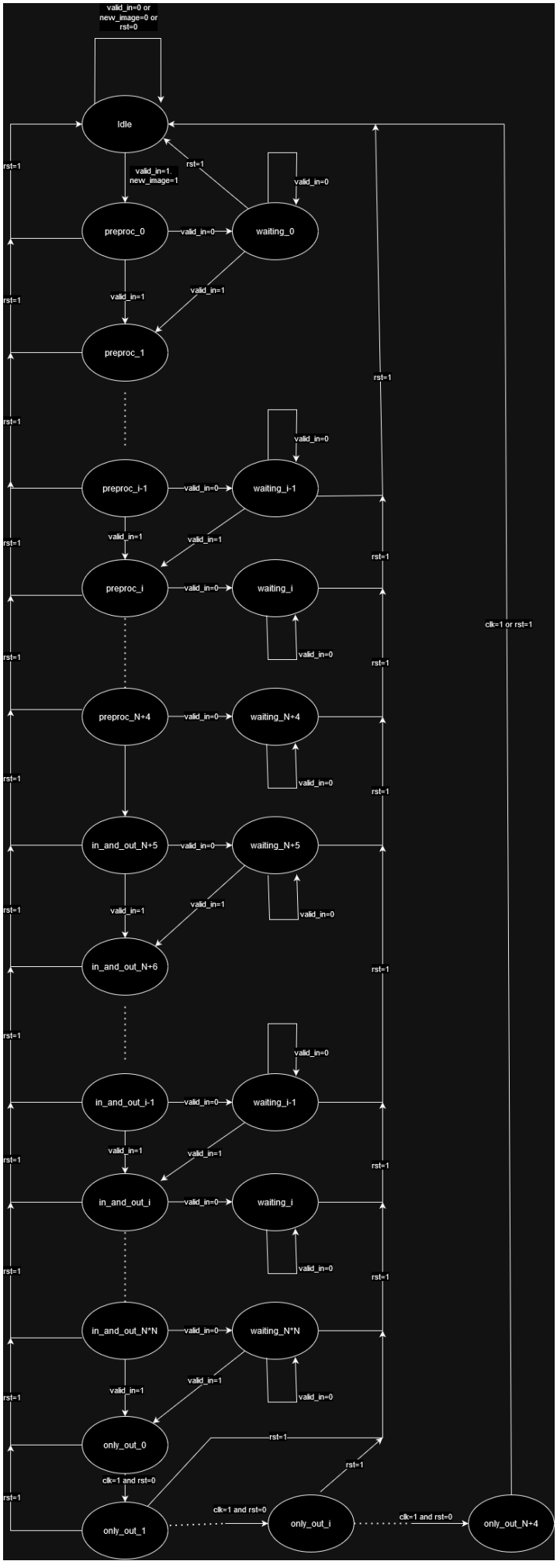
Σε αυτό το σημείο θα παραθέσουμε το FSM που περιγράφει επακριβώς την λειτουργία του Debayer μας. Ωστόσο, αρχικά να κάνουμε κάποιες διευκρινήσεις:

- Το FSM αυτό αντιστοιχίζεται επακριβώς στην λειτουργία του κώδικα. Συνεπώς, για να δείξουμε την επανάληψη κάποιων διαδικασιών εισόδου εξόδου με βάση την τιμή των μετρητών έχουμε φτιάξει πολλές καταστάσεις με το ίδιο όνομα και έναν δείκτη i που καθορίζει το πλήθος των στοιχείων εισόδου εξόδου που έχουμε εξετάσει.
- Για το FSM φτιάξαμε 5 καταστάσεις οι οποίες παρατίθενται ακολούθως με μία μικρή επεξήγηση η κάθε μία
  - Idle : Ο debayer περιμένει κάποια νέα είσοδο, όταν λάβει το new_image=1 και valid_in=1 θα ξεκινήσει την διαδικασία επεξεργασίας της νέας εισόδου που θα λάβει.
  - Preprocess_i : Ο debayer λαμβάνει κάποιο αριθμό εισόδων (συγκεκριμένα N+4) πριν ξεκινήσει να βγάζει μία έξοδο. Αυτή η «καθυστέρηση» είναι απαραίτητη ώστε να έχουμε λάβει στην είσοδο το απαιτούμενο πλήθος pixel εισόδου, ώστε για κάθε Pixel να μπορούμε να υπολογίσουμε σωστά την έξοδο(δηλαδή να έχουμε όλες τις συνιστώσες του)

- Waiting_i : Ο debayer δεν έχει λάβει ακόμα όλη την είσοδο και ενώ είναι σε μία κατάσταση i (είτε preprocess_i είτε in_and_out_i) λαμβάνει valid_in=0. Σε αυτή την περίπτωση ο debayer κάνει gate το clock των fifo και των dff και ουσιαστικά η λειτουργία του κυκλώματος(τιμές μετρητών, κατάσταση fifo και κατάσταση dff) «παγώνουν» και αναμένουν το επόμενο valid_in=1,ώστε να επανέλθει η κανονική ροή
- In_and_out_i: Στο σημείο αυτό εκτελέσαμε την προεπεξεργασία και μπορούμε να συνεχίσουμε στην παραγωγή της εξόδου(=> valid_out=1). Ωστόσο δεν έχουμε λάβει ακόμα όλη την είσοδο ακόμα οπότε συνεχίζουμε να λαμβάνουμε και είσοδο
- Only_out_i : Ο debayer έλαβε όλη την είσοδο και συνεπώς σε κάθε κύκλο ρολογιού παράγει και από μία έξοδο(εκτός αν προφανώς έρθει reset). Προφανώς το πλήθος των κύκλων σε αυτή την φάση είναι ίσος με το πλήθος των κύκλων που κάναμε preprocess.

- Από οποιαδήποτε κατάσταση αν έρθει rst=1 τότε πάμε στην κατάσταση idle.
- Για την μετάβαση από μία κατάσταση στην επόμενη στην κανονική λειτουργία του debayer (χωρίς να έρθει reset) κανονικά στο FSM θα έπρεπε να φαίνεται ότι η μετάβαση γίνεται αν rst=0 και αν το clk είναι στο rising_edge. Ωστόσο για οικονομία χώρου και ευκολία στην ανάγνωση αυτό παραλείφθηκε, αλλά είναι σημαντικό να το λάβουμε υπό όψιν.

Το διάγραμμα θα υπάρχει και μέσα στο παραδοτέο zip, σε περίπτωση που κάποιος θέλει να το δει σε καλύτερη ποιότητα.


valid_in=0 or new_image=0 or rst=0
Idle
rst=1
valid_in=1. new_image=1
valid_in=0
preproc_0
valid_in=0
waiting_0
valid_in=1
valid_in=1
preproc_1
rst=1
valid_in=0
preproc_i-1
valid_in=0
waiting_i-1
valid_in=1
valid_in=1
preproc_i
valid_in=0
waiting_i
valid_in=0
clk=1 or rst=1
preproc_N+4
valid_in=0
waiting_N+4
valid_in=0
in_and_out_N+5
valid_in=0
waiting_N+5
valid_in=1
valid_in=1
valid_in=0
in_and_out_N+6
valid_in=0
in_and_out_i-1
valid_in=0
waiting_i-1
valid_in=1
valid_in=1
in_and_out_i
valid_in=0
waiting_i
valid_in=0
in_and_out_N*N
valid_in=0
waiting_N*N
valid_in=1
valid_in=1
valid_in=0
only_out_0
rst=1
rst=1
clk=1 and rst=0
only_out_1
clk=1 and rst=0
only_out_i
clk=1 and rst=0
only_out_N+4