Ομάδα Α10:

Γεώργιος Παγώνης : 03117030

Δημήτριος - Σταμάτιος Μπούρας : 03117072

8ο εξάμηνο - ΨΗΦΙΑΚΑ ΣΥΣΤΗΜΑΤΑ VLSI

ΣΗΜΜΥ

5ή σειρά ασκήσεων:

Όλα τα κυκλώματα περιέχονται σε ξεχωριστά αρχεία vhd στο zip που αποστείλαμε.

- για το fifo_generator_2048 προσθέτοντας τους embedded register για read latency = 2 , το valid flag , και το data count, σαν μέγεθος δεδομένων τα 8 bits, σαν μέγεθος Fifo 2048.

-για το fifo_generator_2048_32_no_lat επιλέγουμε την επιλογή First word fall through για να έχουμε 0 read latency , μέγεθος δεδομένων τα 32 bits , μέγεθος Fifo 2048 , προσθέτοντας το data count και το valid flag.

1)

Fpga

Οι διαφορές με την προηγουμένη άσκηση είναι :

- έχουμε υλοποιήσει μέσα στο module image την λειτουργία για το new image , που πλέον δεν χρειάζεται να στέλνουμε new - image αλλά μόνο valid in και ξεκινάει την λειτουργία του fpga.
- έχουμε υλοποιήσει και το stall για να μπορούμε αν δεν έρχεται το κατάλληλο ή valid_in ή το T_ready_in (που είναι το σήμα που μας επιτρέπει να διαβάσουμε το αποτέλεσμα από το Fpga στον ARM επεξεργαστή) να κάνει stall την λειτουργία του fpga - καθώς και την pipeline λειτουργία του. Αυτό λαμβάνει ισχύ σε 3 κύκλους από την στιγμή που καλούμε κάποιο από τα παραπάνω σήματα είναι 0 . Για να μην χάνονται τα αποτελέσματα που παράγει το σύστημα μας , έχουμε ένα εξωτερικό module που περιλαμβάνει το image μαζί με μια fifo - χωρίς read latency , η οποία αποθηκεύει - γραφεί δεδομένα για τιμές που το valid out από το image είναι 1 και διαβάζει δεδομένα για τιμές που το T_ready_in από τον επεξεργαστή

είναι 1 . Την έχουμε σε μέγεθος 2048 , αλλά το μέγεθος θα μπορούσε να είναι πολύ μικρότερο (έως και 5 στοιχεια) γιατί το image αν το T_ready_in είναι 0 τότε μετά από 3 κύκλους κάνει stall και δεν δίνει πλέον valid out = 1,αλλα 0 . Το t_ready_out είναι 1 λέμε στον επεξεργαστή να στείλει το επόμενο data για το διάβασμα ,έχουμε κάνει το t_ready_out να λαμβάνει τιμές από το T_ready_in με καθυστέρηση έναν κύκλο συνέχεια για να λαμβάνουμε συνέχεια δεδομένα μέχρι το T_ready_last. Άρα στην πράξη δεν θα χρειαστέι ποτέ να κρατήσουμε πολλά στοιχεία καθώς όταν γίνεαι το stall μετα από λίγους κύκλους τα αποτελέσματα που παράγονται μας είναι άχρηστα.

Arm

Για να τρέξει το πρόγραμμα μας στο Arm , αρχικά δίνουμε τιμές στους
TxBufferPtr = (u8 *)TX_BUFFER ;
RxBufferPtr = (u32 *)RX_BUFFER;

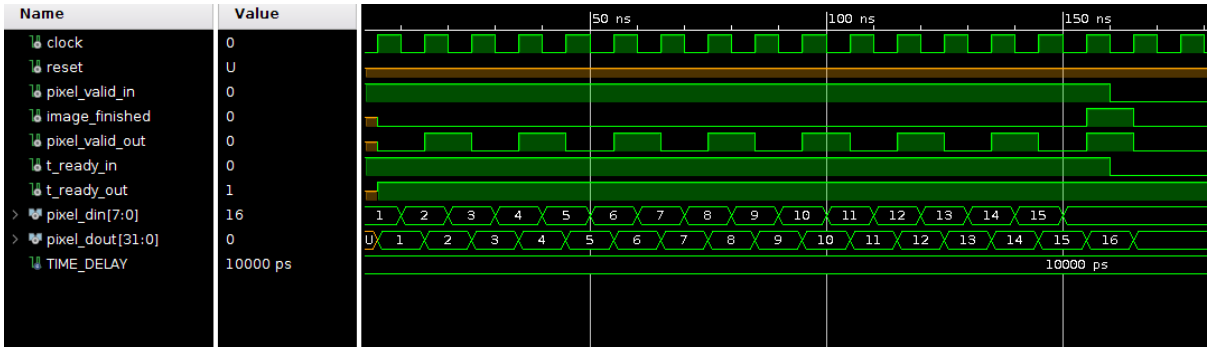
για να αποθηκεύει στην σωστή θέση μνήμης .

Μετά γεμίζουμε έναν πίνακα A ο οποίος θα περιέχει τις τιμές για το software processing .
Κάνουμε initialize τα Tx-dma , Rx-dma .
Γεμίζουμε τον Tx-buffer και πραγματοποιούμε το Simple Transfer πρώτα στον Rx-buffer και μετά στον Tx- buffer.

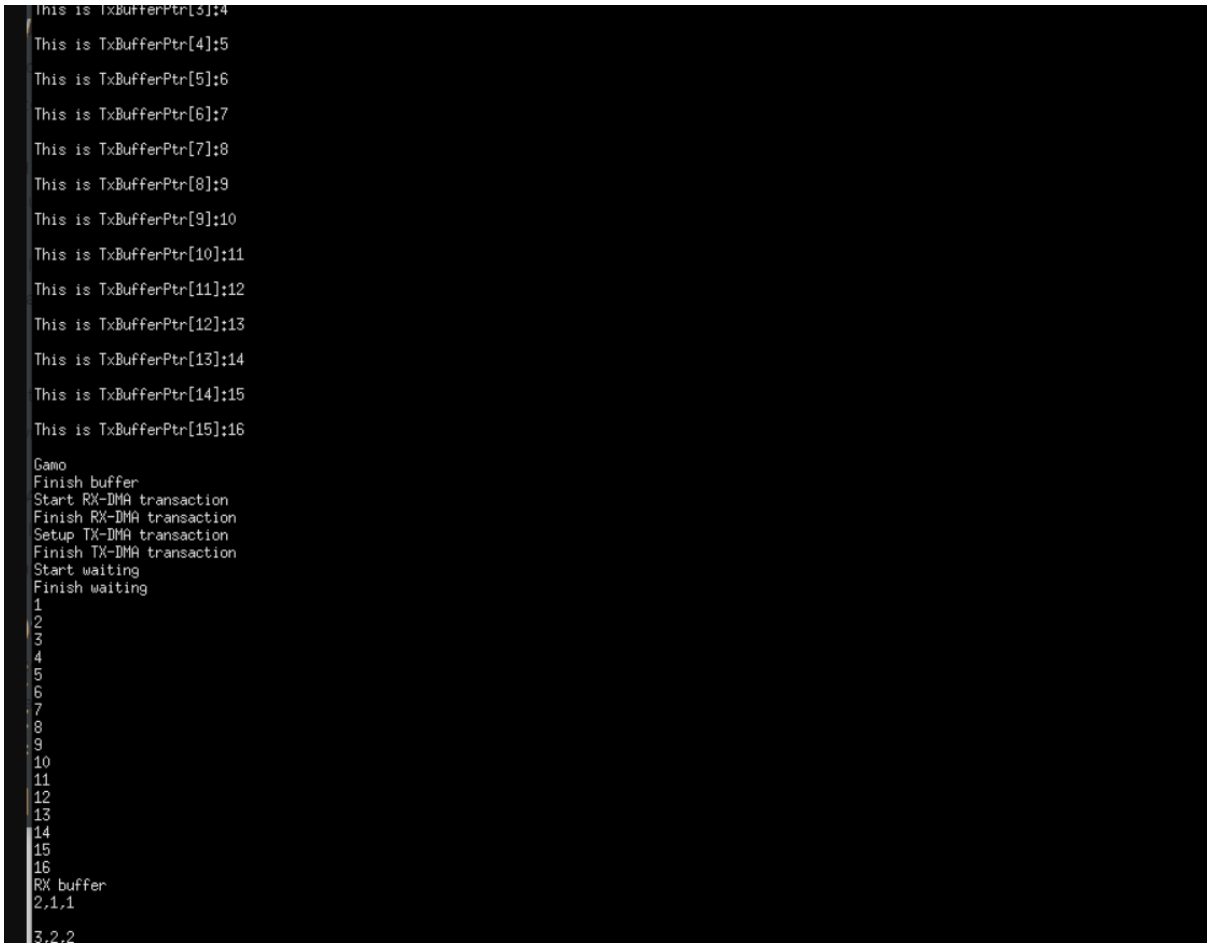
Σε αυτό το κομμάτι δεν μας δούλευε το master-slave όπως είχαμε ονομάσει το fpga module μας. Για αυτό υλοποιήσαμε ένα ακόμα module το test το οποίο απλά δίνει δεδομένα στην έξοδο του fpga , και κατά συνέπεια του επεξεργαστή . Αυτό το πρόγραμμα μας δούλευε και έβγαζε τα επιθυμητά αποτελέσματα στην έξοδο και το βλέπαμε και εμείς . Ωστόσο , ενώ το fpga φίλτρο μας έχει ακριβώς την ίδια λειτουργία με το test (όσο αφορά τις εισόδους και εξόδους του fpga), δεν καταφέραμε να τρέξουμε το master-slave σωστά. Αυτό ήταν απορία απομακρυσμένης πρόσβασης, ορισμένων τεχνικών προβλημάτων που αντιμετωπίζαμε κατά καιρούς και έλλειψη χρόνου.
Από την λειτουργεία τοτ test συμπεραίνουμε πως το .c αρχείο μας είναι σωστό και πως μάλλον το πρόβλημα βρίσκεται στο γεγονός πως η προσωμοίωση δεν είναι 100% ακριβής. Δηλαδή μερικά σήματα που φαίνονται να αλλάζουν σε ακμές ρολογιού ίσως αλλάζουν 1 με 2 ns πιο αργά γεγονός που προκαλεί πρόβλημα στην μεταφορά δεδομένων από το module μας πίσω στον επεξεργαστή . Έχουμε όμως την εντύπωση πως τα δεδομένα μεταφέρονται σωστά από τον buffer στο module.

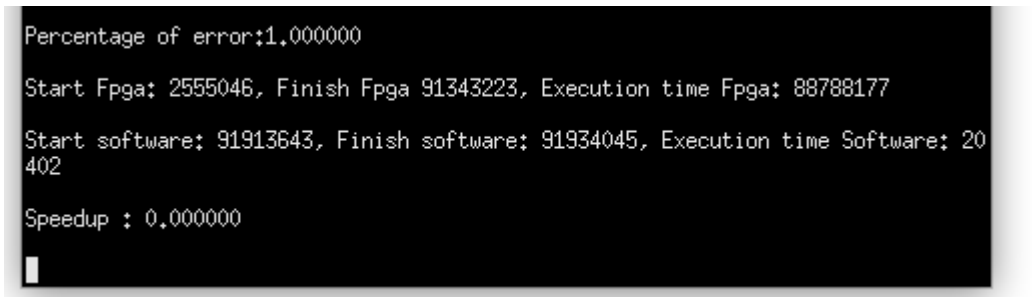
Για να ολοκληρώσουμε το c κώδικα εκτελούμε το software processing , τις τιμές τις οποίες αποθηκεύουμε σε πίνακες R,G,B . Στην αρχή και στο τέλος κάθε διεργασίας έχουμε τα αποτελέσματα από τους χρόνους που λαμβάνει χώρα ή κάθε διεργασία SW-HW.
Έχουμε την συνάρτηση compare που μετράει τα errors και στην συνέχεια τυπώνουμε τους χρόνους και το κλάσμα HW-runtime/Sw-runtime . Ακολουθούν ορισμένες φωτογραφίες από το simulation του καθενός καθώς και από το test όταν έτρεξε στους υπολογιστές του εργαστηρίου:

Test_tb:

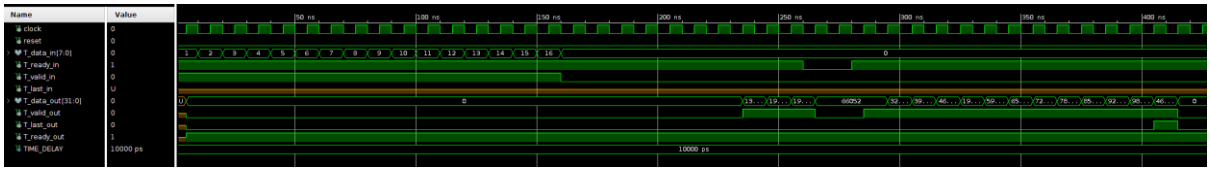


Έχουμε αλλάξει το pixel_valid_out επειδή ψάχναμε το λάθος μας στο master-slave και δούλευε και με αυτό αλλά και με pixel_valid_out 1 σε όλες τις σωστές τιμές .





Master-slave_tb:

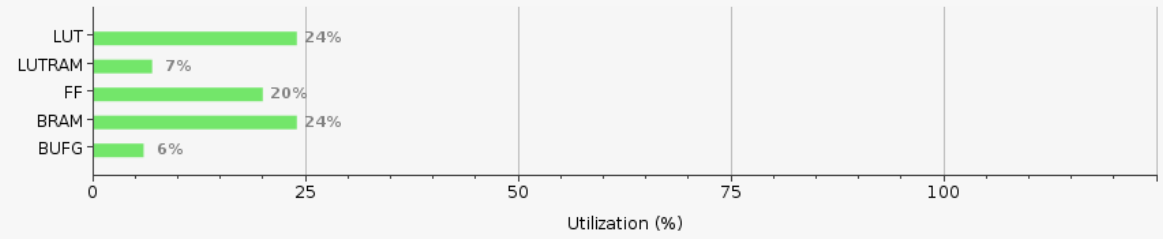


Με μια αυθαίρετη στιγμή το T_ready_in να είναι 0 .

Για το dvlsi :

Utilization Post-Synthesis | **Post-Implementation**

Graph | Table



Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	4266	17600	24.24
LUTRAM	445	6000	7.42
FF	6905	35200	19.62
BRAM	14.50	60	24.17
BUFG	2	32	6.25

Power **Summary** | On-Chip

Total On-Chip Power:	**1.724 W**
Junction Temperature:	**44.9 °C**
Thermal Margin:	40.1 °C (3.4 W)
Effective ϑJA:	11.5 °C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Medium

Implemented Power Report

Για το master-slave :

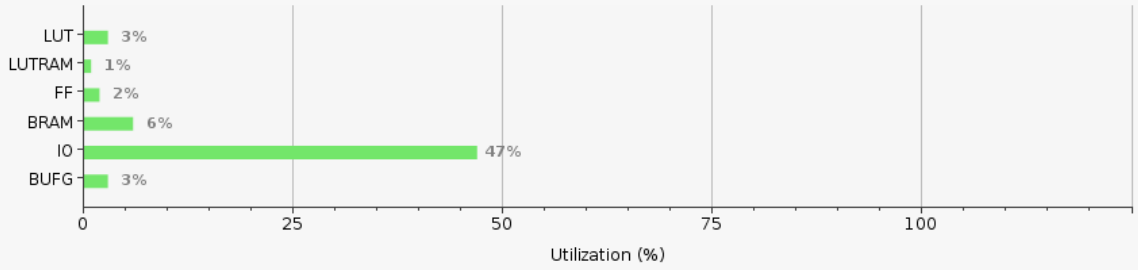
Unconstrained Paths - NONE - NONE - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Sou
Path 1	∞	5	5	22	module/new_image_reg[15]/C	module/cycle_counter[0]/R	4.995	1.123	3.872	∞	
Path 2	∞	5	5	22	module/new_image_reg[15]/C	module/cycle_counter[10]/R	4.995	1.123	3.872	∞	
Path 3	∞	5	5	22	module/new_image_reg[15]/C	module/cycle_counter[11]/R	4.995	1.123	3.872	∞	
Path 4	∞	5	5	22	module/new_image_reg[15]/C	module/cycle_counter[12]/R	4.995	1.123	3.872	∞	
Path 5	∞	5	5	22	module/new_image_reg[15]/C	module/cycle_counter[13]/R	4.995	1.123	3.872	∞	
Path 6	∞	5	5	22	module/new_image_reg[15]/C	module/cycle_counter[14]/R	4.995	1.123	3.872	∞	
Path 7	∞	5	5	22	module/new_image_reg[15]/C	module/cycle_counter[15]/R	4.995	1.123	3.872	∞	
Path 8	∞	5	5	22	module/new_image_reg[15]/C	module/cycle_counter[16]/R	4.995	1.123	3.872	∞	
Path 9	∞	5	5	22	module/new_image_reg[15]/C	module/cycle_counter[17]/R	4.995	1.123	3.872	∞	
Path 10	∞	5	5	22	module/new_image_reg[15]/C	module/cycle_counter[18]/R	4.995	1.123	3.872	∞	

Utilization

Post-Synthesis | **Post-Implementation**

Graph | Table



Graph | **Table**

Resource	Utilization	Available	Utilization %
LUT	464	17600	2.64
LUTRAM	4	6000	0.07
FF	663	35200	1.88
BRAM	3.50	60	5.83
IO	47	100	47.00
BUFG	1	32	3.13

Power

Summary | On-Chip

Total On-Chip Power:	**9.12 W (Junction temp exceeded!)**
Junction Temperature:	**125.0 °C**
Thermal Margin:	-45.2 °C (-3.4 W)
Effective θJA:	11.5 °C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Implemented Power Report