

ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Σχεδίαση Συστημάτων Υλικού - Λογισμικού

---

Εργαστήριο 3

Βελτιστοποιμένες μέθοδοι μεταφοράς δεδομένων από  
τον x86-host στον accelerator με τη χρήση του εργαλείου  
Vitis

---

Α. ΑΘΑΝΑΣΙΑΔΗΣ - Δ. ΚΑΡΑΝΑΣΣΟΣ

Διδάσκων: Ιωάννης Παπαευσταθίου

Version 0.4

Δεκέμβριος 2024

## Περιγραφή

Στο εργαστήριο 2 ασχοληθήκαμε με την υλοποίηση του επιταχυντή μας στο περιβάλλον του εργαλείου Vitis. Στο παρόν εργαστήριο θα ασχοληθούμε με διάφορες μεθόδους βελτιστοποίησης της μεταφοράς των δεδομένων από τον x86-host στον accelerator. Αυτό θα γίνει αντιληπτό μέσω 2 παραδειγμάτων.

- 1) Το πρώτο παράδειγμα παρουσιάζει την επιτάχυνση της πρόσθεσης πινάκων προσθέτοντας τα στοιχεία όχι ένα προς ένα, αλλά ως **διανύσματα στοιχείων του ενός πίνακα με διανύσματα στοιχείων του άλλου**. Αυτή η βελτιστοποίηση είναι δυνατή με την εκμετάλλευση της χωρητικότητας του διαύλου επικοινωνίας μεταξύ του x86-host και του accelerator. Συγκεκριμένα δε μεταφέρονται μέσω του διαύλου ένας προς ένας οι αριθμοί των 32-bit, αλλά σε διανύσματα των δεκαέξι.
- 2) Το δεύτερο παράδειγμα αφορά την **εκμετάλλευση των banks της DDR μνήμης της πλακέτας Alveo** και τη ρύθμιση των εισόδων/εξόδων του accelerator έτσι ώστε να διαβάζουν/γράφουν σε διαφορετικά banks. Με αυτόν τον τρόπο, η διαδικασία της μεταφοράς δεδομένων από και προς τον accelerator να γίνεται ταυτόχρονα.

Διαβάστε προσεκτικά το «Improving\_Performance\_lab» που βρίσκεται στη κατηγορία «Εργασίες και Τεστ» του elearning και εκτελέστε όλα τα βήματα του παραδείγματος που αναφέρονται.

Αφού εκτελέσετε όλα τα βήματα με επιτυχία του παραδείγματος «wide\_vadd» που περιγράφεται στο «Improving\_Performance\_lab», εισάγετε το κώδικα σας που αναπτύξατε στο 1<sup>ο</sup> εργαστήριο και υλοποιήστε τα 2 παραπάνω **optimizations**.

**Hint 1:** Προτείνουμε να χρησιμοποιήσετε ως αναφορά τους κώδικες του παραδείγματος (wide\_vadd) κάνοντας τροποποιήσεις πάνω σε αυτούς και αντικαθιστώντας μόνο το κώδικα του kernel με το δικό σας διατηρώντας τα **interfaces** του παραδείγματος.

**Hint 2:** Παρατηρήστε στο παράδειγμα πως μπορείτε να κάνετε προσπέλαση 32bit αριθμών από 512bit vector (με τη χρήση της range). Να τονίσουμε ότι πρέπει να χρησιμοποιήσετε 1d πίνακες και όχι 2d (δλδ. η προσπέλαση των πινάκων να γίνεται  $i * \text{DIM} + j$ ), και ότι πρέπει να χρησιμοποιήσετε interface 512bits, ενώ τα δεδομένα μπορούν να γίνονται access ανά 32bit με τη .range.

Hint 3: Για να μπορείτε να διαβάσετε τα δεδομένα σε vectors 512bit, ίσως να σας διευκολύνει να κάνετε transpose το 2ο πίνακα στη πλευρά του host όπως φαίνεται στην ακόλουθη Εικόνα.

1	2	3
4	5	6
7	8	9

Input

1	4	7
2	5	8
3	6	9

Output

Hint 4: Δε χρειάζεται να χρησιμοποιήσετε τη stream μέθοδο του παραδείγματος και το dataflow directive καθώς στο κώδικα που αναπτύξατε στα προηγούμενα εργαστήρια όπου γίνεται processing πάνω στα ίδια στοιχεία πολλές φορές, δε θα δείτε σημαντική αύξηση στο performance (αυτό δε σημαίνει ότι δε πρέπει να έχετε κατανοήσει πλήρως τη λειτουργία του και το παράδειγμα που σας έχουμε στείλει, στο οποίο θα εξεταστείτε).

**Παρατήρηση:** Παρακαλώ πολύ πριν ξεκινήσετε, διαγράψτε τα project που έχετε δημιουργήσει στο 2<sup>ο</sup> εργαστήριο και το φάκελο τους στο /mnt/data2/lab\_fpga/StudentX (αφού αποθηκεύσετε πρώτα τα src αρχεία σας). Επίσης κάθε ομάδα να δημιουργεί αυστηρά ENA MONO project στο server καθώς υπάρχει πρόβλημα χώρου.

## Προφορική εξέταση και Quiz

Η επιτυχής ολοκλήρωση του εργαστηρίου θα περιλαμβάνει την επιτυχή εκτέλεση των βημάτων του οδηγού Improving\_Performance\_lab.pdf, την επιτυχή εισαγωγή του kernel σας, την κατανόηση των βελτιστοποιήσεων καθώς και τη σύγκριση των αποτελεσμάτων μεταξύ των 2 παραπάνω βημάτων. Κατά τη διάρκεια της εξέτασης πρέπει να μας αναφέρετε τη βελτίωση του performance που πετύχατε συγκριτικά με το 2<sup>ο</sup> εργαστήριο.

Τέλος, κατά της διάρκεια του τελευταίου εργαστηρίου θα κληθείτε να απαντήσετε και ένα ολιγόλεπτο Quiz που θα περιλαμβάνει ερωτήματα εφ' όλης της ύλης των εργαστηρίων.