Σχεδίαση Συστημάτων Υλικού-Λογισμικού

Εργαστήριο 2

Στο δεύτερο εργαστήριο του μαθήματος, μας ζητήθηκε να δοκιμάσουμε τους κώδικες που δημιουργήσαμε στο πρώτο εργαστήριο, δηλαδή τον πολαπλασιασμό πινάκων, στο Vitis. Σκοπός ήταν να κατανοήσουμε καλύτερα κάποια ζητήματα που αφορούν το HW/SW, καθώς και να κάνουμε κάποια simulations μέσα σε αυτό το platform.

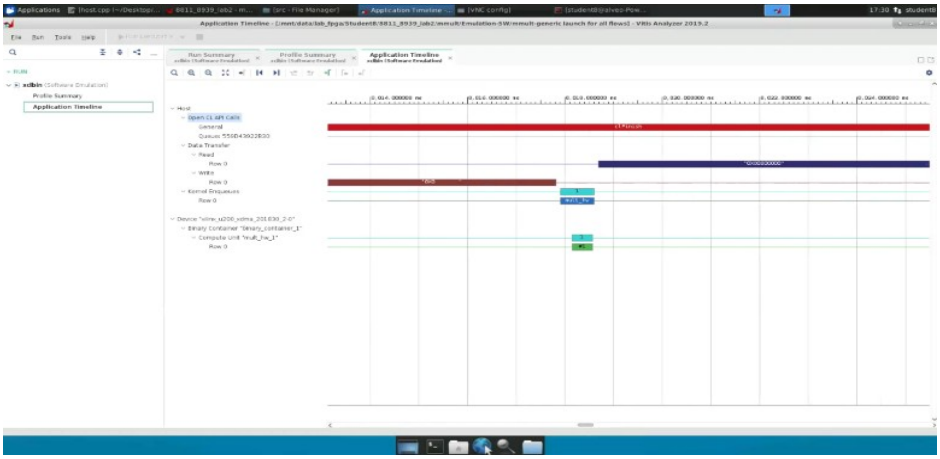
Κώδικες

Στο προηγούμενο εργαστήριο, ο κώδικας χωριζόταν στον testbench και στον hw_emulation κώδικα, που σκοπό είχαν αφενός να μπορέσουμε να συγκρίνουμε την κάθε μια υλοποίηση ξεχωριστά, αλλά και να μπορούμε να κάνουμε την σύνθεση μεταξύ HW/SW, μέσω του Vivado HLS. Όσον αφορά το vitis, βλέπωντας κάποια παραδείγματα στο internet, οι υλοποιήσεις που βρήκαμε, καθώς και το παράδειγμα vadd ήταν κάπως διαφορετικές. Πρώτον, host κώδικας (ο αντίστοιχος testbench κώδικας), είχε μεγάλο κομμάτι γραμμένο σε OpenCL με το οποίο δεν ήμασταν ιδιαίτερα εξοικειωμένοι. Παρ'όλα αυτά, καταλάβαμε πως με την OpenCL, υπάρχουν μεγάλα περιθώρια εξατομίκευσης σε σχέση με το πως ο κώδικας θα περνάει στο HW μας, αλλά και πολύ καλύτερος έλεγχος των σφαλμάτων που μπορεί να υπάρξουν κατά τη διαδικασία αυτήν και κυρίως στην διαδικασία που γράφουμε στη μνήμη του HW.

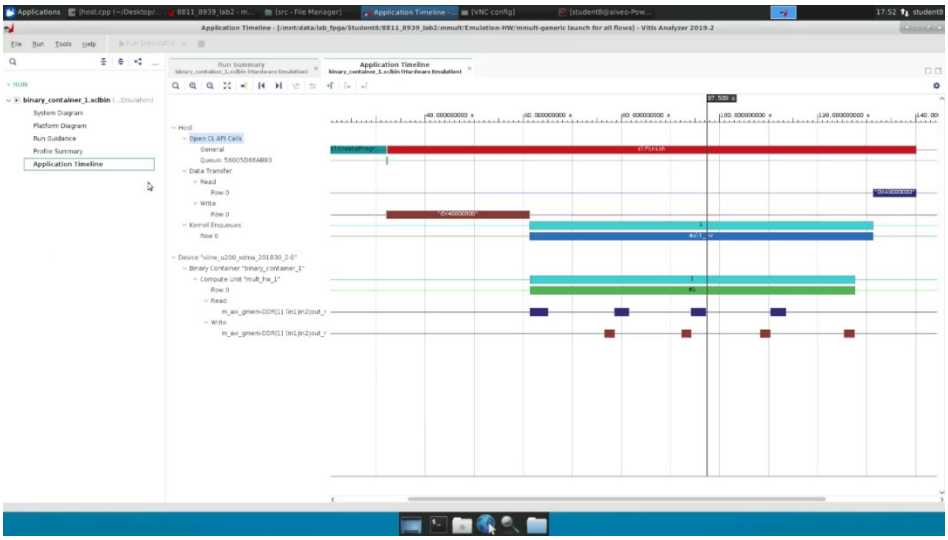
Όσον αφορά τον κώδικα του kernel (ο αντίστοιχος hw_emulation κώδικας), πέρα από τα pragmas του προηγούμενου εργαστηρίου (PIPELINE, LOOP_UNROLL, LOOP_TRIPCOUNT ARRAY_PARTITION), ήρθαμε σε επαφή με τα INTERFACE m_axi και INTERFACE s_axilite. Το INTERFACE pragma γενικά χρησιμοποιήται για να δηλώσουμε πως θα δημιουργηθούν τα ports της RTL κατά τη διαδικασία του interface synthesis. Πιο συγκεκριμένα το m_axi υλοποιεί τα ports σαν AXI4, ενώ το s_axilite τα υλοποιεί σαν AXI4-Lite. Αυτά πρακτικά αποτελούν κάποια communication interface, τα οποία χρησιμοποιούνται από την Xilinx σαν communication buses στα προϊόντα της. Στην πραγματικότητα το AXI4-Lite είναι ένα υποσύστημα του AXI4, με κυρίαρχη διαφορά – μεταξύ άλλων – ότι κάθε data access χρησιμοποιεί ολόκληρο το data bus width (32 ή 64 bits). Με μια λογική λοιπόν ότι κάθε πίνακας χωρίζεται σε buffer_sizes 4 φορές, θα γίνεται τελικά η πράξη που θέλουμε σε 4 "στάδια".

Όσον αφορά τον κώδικα του host, είχαμε μια μεγαλύτερη δυσκολία, διότι στο 1ο εργαστήριο είχαμε προχωρήσει σε στατική δημιουργία πινάκων, ενώ στα παραδείγματα που βρίσκαμε στο ίντερνετ όλα ορίζονταν δυναμικά. Έτσι προχωρήσαμε σε κάποιες τροποποιήσεις του κώδικα ώστε, τελικά, να λειτουργήσει. Λόγω της στατικής δήλωσης των πινάκων, το πρόγραμμα αυτό καθαυτό δεν ήταν λειτουργικό, αφού είχε εισαχθεί περιορισμός στη διαθέσιμη μνήμη που μπορεί να δεσμευτεί με αυτό τον τρόπο. Για την επίλυση αυτού του ζητήματος, μειώσαμε το μέγεθος της κάθε διάστασης των πινάκων από 8 σε 6 κι έτσι μπόρεσαν οι προσομοιώσεις να υλοποιηθούν κανονικά. Παρακάτω σας παραθέτουμε μερικά Screenshot από τα αποτελέσματα των προσομοιώσεων.

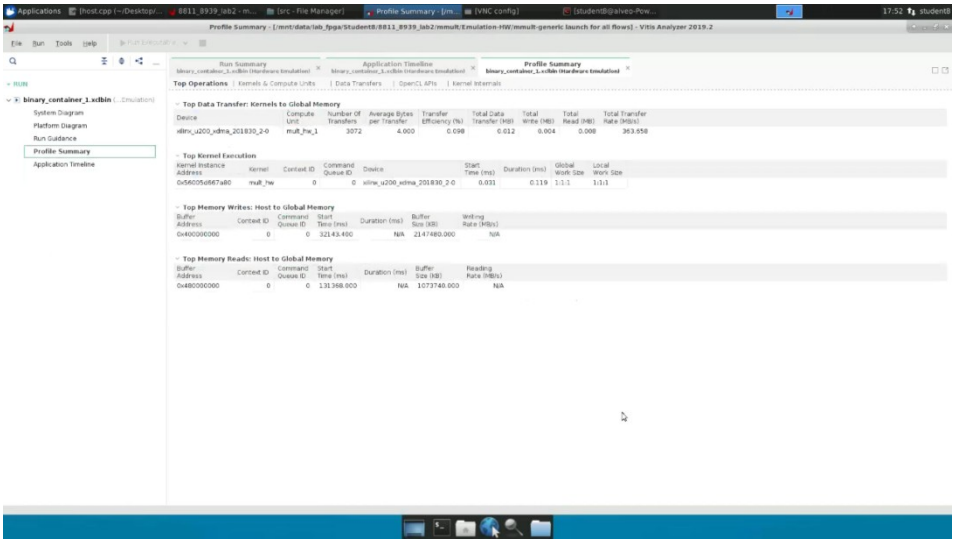
1. Software Emulation Application Timeline



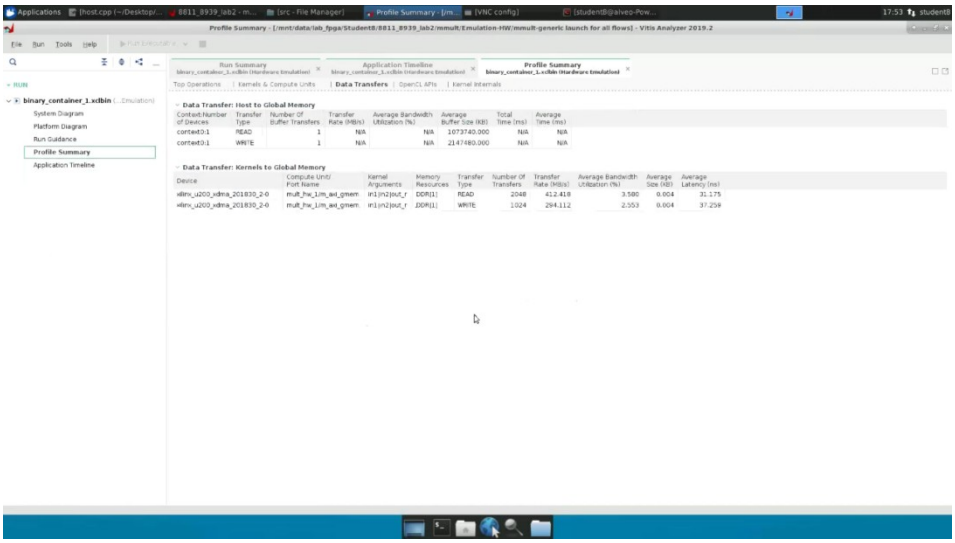
2. Hardware Emulation - Application Timeline



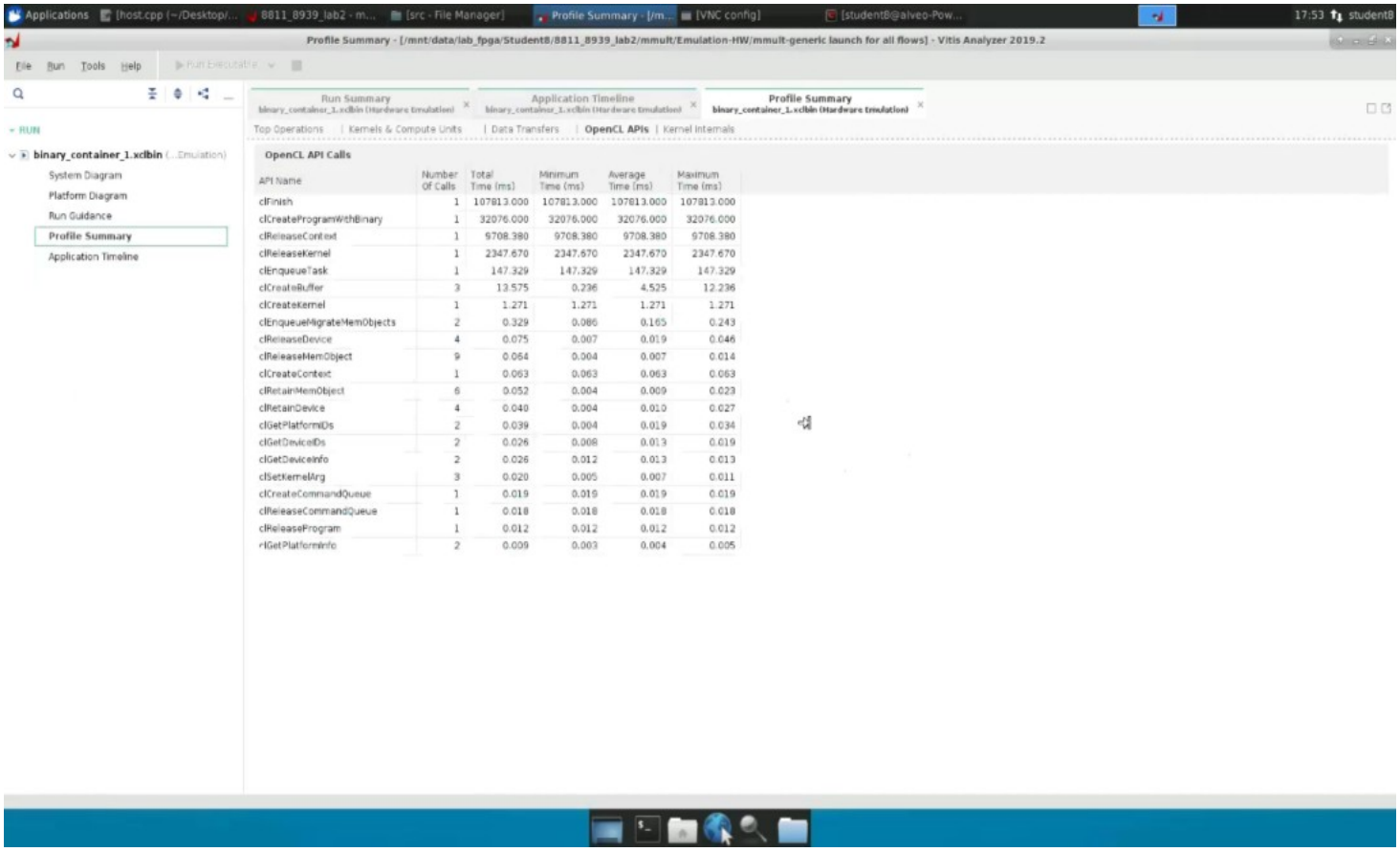
3. Hardware Emulation - Top Operations



4. Hardware Emulation - Data Transfers



5. Hardware Emulation - OpenCL APIs



API Name	Number Of Calls	Total Time (ms)	Minimum Time (ms)	Average Time (ms)	Maximum Time (ms)
clFinish	1	107813.000	107813.000	107813.000	107813.000
clCreateProgramWithBinary	1	32076.000	32076.000	32076.000	32076.000
clReleaseContext	1	9708.380	9708.380	9708.380	9708.380
clReleaseKernel	1	2347.670	2347.670	2347.670	2347.670
clEnqueueTask	1	147.329	147.329	147.329	147.329
clCreateBuffer	3	13.575	0.236	4.525	12.236
clCreateKernel	1	1.271	1.271	1.271	1.271
clEnqueueMigrateMemObjects	2	0.329	0.086	0.165	0.243
clReleaseDevice	4	0.075	0.007	0.019	0.046
clReleaseMemObject	9	0.064	0.004	0.007	0.014
clCreateContext	1	0.063	0.063	0.063	0.063
clRetainMemObject	6	0.052	0.004	0.009	0.023
clRetainDevice	4	0.040	0.004	0.010	0.027
clGetPlatformIDs	2	0.039	0.004	0.019	0.034
clGetDeviceIDs	2	0.026	0.008	0.013	0.019
clGetDeviceInfo	2	0.026	0.012	0.013	0.013
clSetKernelArg	3	0.020	0.005	0.007	0.011
clCreateCommandQueue	1	0.019	0.019	0.019	0.019
clReleaseCommandQueue	1	0.018	0.018	0.018	0.018
clReleaseProgram	1	0.012	0.012	0.012	0.012
clGetPlatformInfo	2	0.009	0.003	0.004	0.005