AAHLS (Lab 3)

1. 簡介

本報告探討 AAHLS（Lab3），內容分為三個部分。第一部分著重在解釋 host program(host.cpp)中，透過 OpenCL 配置以及調用 u50 對 application program 進行硬體加速的流程。在二部分則著重在 kernel 的優化，分析不同優化方法對於 kernel 以及整體 application program 的 timeline 影響。第三部分則是 LAB3 的相關截圖。

2. 報告內容

#1:

(1) 內容:

整體 host program 分為幾個部分，分別是: Target Device 的識別、OpenCL 的 context、program 以及 CommandQueue 的建立、load Binary file 以及 create Kernel，再來是創建 system/global memory，設定 kernel argument 和 data transfer，然後是配置 CommandQueue，標示不同 kernel 之間的 dependency，已執行 kernel，最後是將結果讀回 system 以確認是否正確。(Option) : 可以透過 profiling 分析不同 kernel 以及 kernel 對 system 的執行狀況優化 host program 或 kernel design。接下來會進行進一步的講解。

首先先確認 main 的 argc 的數量是否是 4，接著依序進行 PCIE 上所有 platform 的搜尋，找到目標 platform 後進行同一 platform 下 devices 的搜尋，最終回傳目標 device(U50)。

接著創建 context，依據 context 創建 CommandQueue，並將 binary file load 進 memory，透過 context 和 xclbin_Memory 創建 program，再將 target device 和 program 綁定。

再來就能透過 program 創建 kernel，並分配 host memory(此 project 有 3 個)，以及 global memory(此 project 有 7 個)，接著分配 arguments 到各 kernel，為 kernel 開始運作做準備。在分配 host memory 部分，要注意 4K alignment，在傳輸 data 到 global memory 才能更迅速。

然後就可以開始 copy data 從 host memory 到 global memory 並且配置 kernel 的 CommandQueue。這邊有兩點要注意:
第一是從 host memory 傳輸 data 要使用 clEnqueueMigrateMemObjects，因為它能更靈活判斷傳輸 data 的時機，在確保資訊在 dependent command 執行前進行傳輸，而不用向 clEnqueueWriteBuffer / clEnqueueWriteBuffer 一樣會 halt 住，等待傳輸完成才能繼續執行。而 buffer size 在這邊是一個重要的點，過大的 buffer size 會導致 waste，而

過小則會使的 PCIE 的 utilization 太低，可以透過簡單實驗找出平衡點，根據具體情況進行 tradeoff。

第二則是 CommandQueue 的配置要描述清楚不同 kernel 間 dependency。尤其是 out-of-order 的模式，這樣才能讓 accelerator 的 utilization 最高。

最後則是執行 kernel 並將結果從 global memory transfer 到 host memory，並比較結果是否正確以及對整體系統進行 profiling。後兩步是 option，通常是在開發階段會採用。當然也不要忘了最終要 release mem object 以及 free host memory，否則執行時間久了會出現 memory leakage 問題。

#2:

(1) 內容:

從 baseline 可以看出，CommandQueue 雖然是 out-of-order 模式，但 host 並沒有真正設定 parallel execution，而是依序執行。

而到 KernelParallel 部分，CommandQueue 總算設定好不同 kernel 之間的 dependency，使得 KB 和 KA，KpB 和 KvConstantAdd 可以同時執行，此時 KB 要在 KpB 後執行，KA 要在 KvConstantAdd 後執行，而 KCalc 則在 KA 和 KB 之後。

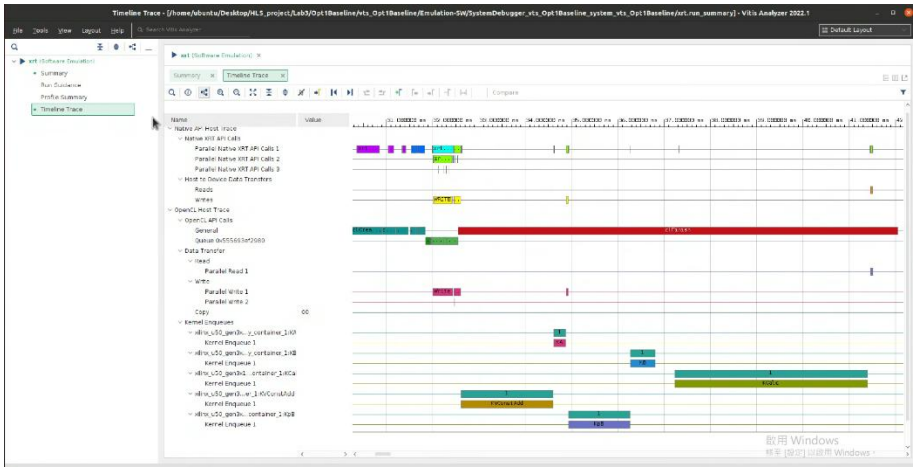
再來到 DataBurst 支援使用 burst 方式進行資料傳輸，這部分優化加快 kernel KpB 的執行速度，讓它跟 KvConstantAdd 一樣快，使 KB 執行更快，降低 KCalc 等待時間。

最後是使用 array partition 方式，進一步 boost kernel 的執行速度。

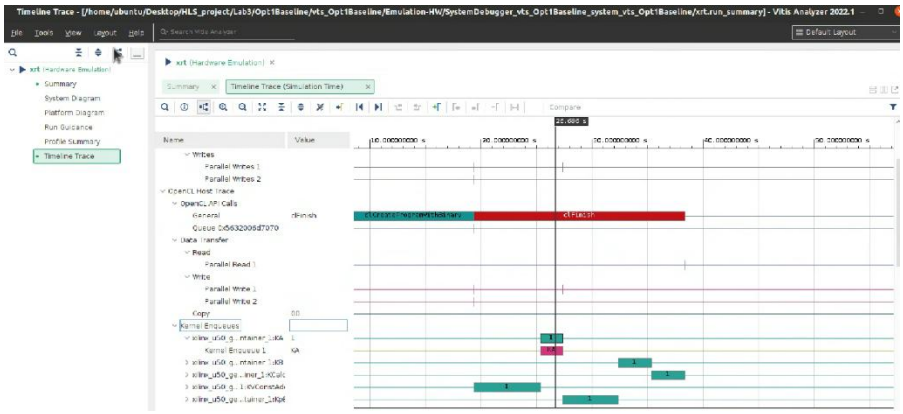
#3:

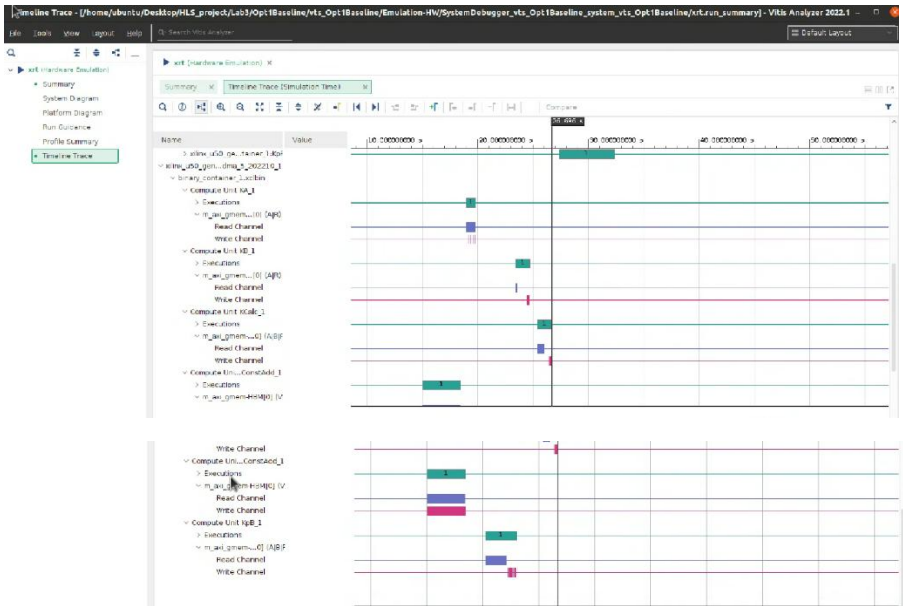
(1) Opt2Baseline

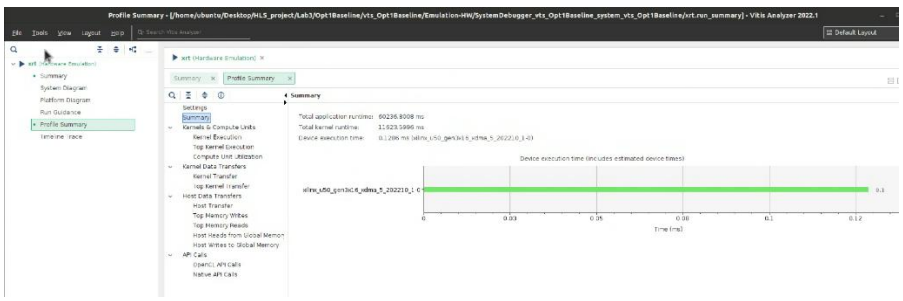
I. Software emulation



II. Hardware emulation

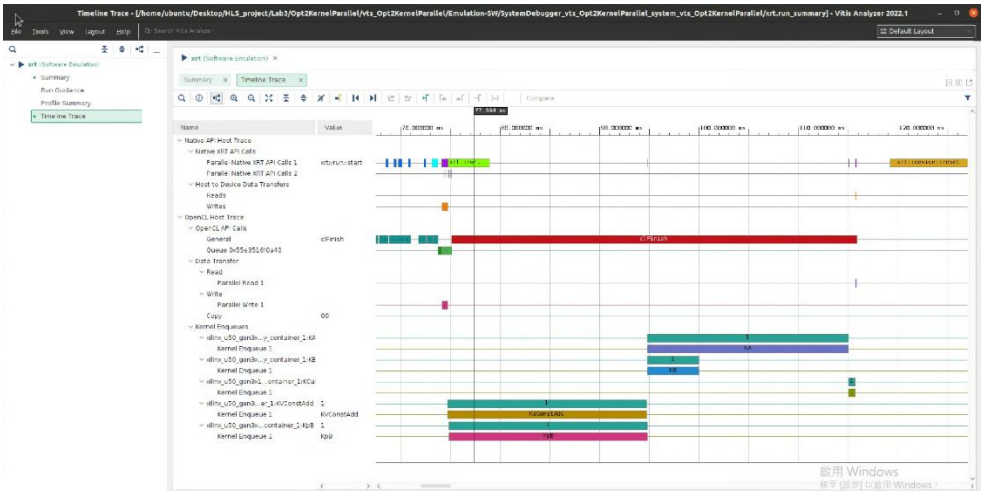


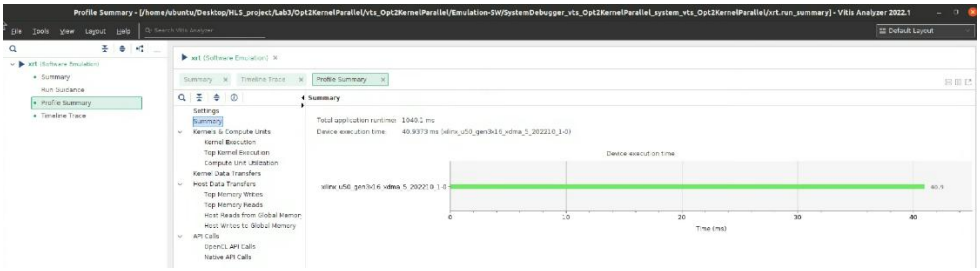




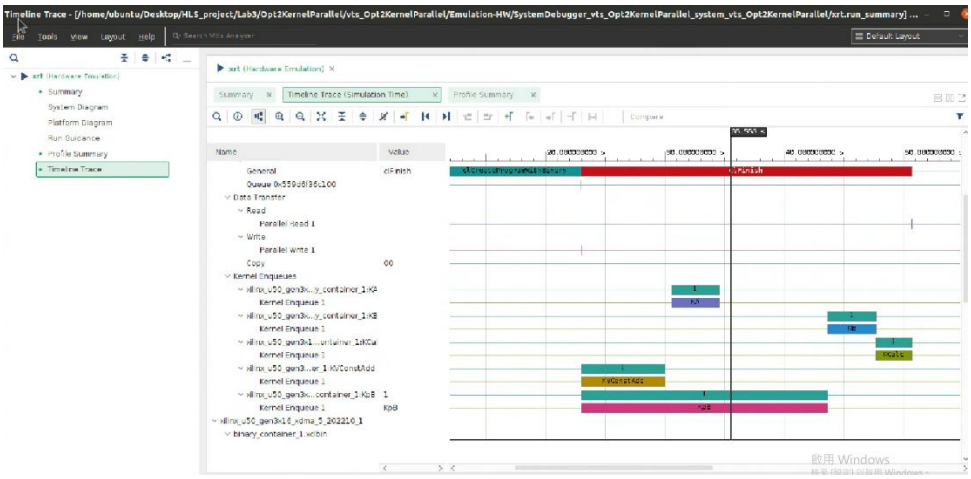
(2) Opt2KernelParallel

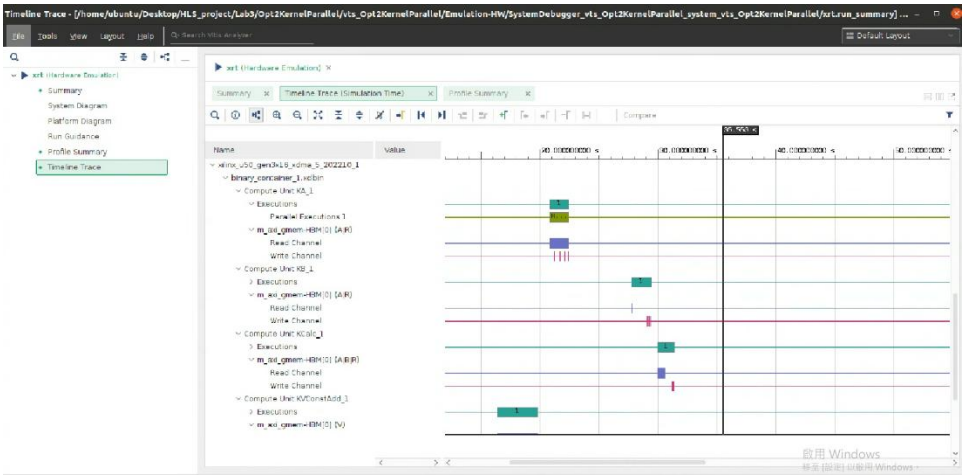
I. Software emulation

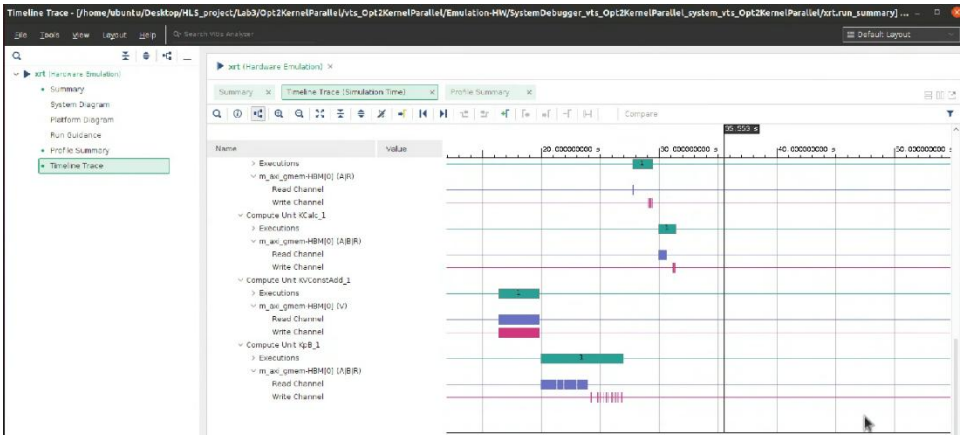




II. Hardware emulation

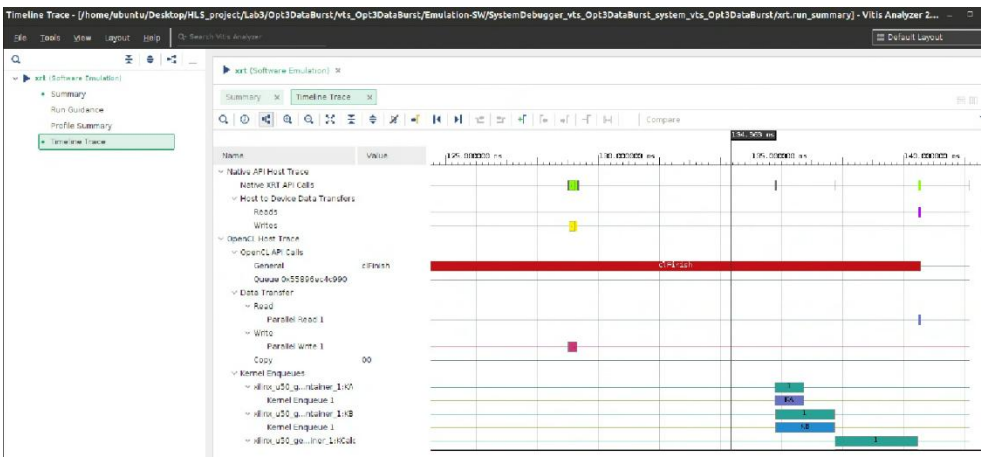


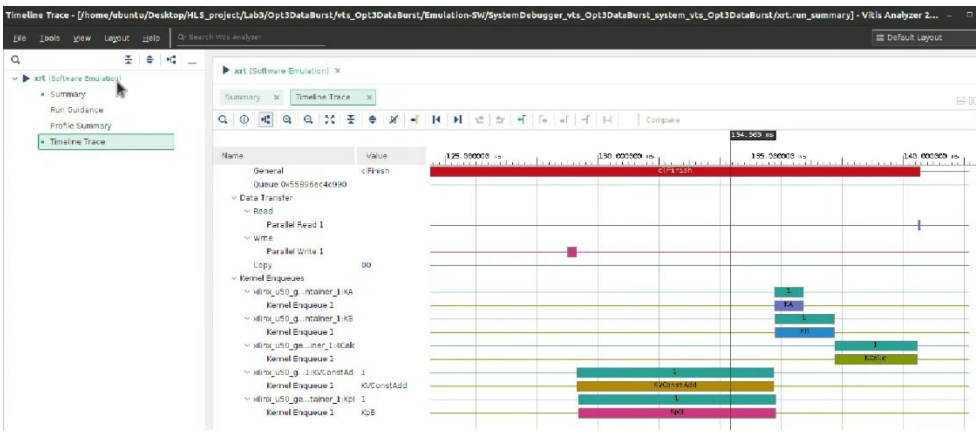


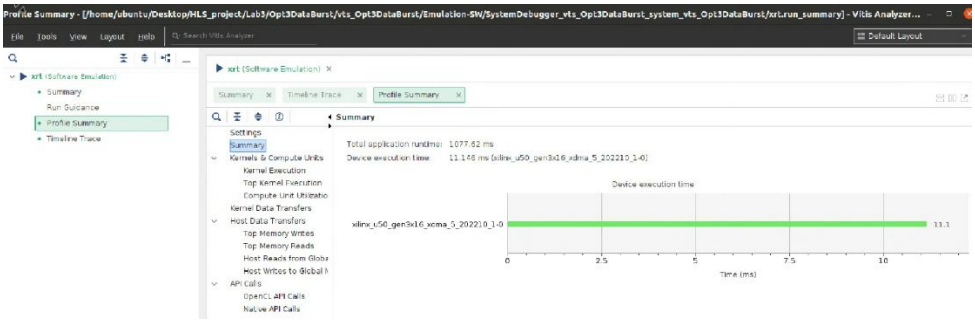


(3) Opt3DataBurst

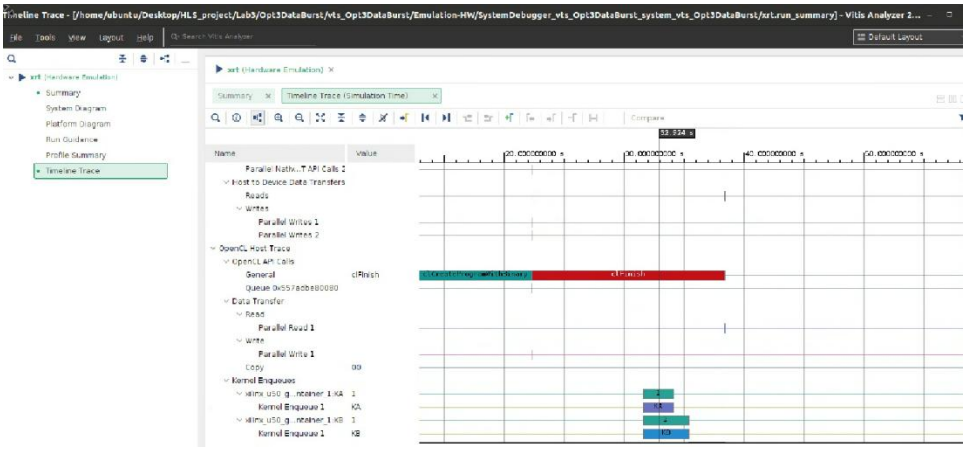
I. Software emulation

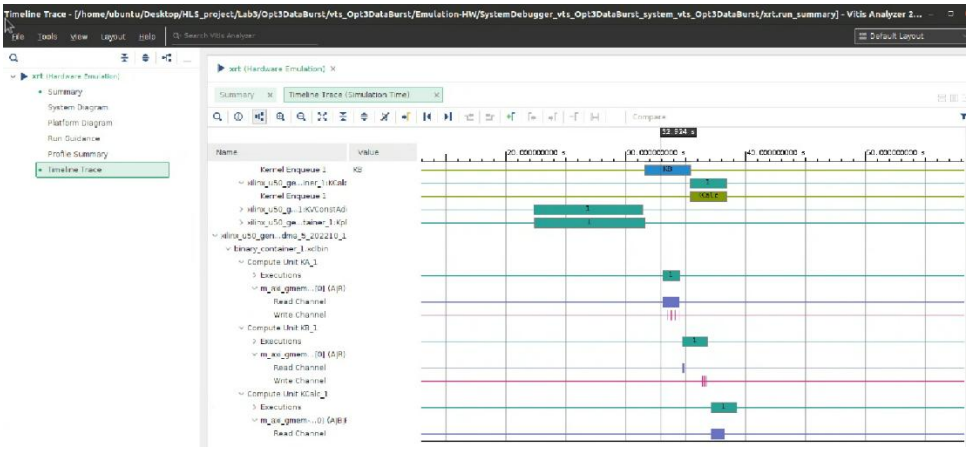


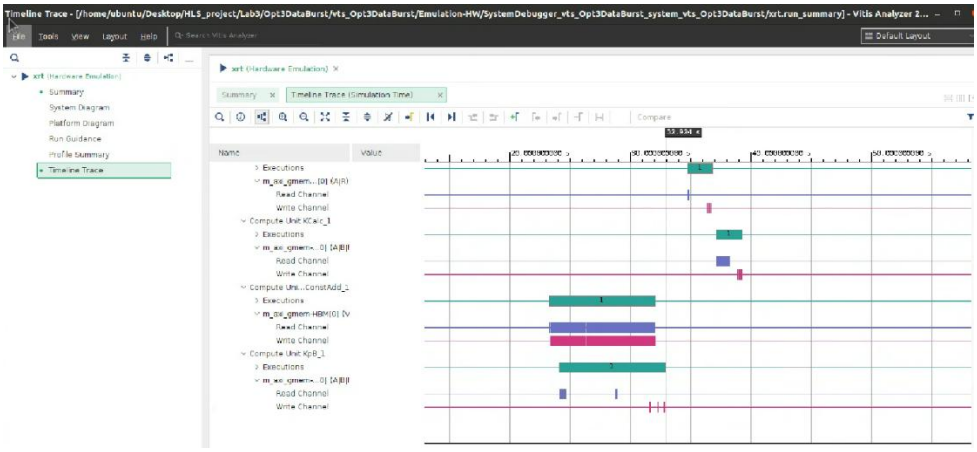


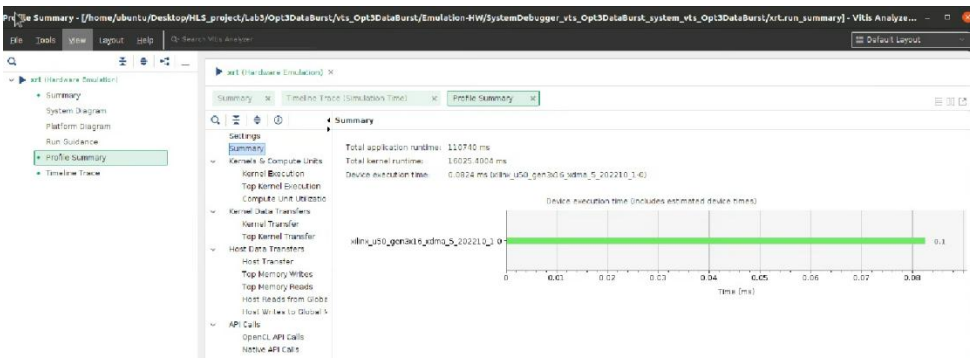


II. Hardware emulation



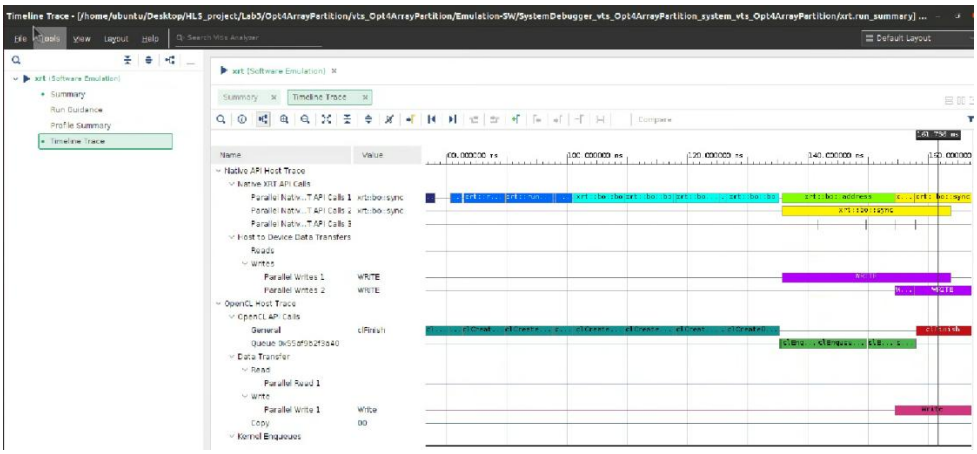


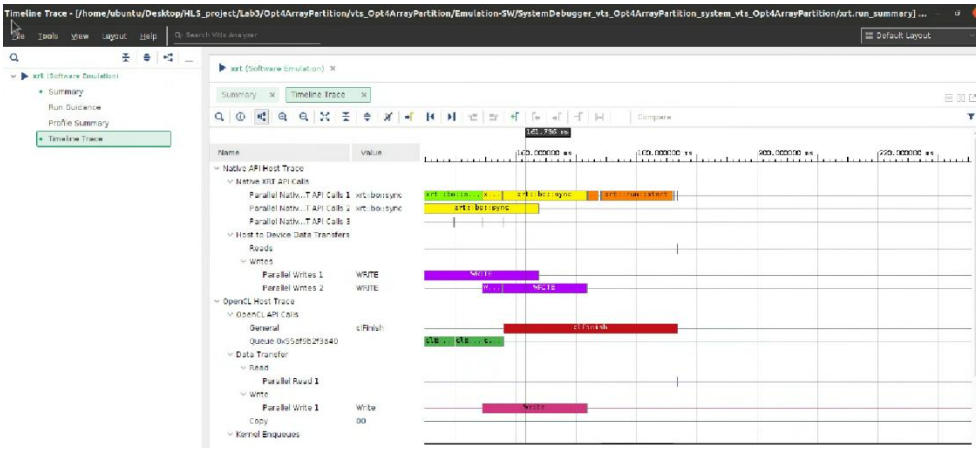


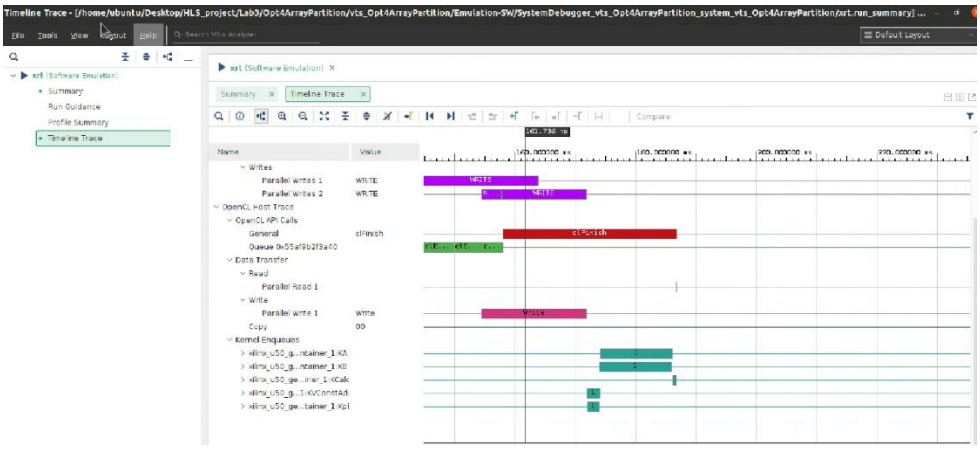


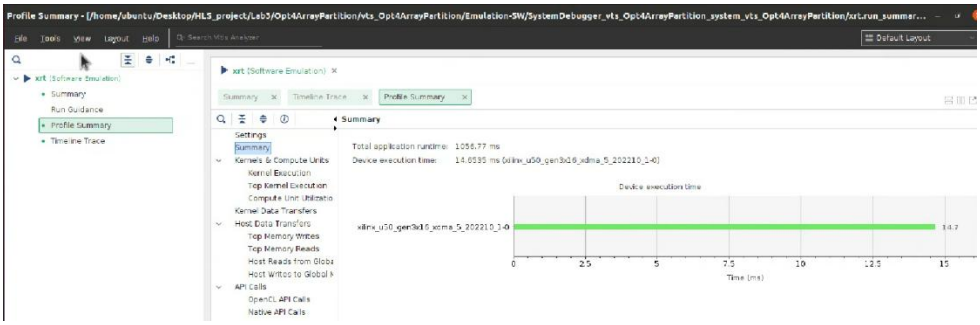
(4) Opt4ArrayPartition

I. Software emulation

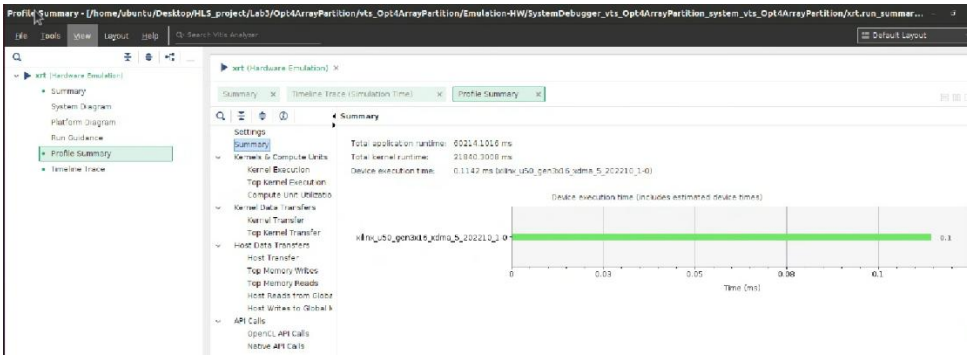


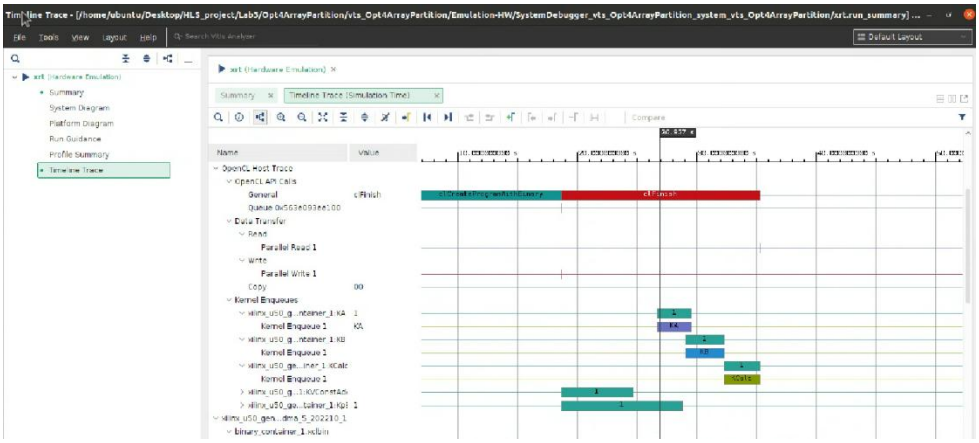


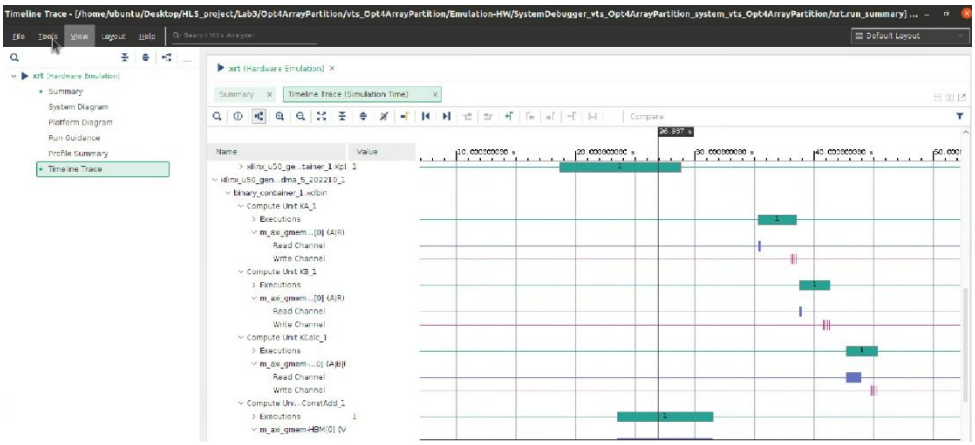


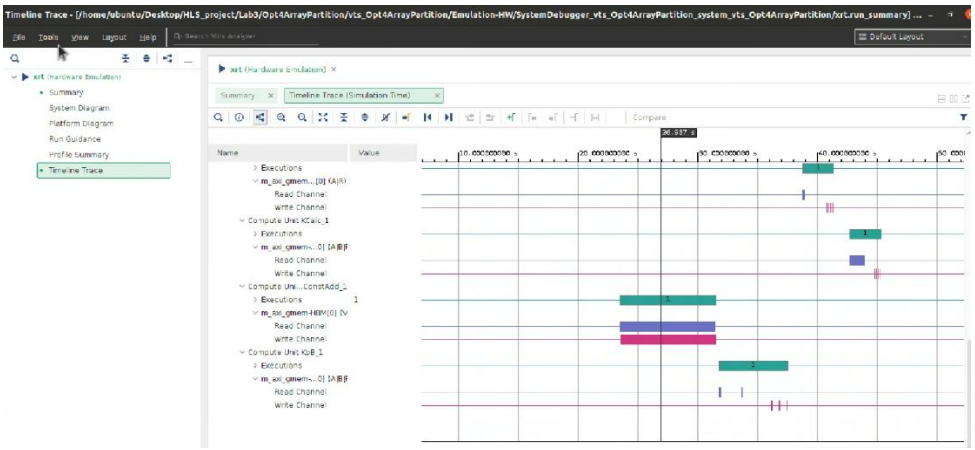


II. Hardware emulation

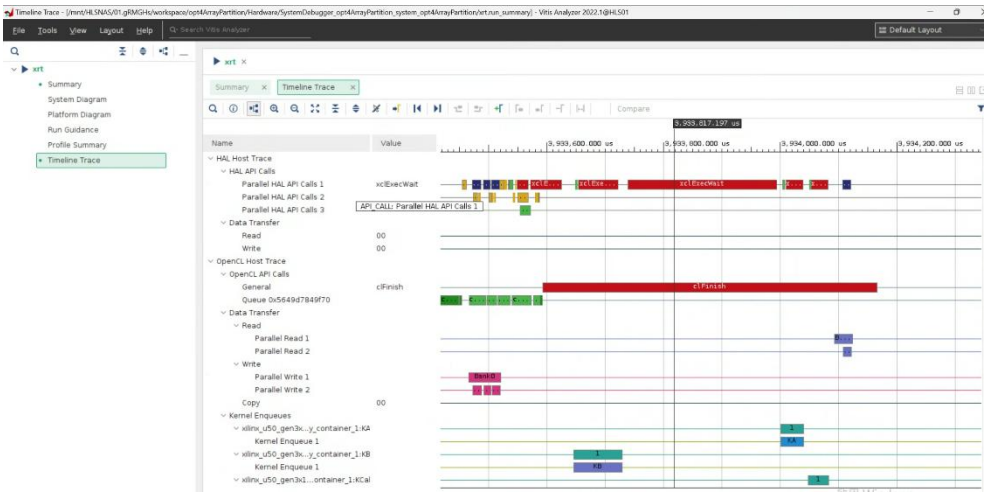


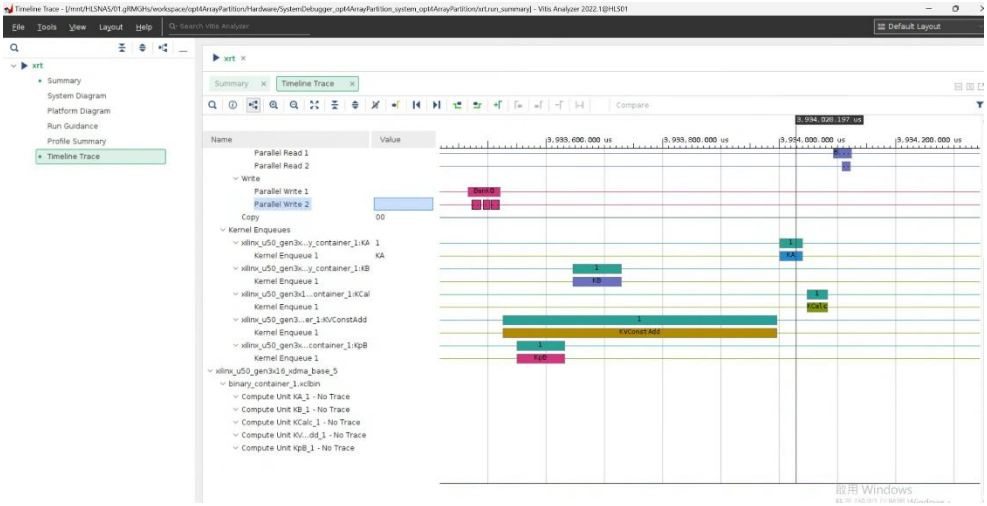






III. Hardware





Console | Problems | Vitis Log | Guidance

<terminated> (exit value: 0) SystemDebugger_opt4ArrayPartition_system_opt4ArrayPartition [OpenCL] /mnt/HLSNA

```
Host-Info: ============================================================
Host-Info: Verifying final results (only 5 first failures are printed)
Host-Info: ============================================================
Host-Info: Test Successful
HOST-Info: ============================================================
HOST-Info: (Step 7) Custom Profiling
HOST-Info: ============================================================
HOST-Info: ------------------------------------------------------------------------------
HOST-Info: Name           | type          | start        | end          | Duration(ms)
HOST-Info: ------------------------------------------------------------------------------
HOST-Info: K_KVConstAdd   | kernel        |   3933525468 |   3933995339 |     0.469871
HOST-Info: K_KpB          | kernel        |   3933549777 |   3933631475 |     0.081698
HOST-Info: K_KA           | kernel        |   3934000465 |   3934039694 |     0.039229
HOST-Info: K_KB           | kernel        |   3933645215 |   3933728152 |     0.082937
HOST-Info: K_KCalc        | kernel        |   3934047294 |   3934082775 |     0.035481
HOST-Info: Transfer_1     | mem (H<->G)   |   3933450120 |   3933520697 |     0.070577
HOST-Info: Transfer_2     | mem (H<->G)   |   3933453099 |   3933453099 |            0
HOST-Info: Transfer_3     | mem (H<->G)   |   3934091924 |   3934123853 |     0.031929
HOST-Info: ------------------------------------------------------------------------------
HOST-Info:     Kernels Execution Time (ms) :  0.557307  (K_KCalc'end - K_KVConstAdd'begin)
HOST-Info: Application Execution Time (ms) :  0.673733  (Transfer_3'end - Transfer_1'begin)
HOST-Info: ------------------------------------------------------------------------------


HOST-Info: DONE
```