HLS Lab 3 – Xilinx Alveo U50 XRT with OpenCL

B07902143 陳正康

Opt 1 Baseline

Profile Summary

Top Operations | Kernels & Compute Units | Data Transfers | OpenCL APIs

Top Data Transfer: Kernels to Global Memory

No Data. Please use 'v++ -l --profile_kernel' to monitor and report kernel data transfers and execution.

Top Kernel Execution

Kernel Instance Address	Kernel	Context ID	Command Queue ID	Device	Start Time (ms)	Duration (ms)	Global Work Size	Local Work Size
0x55fdbdfb29f0	KpB	0	0	xilinx_u50_gen3x16_xdma_201920_3-1	2168.750	0.269	1:1:1	1:1:1
0x55fdbdfb2ac0	KA	0	0	xilinx_u50_gen3x16_xdma_201920_3-1	2168.460	0.217	1:1:1	1:1:1
0x55fdbdfb2280	KVConstAdd	0	0	xilinx_u50_gen3x16_xdma_201920_3-1	2168.270	0.155	1:1:1	1:1:1
0x55fdbdfd3540	KCalc	0	0	xilinx_u50_gen3x16_xdma_201920_3-1	2169.130	0.136	1:1:1	1:1:1
0x55fdbdfd3a60	KB	0	0	xilinx_u50_gen3x16_xdma_201920_3-1	2169.040	0.070	1:1:1	1:1:1

Top Memory Writes: Host to Global Memory

Buffer Address	Context ID	Command Queue ID	Start Time (ms)	Duration (ms)	Buffer Size (KB)	Writing Rate (MB/s)
0x0	0	0	2168.070	0.073	16.384	224.030
0x4000	0	0	2168.080	0.071	12.288	172.463
0x7000	0	0	2168.710	0.024	12.288	505.721

Top Memory Reads: Host to Global Memory

Buffer Address	Context ID	Command Queue ID	Start Time (ms)	Duration (ms)	Buffer Size (KB)	Reading Rate (MB/s)
0xf000	0	0	2169.300	0.030	4.096	135.003

Top Operations | **Kernels & Compute Units** | Data Transfers | OpenCL APIs

Kernel Execution

Kernel	Number Of Enqueues	Total Time (ms)	Minimum Time (ms)	Average Time (ms)	Maximum Time (ms)
KA	1	0.217	0.217	0.217	0.217
KB	1	0.070	0.070	0.070	0.070
KCalc	1	0.136	0.136	0.136	0.136
KVConstAdd	1	0.155	0.155	0.155	0.155
KpB	1	0.269	0.269	0.269	0.269

Data Transfer: Host to Global Memory

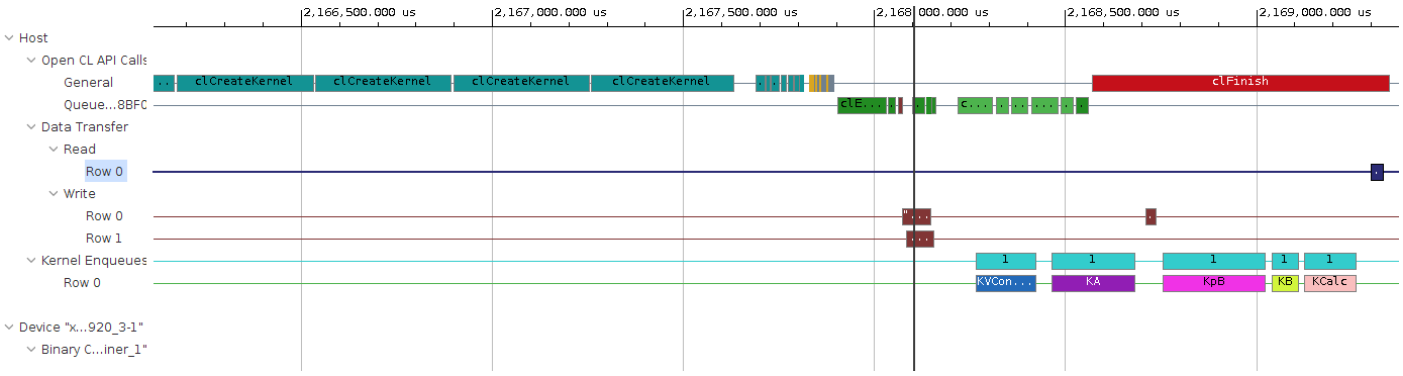
Context:Number of Devices	Transfer Type	Number Of Buffer Transfers	Transfer Rate (MB/s)	Average Bandwidth Utilization (%)	Average Buffer Size (KB)	Total Time (ms)	Average Time (ms)
context0:1	READ	1	135.003	1.406	4.096	0.030	0.030
context0:1	WRITE	3	242.825	2.529	13.653	0.169	0.056

OpenCL API Calls

API Name	Number Of Calls	Total Time (ms)	Minimum Time (ms)	Average Time (ms)	Maximum Time (ms)
clCreateProgramWithBinary	1	1770.100	1770.100	1770.100	1770.100
clGetDeviceInfo	4	164.741	0.004	41.185	91.962
clReleaseContext	1	74.908	74.908	74.908	74.908
clCreateContext	1	8.416	8.416	8.416	8.416
clCreateKernel	5	1.824	0.354	0.365	0.386
clReleaseProgram	1	1.422	1.422	1.422	1.422
clFinish	1	0.779	0.779	0.779	0.779
clEnqueueTask	5	0.265	0.031	0.053	0.089
clEnqueueMigrateMemObjects	7	0.230	0.004	0.033	0.127
clCreateBuffer	7	0.101	0.008	0.014	0.024
clGetEventProfilingInfo	26	0.076	0.002	0.003	0.008
clReleaseKernel	5	0.059	0.003	0.012	0.024
clReleaseEvent	13	0.040	0.002	0.003	0.004
clSetKernelArg	12	0.036	0.002	0.003	0.006
clGetPlatformIDs	2	0.026	0.004	0.013	0.022
clReleaseMemObject	7	0.019	0.002	0.003	0.004
clGetDeviceIDs	2	0.013	0.005	0.007	0.008
clCreateCommandQueue	1	0.012	0.012	0.012	0.012
clEnqueueWriteBuffer	1	0.011	0.011	0.011	0.011
clGetPlatformInfo	2	0.010	0.005	0.005	0.006
clBuildProgram	1	0.010	0.010	0.010	0.010
clEnqueueBarrierWithWaitList	1	0.009	0.009	0.009	0.009
clReleaseCommandQueue	1	0.006	0.006	0.006	0.006
clReleaseDevice	1	0.005	0.005	0.005	0.005

Application Timeline

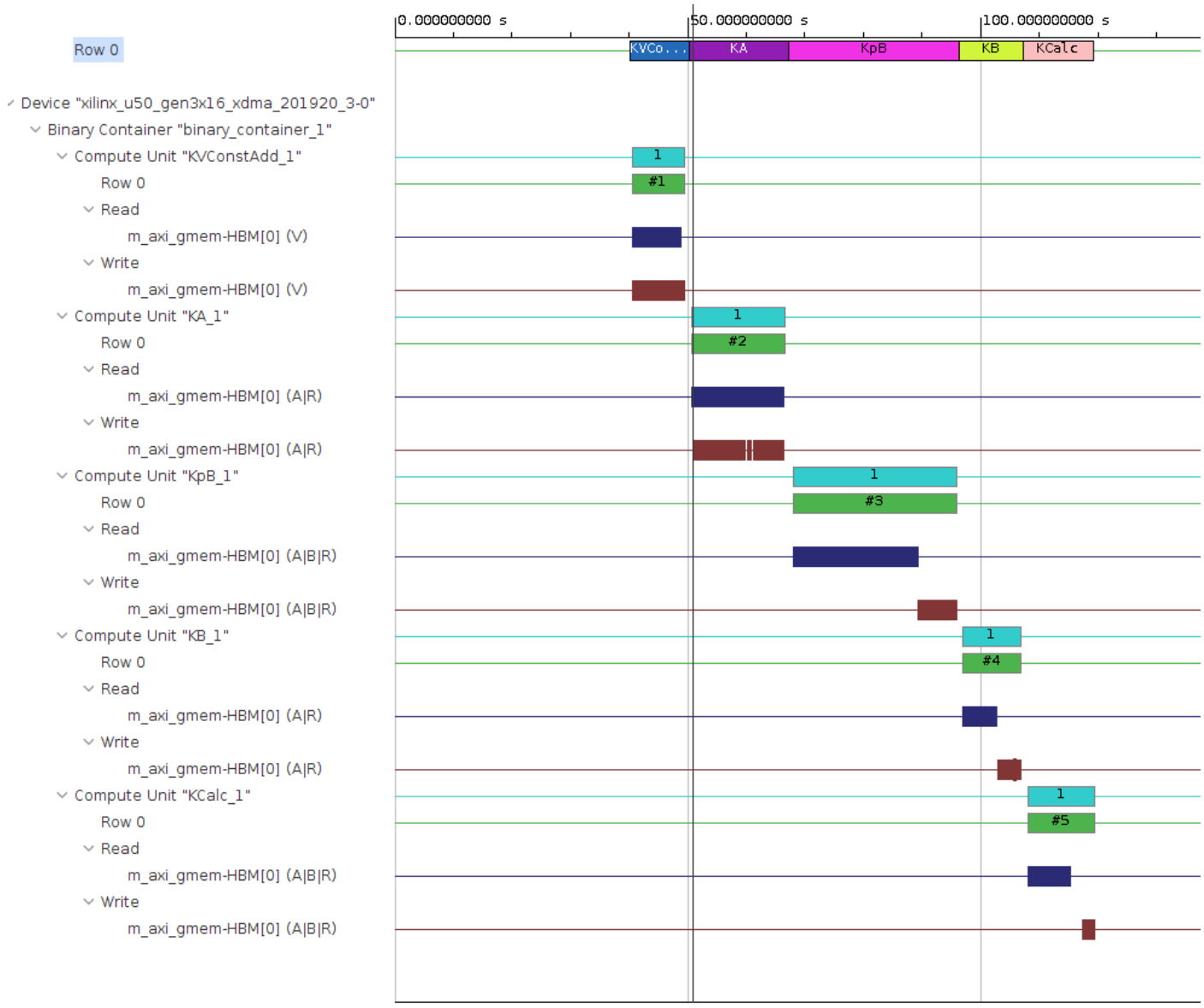
下圖是在硬體上 overall 的 timing (hardware profile)，kernel function 都是 in-order 執行，前面 OpenCL API Setup 花了不少時間



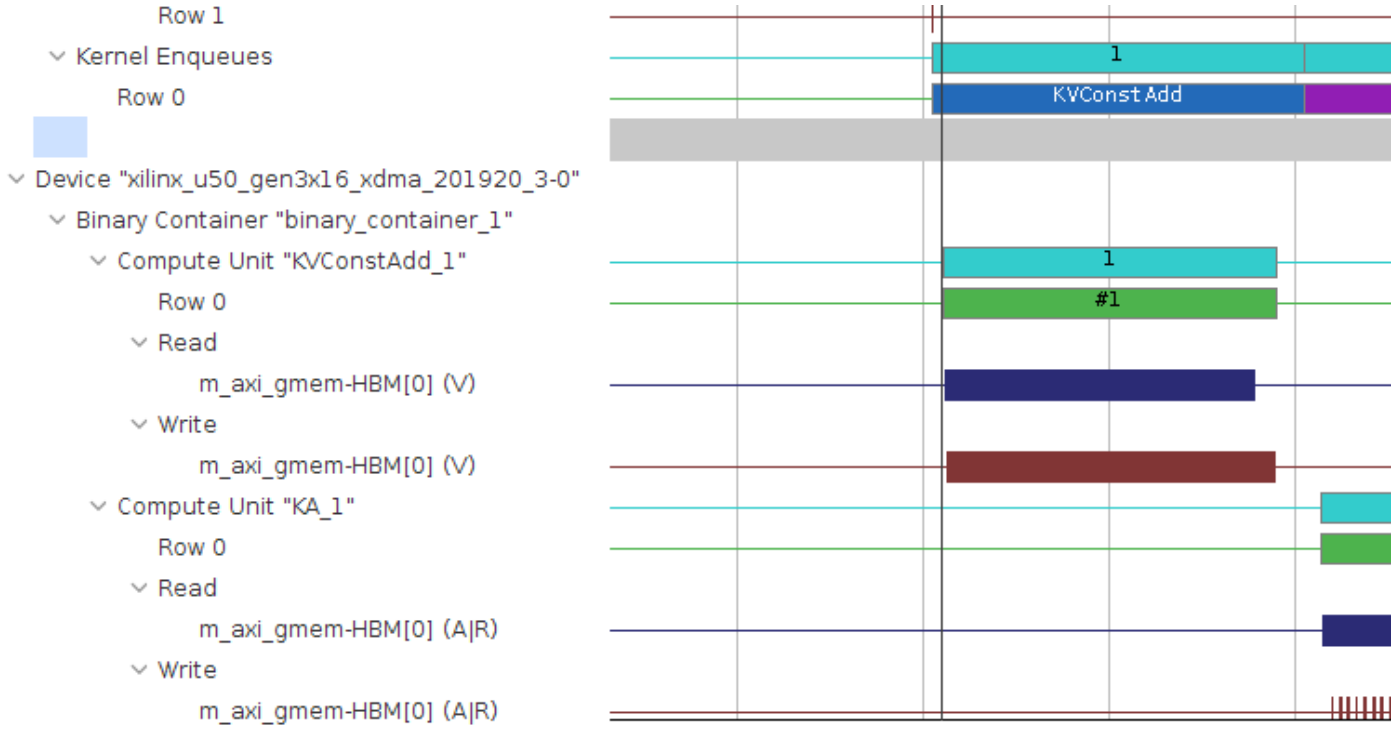
Analysis with Hardware Emulation

因為 Hardware Profile 好像沒有 Off-chip memory 和 Kernel function 間的 IO，因此以下由 hardware emulation profile 來分析

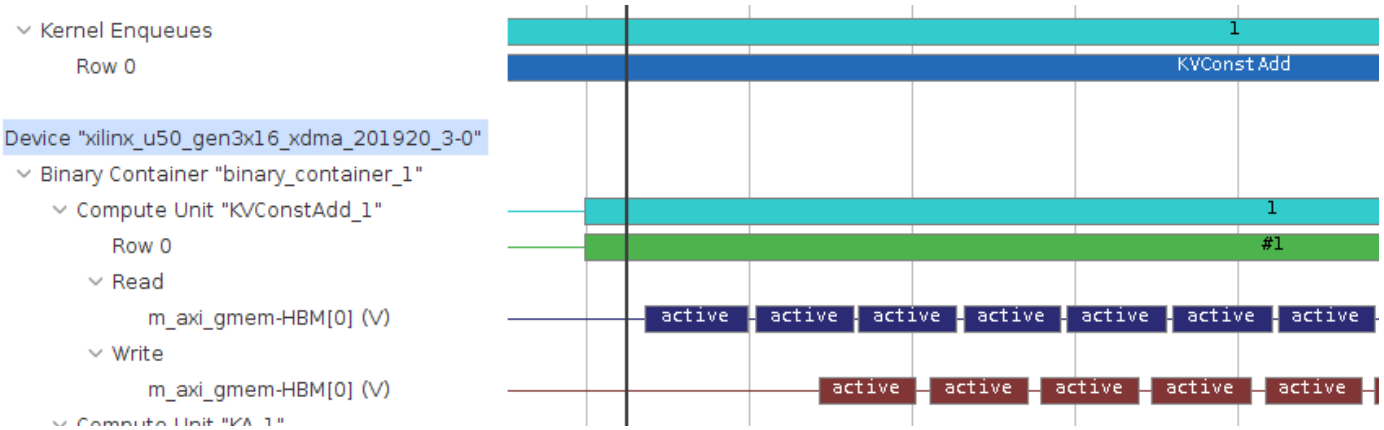
以下是 overall 的 timing



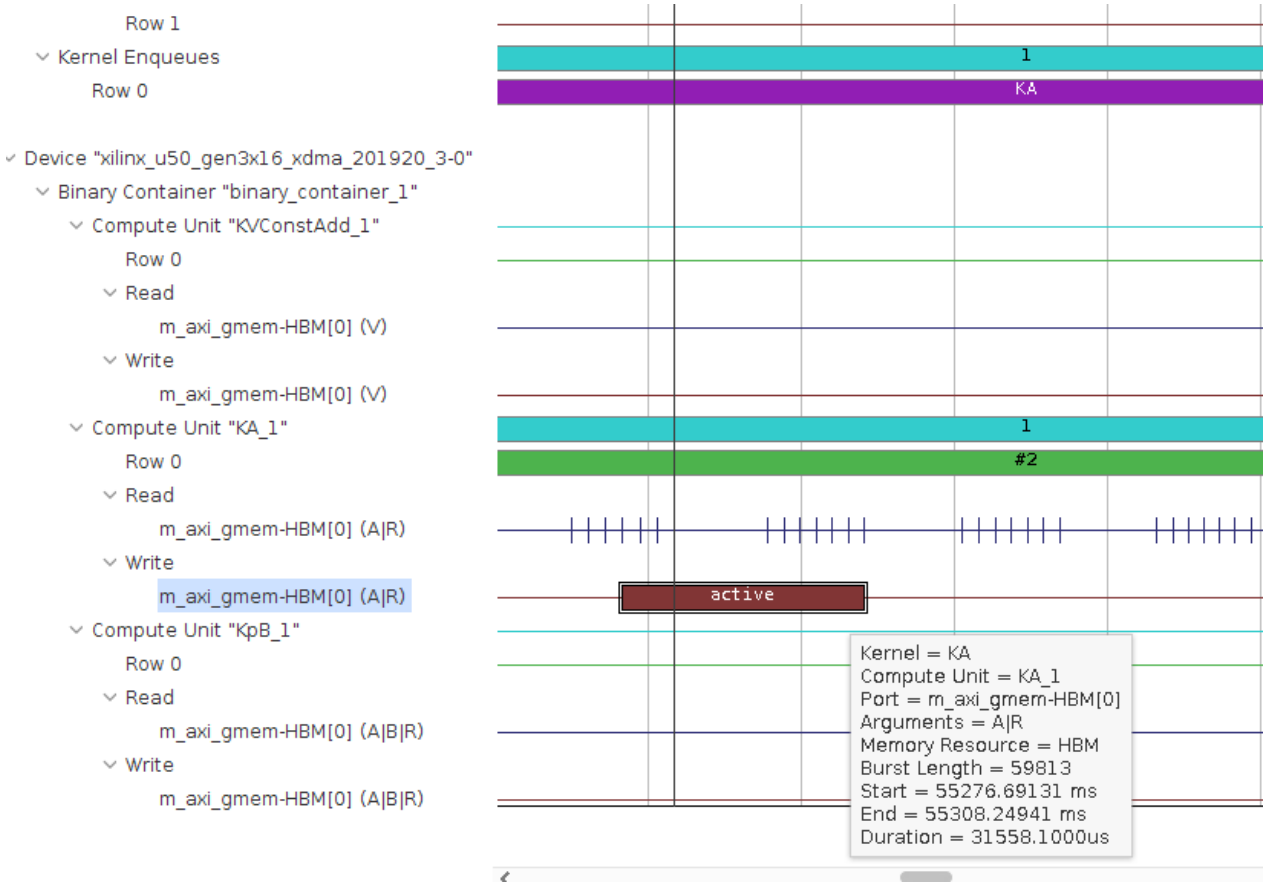
下圖中的 kernel function kvconstadd 開始後隔一段時間 memory 才被讀
可以看出 clEnqueueMigrateMemObject 有 transfer data on demand 的特性



再放大一點看，read 和 write 之間有 hide latency，2 次 read 之間的 gap 猜測是 setup 時間



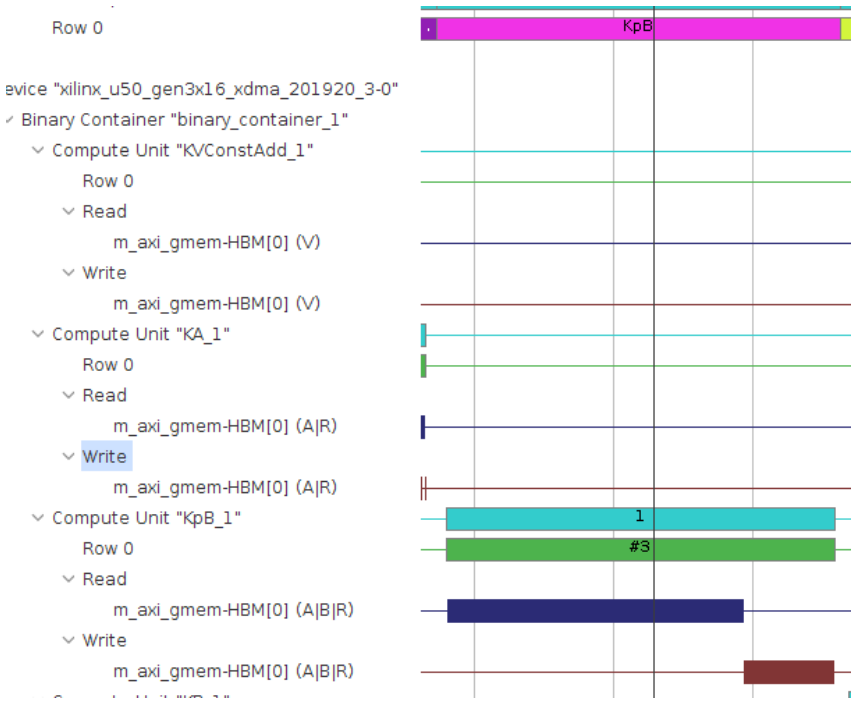
再來看 ka，有很多次小的 read，對應一段大的 write，是因為 write 有 burst 的關係，一次寫多個來降低 overhead，而 read 卻沒有



ka 的 code 如下，可以推測是因 A 陣列的讀取並不是 sequential 造成 read 沒有 burst

```
loop_st_1:
for (int i=0; i<SIZE_BUF_KA; i++) {
    R[i] = A[2*i]*3 + A[2*i+1]*5 + A[2*i+2]*7 + A[2*i+3]*9;
}
```

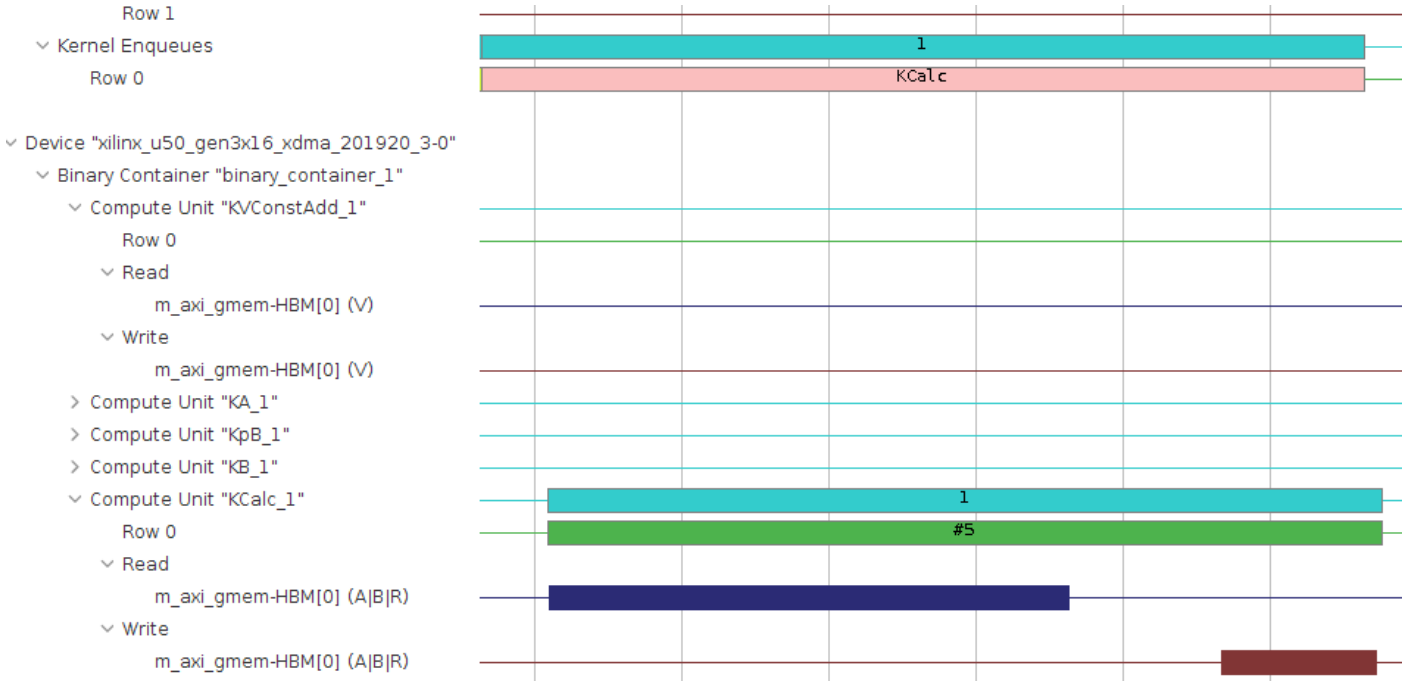
再來看 kpb，發現 read 和 write 並沒有並行



看 KpB 的 code 發現，因為有 local buffer TMP_RES，要先等 A 和 B 全部讀完之後才會寫入 R，在 KB 中也有類似的情形，有優化空間

```
for(int i=0; i < SIZE_BUF_KpB; i+=1) {
    TMP_RES[i] = A[i] + B[i];
}

for(int i=0; i < SIZE_BUF_KpB; i+=1) {
    R[i] = TMP_RES[i] % 3;
}
```

再看下圖 KCalc 中明顯的是 read 和 write 間的 gap



原因是下圖的中間那個迴圈只做運算而沒有 io，有優化空間

```
int TMP_A[SIZE_RES], TMP_B[SIZE_RES];
for (int i=0; i<SIZE_RES; i++) {
    #pragma HLS PIPELINE
    TMP_A[i] = A[i]; TMP_B[i] = B[i];
}


for (int i=0; i<SIZE_RES; i++) {
    val1 = (TMP_A[i] - TMP_B[i]) * (TMP_A[i] + TMP_B[i]);

    if (val1 >= 0)
        val2 = val1 % 3;
    else
        val2 = (val1 % 6) * val1;

    TMP_R[i] = val2;
}

for (int i=0; i<SIZE_RES; i++) {
    #pragma HLS PIPELINE
    R[i] = TMP_R[i];
}
```

Opt 2 Kernel Parallel

Profile Summary

Top Kernel Execution

Kernel Instance Address	Kernel	Context ID	Command Queue ID	Device	Start Time (ms)	Duration (ms)	Global Work Size	Local Work Size
0x55f6cce879e0	KpB	0	0	xilinx_u50_gen3x16_xdma_201920_3-1	207.325	0.296	1:1:1	1:1:1
0x55f6cce87ab0	KA	0	0	xilinx_u50_gen3x16_xdma_201920_3-1	207.433	0.224	1:1:1	1:1:1
0x55f6cce87270	KVConstAdd	0	0	xilinx_u50_gen3x16_xdma_201920_3-1	207.218	0.168	1:1:1	1:1:1
0x55f6ccea8530	KCalc	0	0	xilinx_u50_gen3x16_xdma_201920_3-1	207.736	0.141	1:1:1	1:1:1
0x55f6ccea8a50	KB	0	0	xilinx_u50_gen3x16_xdma_201920_3-1	207.639	0.066	1:1:1	1:1:1

Top Memory Writes: Host to Global Memory

Buffer Address	Context ID	Command Queue ID	Start Time (ms)	Duration (ms)	Buffer Size (KB)	Writing Rate (MB/s)
0x0	0	0	207.120	0.053	40.960	772.393

Top Memory Reads: Host to Global Memory

Buffer Address	Context ID	Command Queue ID	Start Time (ms)	Duration (ms)	Buffer Size (KB)	Reading Rate (MB/s)
0xf000	0	0	207.912	0.025	4.096	163.604

Kernel Execution

Kernel	Number Of Enqueues	Total Time (ms)	Minimum Time (ms)	Average Time (ms)	Maximum Time (ms)
KA	1	0.224	0.224	0.224	0.224
KB	1	0.066	0.066	0.066	0.066
KCalc	1	0.141	0.141	0.141	0.141
KVConstAdd	1	0.168	0.168	0.168	0.168
KpB	1	0.296	0.296	0.296	0.296

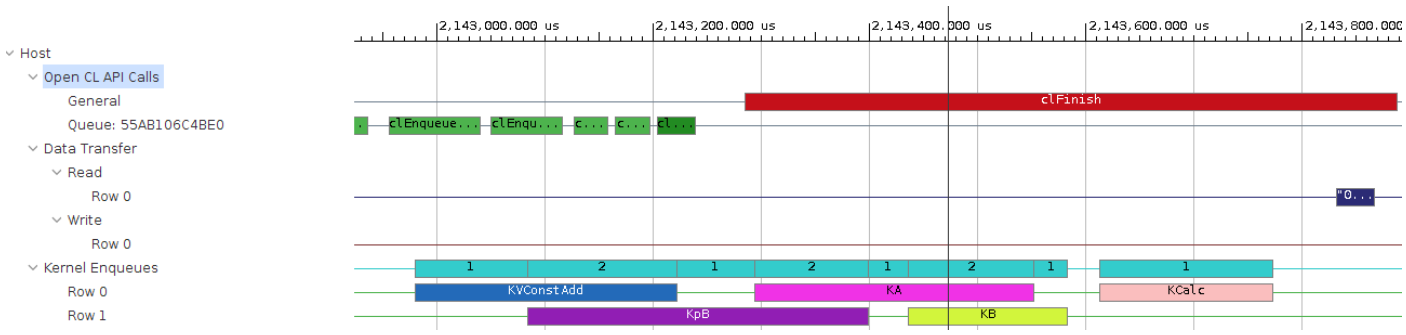
Data Transfer: Host to Global Memory

Context:Number of Devices	Transfer Type	Number Of Buffer Transfers	Transfer Rate (MB/s)	Average Bandwidth Utilization (%)	Average Buffer Size (KB)	Total Time (ms)	Average Time (ms)
context0:1	READ	1	163.604	1.704	4.096	0.025	0.025
context0:1	WRITE	1	772.393	8.046	40.960	0.053	0.053

OpenCL API Calls

API Name	Number Of Calls	Total Time (ms)	Minimum Time (ms)	Average Time (ms)	Maximum Time (ms)
clGetDeviceInfo	4	156.264	0.005	39.066	91.977
clReleaseContext	1	71.363	71.363	71.363	71.363
clCreateProgramWithBinary	1	28.704	28.704	28.704	28.704
clCreateContext	1	5.226	5.226	5.226	5.226
clCreateKernel	5	1.918	0.368	0.384	0.409
clReleaseProgram	1	1.520	1.520	1.520	1.520
clFinish	1	0.418	0.418	0.418	0.418
clEnqueueTask	5	0.342	0.033	0.068	0.086
clEnqueueMigrateMemObjects	3	0.170	0.009	0.057	0.147
clCreateBuffer	7	0.102	0.006	0.015	0.035
clReleaseKernel	5	0.054	0.004	0.011	0.018
clGetEventProfilingInfo	16	0.054	0.002	0.003	0.013
clSetKernelArg	12	0.040	0.002	0.003	0.007
clReleaseEvent	8	0.031	0.003	0.004	0.008
clGetPlatformIDs	2	0.023	0.003	0.011	0.019
clReleaseMemObject	7	0.019	0.002	0.003	0.004
clGetDeviceIDs	2	0.012	0.005	0.006	0.007
clCreateCommandQueue	1	0.011	0.011	0.011	0.011
clReleaseCommandQueue	1	0.010	0.010	0.010	0.010
clReleaseDevice	1	0.009	0.009	0.009	0.009
clEnqueueBarrierWithWaitList	1	0.009	0.009	0.009	0.009
clBuildProgram	1	0.007	0.007	0.007	0.007
clGetPlatformInfo	2	0.005	0.002	0.003	0.003

Application Timeline



Compare to Opt 1 Baseline: Data Transfer

OPT2 在 host to global memory 以及 global memory to kernel 的資料傳輸都做了優化，把多個 buffer 合在一起傳輸

```
cl_mem Mem_Pointers_1[3], Mem_Pointers_2[4];

Mem_Pointers_1[0] = GlobMem_BUF_DataIn_1;
Mem_Pointers_1[1] = GlobMem_BUF_DataIn_2;
Mem_Pointers_1[2] = GlobMem_BUF_DataIn_3;

errCode = clEnqueueMigrateMemObjects(Command_Queue, 3, Mem_Pointers_1, 0, 0, NULL, &Mem_op_event[0]);
if (errCode != CL_SUCCESS) {
    cout << endl << "Host-Error: Failed Migrate GlobMem_BUF_DataIn_1, GlobMem_BUF_DataIn_2, GlobMem_B
    return EXIT_FAILURE;
}

Mem_Pointers_2[0] = GlobMem_BUF_KpB;
Mem_Pointers_2[1] = GlobMem_BUF_KA;
Mem_Pointers_2[2] = GlobMem_BUF_KB;
Mem_Pointers_2[3] = GlobMem_BUF_RES;

errCode = clEnqueueMigrateMemObjects(Command_Queue, 4, Mem_Pointers_2, CL_MIGRATE_MEM_OBJECT_CONTENT_
if (errCode != CL_SUCCESS) {
    cout << endl << "Host-Error: Failed Migrate GlobMem_BUF_KpB, GlobMem_BUF_KA, GlobMem_BUF_KB, Glob
    return EXIT_FAILURE;
}
```

比較 OPT1 和 OPT2，後者的 write 次數(host to global memory)為前者的 1/3 倍，bandwidth utilization 為 3 倍

Top Operations | Kernels & Compute Units | **Data Transfers** | OpenCL APIs

Data Transfer: Host to Global Memory

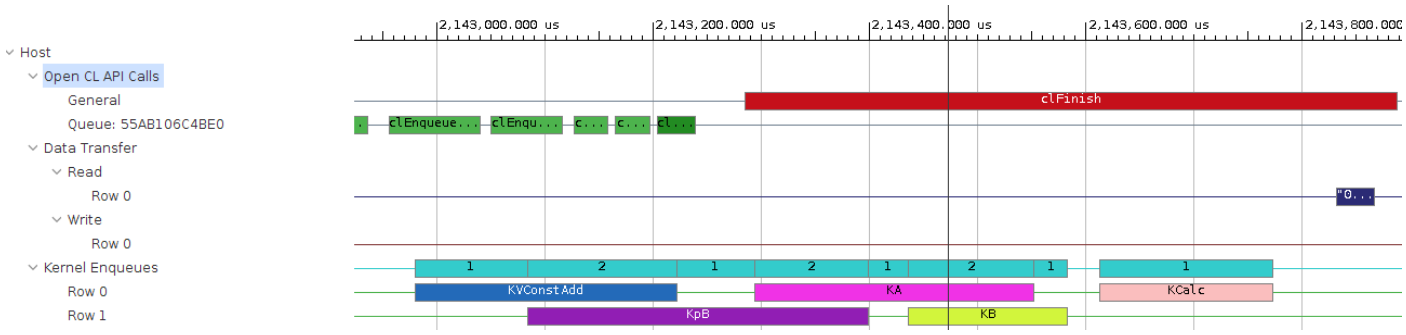
Context:Number of Devices	Transfer Type	Number Of Buffer Transfers	Transfer Rate (MB/s)	Average Bandwidth Utilization (%)	Average Buffer Size (KB)	Total Time (ms)	Average Time (ms)
context0:1	READ	1	135.003	1.406	4.096	0.030	0.030
context0:1	WRITE	3	242.825	2.529	13.653	0.169	0.056

Data Transfer: Host to Global Memory

Context:Number of Devices	Transfer Type	Number Of Buffer Transfers	Transfer Rate (MB/s)	Average Bandwidth Utilization (%)	Average Buffer Size (KB)	Total Time (ms)	Average Time (ms)
context0:1	READ	1	163.604	1.704	4.096	0.025	0.025
context0:1	WRITE	1	772.393	8.046	40.960	0.053	0.053

Compare to Opt 1 Baseline: Kernel Parallelism

由 application timeline 可看出 KpB 和 KVConstAdd 平行執行



Opt 3 Data Burst

Profile Summary

Top Data Transfer: Kernels to Global Memory

Device	Compute Unit	Number Of Transfers	Average Bytes per Transfer	Transfer Efficiency (%)	Total Data Transfer (MB)	Total Write (MB)	Total Read (MB)	Total Transfer Rate (MB/s)
xilinx_u50_gen3x16_xdma_201920_3-0	KpB_1	6336	5.818	0.142	0.037	0.012	0.025	1175.510
xilinx_u50_gen3x16_xdma_201920_3-0	KA_1	4160	4.923	0.120	0.020	0.004	0.016	1185.190
xilinx_u50_gen3x16_xdma_201920_3-0	KCalc_1	2112	5.818	0.142	0.012	0.004	0.008	1175.510
xilinx_u50_gen3x16_xdma_201920_3-0	KVConstAdd_1	512	64.000	1.563	0.033	0.016	0.016	1163.640
xilinx_u50_gen3x16_xdma_201920_3-0	KB_1	256	64.000	1.563	0.016	0.004	0.012	1181.540

Top Kernel Execution

Kernel Instance Address	Kernel	Context ID	Command Queue ID	Device	Start Time (ms)	Duration (ms)	Global Work Size	Local Work Size
0x56031036ff40	KpB	0	0	xilinx_u50_gen3x16_xdma_201920_3-0	0.086	0.048	1:1:1	1:1:1
0x56031036f900	KA	0	0	xilinx_u50_gen3x16_xdma_201920_3-0	0.060	0.026	1:1:1	1:1:1
0x56031036faa0	KCalc	0	0	xilinx_u50_gen3x16_xdma_201920_3-0	0.153	0.022	1:1:1	1:1:1
0x56031036f9d0	KB	0	0	xilinx_u50_gen3x16_xdma_201920_3-0	0.134	0.019	1:1:1	1:1:1
0x56031035a4d0	KVConstAdd	0	0	xilinx_u50_gen3x16_xdma_201920_3-0	0.045	0.016	1:1:1	1:1:1

Top Memory Writes: Host to Global Memory

Buffer Address	Context ID	Command Queue ID	Start Time (ms)	Duration (ms)	Buffer Size (KB)	Writing Rate (MB/s)
0x4000	0	0	41134.500	N/A	12.288	N/A
0x0	0	0	41134.400	N/A	16.384	N/A
0x7000	0	0	70153.500	N/A	12.288	N/A

Top Memory Reads: Host to Global Memory

Buffer Address	Context ID	Command Queue ID	Start Time (ms)	Duration (ms)	Buffer Size (KB)	Reading Rate (MB/s)
0xf000	0	0	126162.000	N/A	4.096	N/A

Kernel Execution (includes estimated device times)

Kernel	Number Of Enqueues	Total Time (ms)	Minimum Time (ms)	Average Time (ms)	Maximum Time (ms)
KA	1	0.026	0.026	0.026	0.026
KB	1	0.019	0.019	0.019	0.019
KCalc	1	0.022	0.022	0.022	0.022
KVConstAdd	1	0.016	0.016	0.016	0.016
KpB	1	0.048	0.048	0.048	0.048

Compute Unit Utilization (includes estimated device times)

Device	Compute Unit	Kernel	Global Work Size	Local Work Size	Number Of Calls	Dataflow Execution	Max Parallel Executions	Dataflow Acceleration	Total Time (ms)	Minimum Time (ms)	Average Time (ms)	Maximum Time (m
xilinx_u50_gen3x16_xdma_201920_3-0	KA_1	KA	1:1:1	1:1:1	1	No	1	1.000000x	0.024	0.024	0.024	0.0
xilinx_u50_gen3x16_xdma_201920_3-0	KB_1	KB	1:1:1	1:1:1	1	No	1	1.000000x	0.017	0.017	0.017	0.0
xilinx_u50_gen3x16_xdma_201920_3-0	KCalc_1	KCalc	1:1:1	1:1:1	1	No	1	1.000000x	0.019	0.019	0.019	0.0
xilinx_u50_gen3x16_xdma_201920_3-0	KVConstAdd_1	KVConstAdd	1:1:1	1:1:1	1	No	1	1.000000x	0.015	0.015	0.015	0.0
xilinx_u50_gen3x16_xdma_201920_3-0	KpB_1	KpB	1:1:1	1:1:1	1	No	1	1.000000x	0.046	0.046	0.046	0.0

Data Transfer: Host to Global Memory

Context:Number of Devices	Transfer Type	Number Of Buffer Transfers	Transfer Rate (MB/s)	Average Bandwidth Utilization (%)	Average Buffer Size (KB)	Total Time (ms)	Average Time (ms)
context0:1	READ	1	N/A	N/A	4.096	N/A	N/A
context0:1	WRITE	3	N/A	N/A	13.653	N/A	N/A

Data Transfer: Kernels to Global Memory

Device	Compute Unit/ Port Name	Kernel Arguments	Memory Resources	Transfer Type	Number Of Transfers	Transfer Rate (MB/s)	Average Bandwidth Utilization (%)	Average Size (KB)	Average Latency (ns)
xilinx_u50_gen3x16_xdma_201920_3-0	KA_1/m_axi_gmem	A|R	HBM[0]	READ	4096	1200.000	10.417	0.004	3.333
xilinx_u50_gen3x16_xdma_201920_3-0	KA_1/m_axi_gmem	A|R	HBM[0]	WRITE	64	1129.410	9.804	0.064	56.667
xilinx_u50_gen3x16_xdma_201920_3-0	KB_1/m_axi_gmem	A|R	HBM[0]	READ	192	1200.000	10.417	0.064	805.694
xilinx_u50_gen3x16_xdma_201920_3-0	KB_1/m_axi_gmem	A|R	HBM[0]	WRITE	64	1129.410	9.804	0.064	56.667
xilinx_u50_gen3x16_xdma_201920_3-0	KCalc_1/m_axi_gmem	A|B|R	HBM[0]	READ	2048	1200.000	10.417	0.004	3.333
xilinx_u50_gen3x16_xdma_201920_3-0	KCalc_1/m_axi_gmem	A|B|R	HBM[0]	WRITE	64	1129.410	9.804	0.064	56.667
xilinx_u50_gen3x16_xdma_201920_3-0	KVConstAdd_1/m_axi_gmem	V	HBM[0]	READ	256	1200.000	10.417	0.064	814.271
xilinx_u50_gen3x16_xdma_201920_3-0	KVConstAdd_1/m_axi_gmem	V	HBM[0]	WRITE	256	1129.410	9.804	0.064	56.667
xilinx_u50_gen3x16_xdma_201920_3-0	KpB_1/m_axi_gmem	A|B|R	HBM[0]	READ	6144	1200.000	10.417	0.004	3.333
xilinx_u50_gen3x16_xdma_201920_3-0	KpB_1/m_axi_gmem	A|B|R	HBM[0]	WRITE	192	1129.410	9.804	0.064	56.667

OpenCL API Calls

API Name	Number Of Calls	Total Time (ms)	Minimum Time (ms)	Average Time (ms)	Maximum Time (ms)
clFinish	1	84753.500	84753.500	84753.500	84753.500
clCreateProgramWithBinary	1	41067.300	41067.300	41067.300	41067.300
clReleaseProgram	1	11262.600	11262.600	11262.600	11262.600
clReleaseContext	1	8251.090	8251.090	8251.090	8251.090
clEnqueueTask	5	275.018	0.838	55.004	167.415
clCreateBuffer	7	9.108	0.164	1.301	6.668
clReleaseKernel	5	2.162	0.008	0.432	1.438
clCreateKernel	5	1.947	0.376	0.389	0.428
clEnqueueMigrateMemObjects	7	0.432	0.010	0.062	0.267
clGetEventProfilingInfo	26	0.190	0.006	0.007	0.030
clReleaseEvent	13	0.111	0.006	0.009	0.015
clSetKernelArg	12	0.098	0.006	0.008	0.019
clGetDeviceInfo	2	0.070	0.003	0.035	0.067
clReleaseMemObject	7	0.046	0.006	0.007	0.008
clEnqueueBarrierWithWaitList	1	0.031	0.031	0.031	0.031
clGetPlatformIDs	2	0.022	0.003	0.011	0.019
clEnqueueWriteBuffer	1	0.017	0.017	0.017	0.017
clCreateContext	1	0.015	0.015	0.015	0.015
clReleaseCommandQueue	1	0.011	0.011	0.011	0.011
clReleaseDevice	1	0.010	0.010	0.010	0.010
clCreateCommandQueue	1	0.009	0.009	0.009	0.009
clBuildProgram	1	0.008	0.008	0.008	0.008
clGetDeviceIDs	2	0.006	0.003	0.003	0.004
clGetPlatformInfo	2	0.006	0.002	0.003	0.003

Compute Units: Running Time and Stalls

Compute Unit	Running Time (us)	Intra-Kernel Stream Stalls (%)	External Memory Stalls (%)	External Stream Stalls (%)
KpB_1	46.239	0.0	-275315.357	0.0
KA_1	23.545	0.0	100.000	0.0
KB_1	17.210	0.0	10.519	0.0
KVConstAdd_1	14.636	0.0	100.000	0.0
KCalc_1	19.040	0.0	100.000	0.0

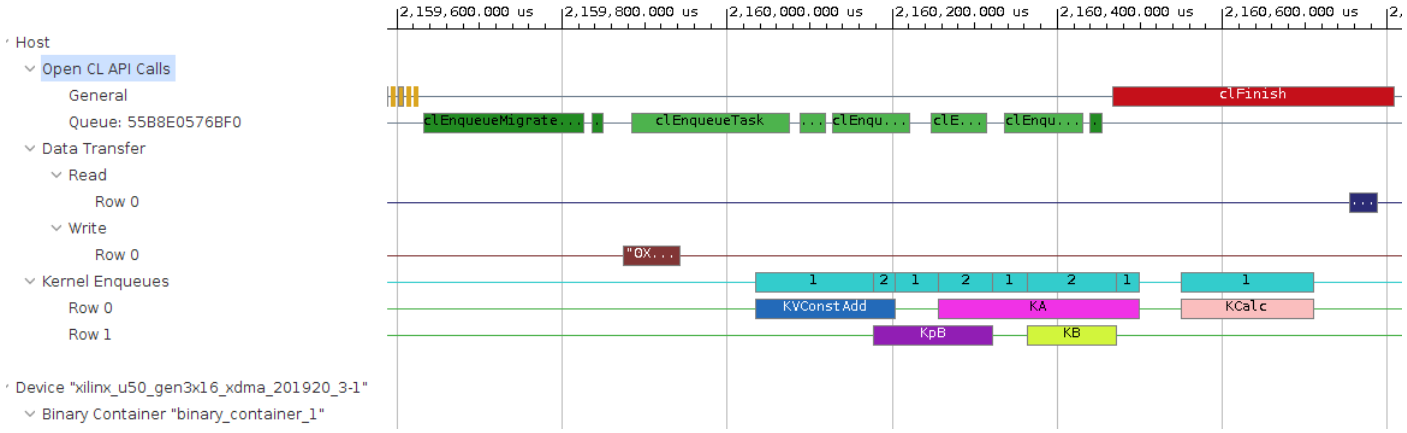
Functions: Running Time and Stalls

No Data. Some tables are not supporte

Compute Units: Port Data Transfer

Compute Unit	Port	Write Time (us)	Outstanding Write (%)	Read Time (us)	Outstanding Read (%)
KpB_1	m_axi_gmem	0.0	0.0	0.0	0.0
KA_1	m_axi_gmem	0.0	0.0	0.0	0.0
KB_1	m_axi_gmem	0.0	0.0	0.0	0.0
KVConstAdd_1	m_axi_gmem	0.0	0.0	0.0	0.0
KCalc_1	m_axi_gmem	0.0	0.0	0.0	0.0

Application Timeline



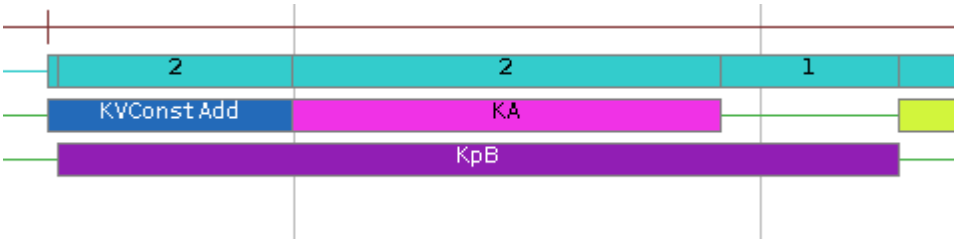
Compare to Opt 2

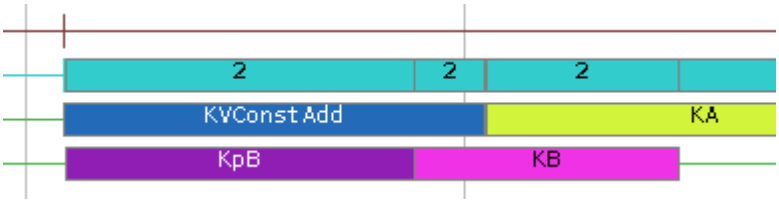
OPT3 對 KpB 做了 data burst 的優化，在 kernel 裡讓 input output 多一層 buffer，可能是這樣才能得到 data burst 的好處

```
#pragma HLS dataflow

#pragma HLS INTERFACE m_axi port=A  offset=slave bundle=gmem max_read_burst_length=256 max_write_burst_length=256
#pragma HLS INTERFACE m_axi port=B  offset=slave bundle=gmem max_read_burst_length=256 max_write_burst_length=256
#pragma HLS INTERFACE m_axi port=R  offset=slave bundle=gmem max_read_burst_length=256 max_write_burst_length=256

int A_tmp[SIZE_BUF_KpB], B_tmp[SIZE_BUF_KpB], R_tmp[SIZE_BUF_KpB];

data_read(A,B,A_tmp,B_tmp);

for(int i=0; i < SIZE_BUF_KpB; i+=1) {
    TMP_RES[i] = A_tmp[i] + B_tmp[i];
}

for(int i=0; i < SIZE_BUF_KpB; i+=1) {
    R_tmp[i] = TMP_RES[i] % 3;
}

memcpy(R,R_tmp,SIZE_BUF_KpB * sizeof (int));
```

```
#ifndef USE_BURST_TRANSFER_DATAFLOW
#else
void data_read (int* A, int* B, int* A_tmp, int* B_tmp){
    memcpy(A_tmp,A,SIZE_BUF_KpB * sizeof (int));
    memcpy(B_tmp,B,SIZE_BUF_KpB * sizeof (int));
}
#endif
```

比較 kpb 的執行時間，在硬體模擬下有大幅降低(但在硬體上執行時不是很明顯)





Opt 4 Array Partition

Profile Summary

Top Data Transfer: Kernels to Global Memory

No Data. Please use 'v++ -l --profile_kernel' to monitor and report kernel data transfers and execution.

Top Kernel Execution

Kernel Instance Address	Kernel	Context ID	Command Queue ID	Device	Start Time (ms)	Duration (ms)	Global Work Size	Local Work Size
0x56440a3189c0	KpB	0	0	xilinx_u50_gen3x16_xdma_201920_3-1	217.556	0.235	1:1:1	1:1:1
0x56440a3340a0	KVConstAdd	0	0	xilinx_u50_gen3x16_xdma_201920_3-1	217.432	0.218	1:1:1	1:1:1
0x56440a319370	KCalc	0	0	xilinx_u50_gen3x16_xdma_201920_3-1	217.954	0.155	1:1:1	1:1:1
0x56440a318a90	KA	0	0	xilinx_u50_gen3x16_xdma_201920_3-1	217.707	0.152	1:1:1	1:1:1
0x56440a319190	KB	0	0	xilinx_u50_gen3x16_xdma_201920_3-1	217.825	0.103	1:1:1	1:1:1

Top Memory Writes: Host to Global Memory

Buffer Address	Context ID	Command Queue ID	Start Time (ms)	Duration (ms)	Buffer Size (KB)	Writing Rate (MB/s)
0x0	0	0	217.227	0.092	40.960	446.046

Top Memory Reads: Host to Global Memory

Buffer Address	Context ID	Command Queue ID	Start Time (ms)	Duration (ms)	Buffer Size (KB)	Reading Rate (MB/s)
0xf000	0	0	218.153	0.027	4.096	150.004

Kernel Execution

Kernel	Number Of Enqueues	Total Time (ms)	Minimum Time (ms)	Average Time (ms)	Maximum Time (ms)
KA	1	0.152	0.152	0.152	0.152
KB	1	0.103	0.103	0.103	0.103
KCalc	1	0.155	0.155	0.155	0.155
KVConstAdd	1	0.218	0.218	0.218	0.218
KpB	1	0.235	0.235	0.235	0.235

Compute Unit Utilization

No Data. Please use 'v++ -l --p

Compute Units: Stall Information

No Data. Please use 'v++ ·

Data Transfer: Host to Global Memory

Context:Number of Devices	Transfer Type	Number Of Buffer Transfers	Transfer Rate (MB/s)	Average Bandwidth Utilization (%)	Average Buffer Size (KB)	Total Time (ms)	Average Time (ms)
context0:1	READ	1	150.004	1.563	4.096	0.027	0.027
context0:1	WRITE	1	446.046	4.646	40.960	0.092	0.092

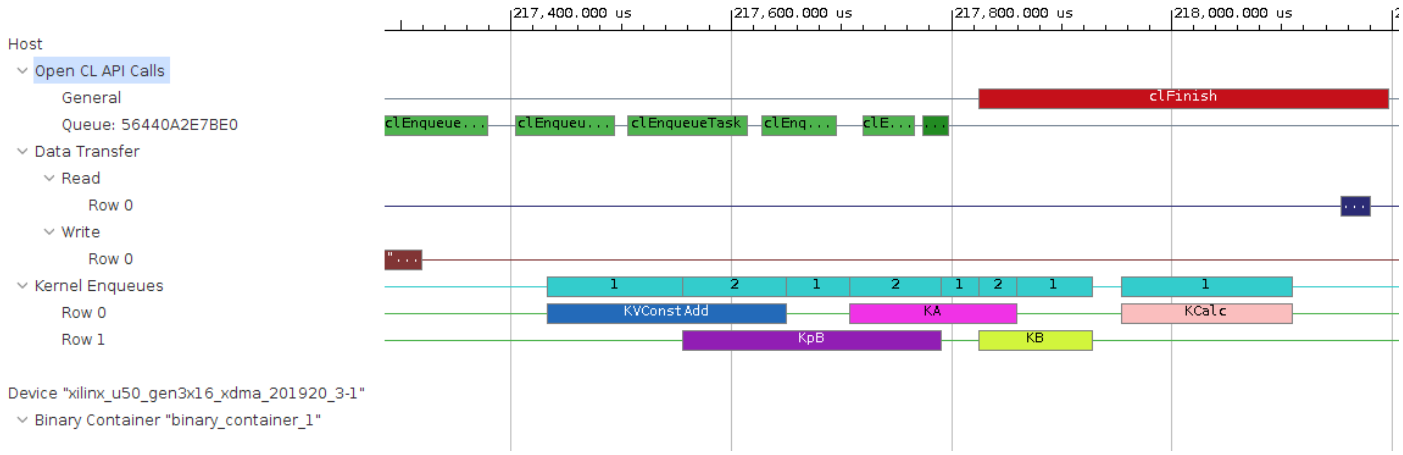
Data Transfer: Kernels to Global Memory

No Data. Please use 'v++ -l --profile_kernel' to monitor and report kernel data t

OpenCL API Calls

API Name	Number Of Calls	Total Time (ms)	Minimum Time (ms)	Average Time (ms)	Maximum Time (ms)
clGetDeviceInfo	4	160.401	0.006	40.100	80.431
clReleaseContext	1	53.197	53.197	53.197	53.197
clCreateProgramWithBinary	1	31.355	31.355	31.355	31.355
clCreateContext	1	5.579	5.579	5.579	5.579
clReleaseProgram	1	2.681	2.681	2.681	2.681
clCreateKernel	5	2.011	0.352	0.402	0.576
clEnqueueTask	5	0.423	0.046	0.085	0.111
clFinish	1	0.372	0.372	0.372	0.372
clEnqueueMigrateMemObjects	3	0.236	0.015	0.079	0.199
clReleaseKernel	5	0.153	0.008	0.031	0.048
clGetEventProfilingInfo	16	0.146	0.006	0.009	0.043
clCreateBuffer	7	0.117	0.008	0.017	0.035
clReleaseEvent	8	0.078	0.006	0.010	0.018
clReleaseMemObject	7	0.045	0.006	0.006	0.008
clGetPlatformIDs	2	0.044	0.008	0.022	0.036
clSetKernelArg	12	0.037	0.003	0.003	0.005
clEnqueueBarrierWithWaitList	1	0.017	0.017	0.017	0.017
clGetDeviceIDs	2	0.015	0.007	0.008	0.009
clReleaseCommandQueue	1	0.014	0.014	0.014	0.014
clCreateCommandQueue	1	0.014	0.014	0.014	0.014
clGetPlatformInfo	2	0.013	0.006	0.007	0.007
clBuildProgram	1	0.013	0.013	0.013	0.013
clReleaseDevice	1	0.010	0.010	0.010	0.010

Application Timeline



Compare to Opt 2

Opt 4 對 KA 做了 array partition 的優化，倍率為 4

```
#ifndef USE_BURST_TRANSFER_ARRAY_PARTITION
#else
    int TMP_A[SIZE_DataIn_1];

    #pragma HLS array_partition variable=TMP_A cyclic factor=4 dim=1

    for (int i=0; i<SIZE_DataIn_1; i++) TMP_A[i]= A[i];
#endif

    loop_st_1:
    for (int i=0; i<SIZE_BUF_KA; i++) {
#ifndef USE_BURST_TRANSFER_ARRAY_PARTITION
        R[i] = A[2*i]*3 + A[2*i+1]*5 + A[2*i+2]*7 + A[2*i+3]*9;
#else
        R[i] = TMP_A[2*i]*3 + TMP_A[2*i+1]*5 + TMP_A[2*i+2]*7 + TMP_A[2*i+3]*9;
#endif
```

可以看到 KA 的執行時間有降低