ACA HLS Final Project
DNA Sequence Analysis Acceleration

Team12
陳正康
蔡德育
王則勛

https://github.com/soyccan/hls_project

Outline

- Problem Statement
- Architecture
- Analysis and Optimization
  - System
  - Kernel
- Implementation Issues
- Conclusion

Problem Statement

- DNA sequence analysis includes an operation of string matching
- Mapping many short reads onto one long reference genome (find common substrings)

Problem Statement

- Burrows-Wheeler Aligner
- Step 1: Build FM-index table O and C of the reference genome

Position	0	1	2	3	4	5	6
Ref. sequence (X)	C	C	T	G	A	G	$

Position	0	1	2
Short read (W)	C	G	A

a	A	C	G	T
$C(a)$	1	2	4	6

(a) Reference sequence X, short-read W and $C(a)$ of X

Position	0	1	2	3	4	5	6
0	C	C	T	G	A	G	$
1	C	T	G	A	G	$	C
2	T	G	A	G	$	C	C
3	G	A	G	$	C	C	T
4	A	G	$	C	C	T	G
5	G	$	C	C	T	G	A
6	$	C	C	T	G	A	G

(b) Rotation of X

BWT string (B)

Occurrence array O(. , .)

SA	Position	0	1	2	3	4	5	6	$	A	C	G	T
0	6	$	C	C	T	G	A	G	0	0	0	1	0
1	4	A	G	$	C	C	T	G	0	0	0	2	0
2	0	C	C	T	G	A	G	$	1	0	0	2	0
3	1	C	T	G	A	G	$	C	1	0	1	2	0
4	5	G	$	C	C	T	G	A	1	1	1	2	0
5	3	G	A	G	$	C	C	T	1	1	1	2	1
6	2	T	G	A	G	$	C	C	1	1	2	2	1

(c) Sorting result

Problem Statement

- Burrows-Wheeler Aligner
- Step 2: With index O and C, perform the alignment process of a read W
- Recurrence is implemented by ring buffer

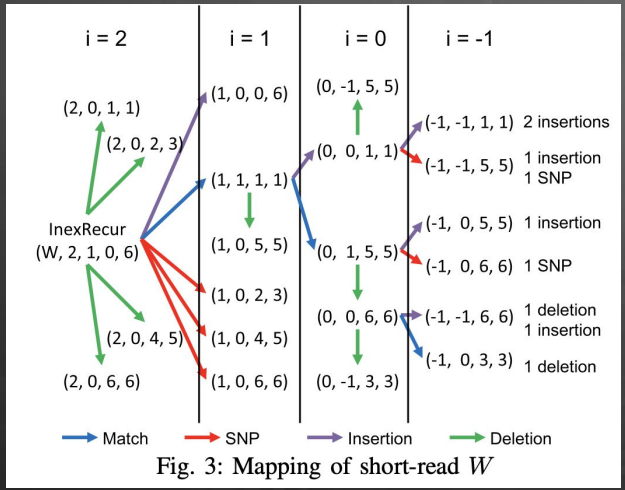


Fig. 3: Mapping of short-read W

Precalculation:

Calculate BWT string B for reference string X
Calculate array $C(\cdot)$ and $O(\cdot,\cdot)$ from B
Calculate BWT string B' for the reverse reference
Calculate array $O'(\cdot,\cdot)$ from B'

Procedures:

INEXRECUR(W,i,z,k,l)
  if $z < D(i)$ **then**
    return $\emptyset$
  if $i < 0$ **then**
    return $\{[k,l]\}$
  $I \leftarrow \emptyset$
  $I \leftarrow I \cup$ INEXRECUR$(W,i-1,z-1,k,l)$
  for each $b \in \{A,C,G,T\}$ **do**
    $k \leftarrow C(b)+O(b,k-1)+1$
    $l \leftarrow C(b)+O(b,l)$
    if $k \le l$ **then**
      $I \leftarrow I \cup$ INEXRECUR$(W,i,z-1,k,l)$
      if $b = W[i]$ **then**
        $I \leftarrow I \cup$ INEXRECUR$(W,i-1,z,k,l)$
      else
        $I \leftarrow I \cup$ INEXRECUR$(W,i-1,z-1,k,l)$
  return I

Preprocessing

- We implemented an **FM-index generator** using openBWT in software
    - BWT: Burrows-Wheeler Transform
- Our kernel will use this index as input

```
ref_size 100
endmark 85
sa bwt occ[0:3]
99 A 1 0 0 0
19 G 1 0 1 0
67 C 1 1 1 0
20 C 1 2 1 0
87 A 2 2 1 0
76 G 2 2 2 0
29 T 2 2 2 1
45 T 2 2 2 2
68 G 2 2 3 2
84 A 3 2 3 2
53 C 3 3 3 2
35 C 3 4 3 2
3 C 3 5 3 2
21 G 3 5 4 2
88 A 4 5 4 2
```

Test Dataset

- Reference genome length = 100
  - Index table size (O) = 100 x 4
- #Reads = 10
- Read length = 10

Architecture

ring buffer (global memory)

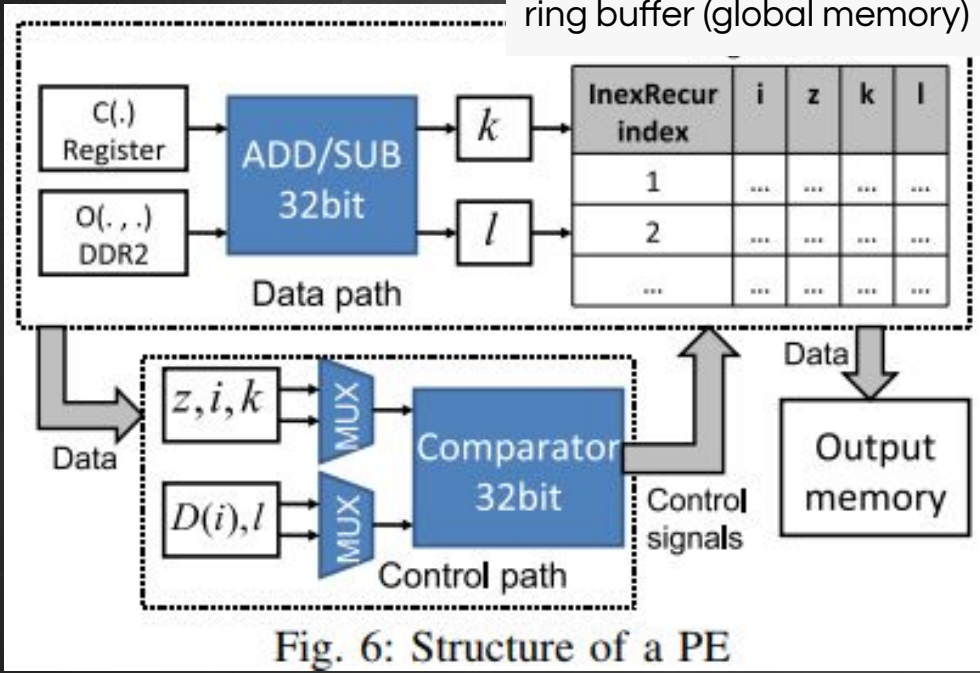
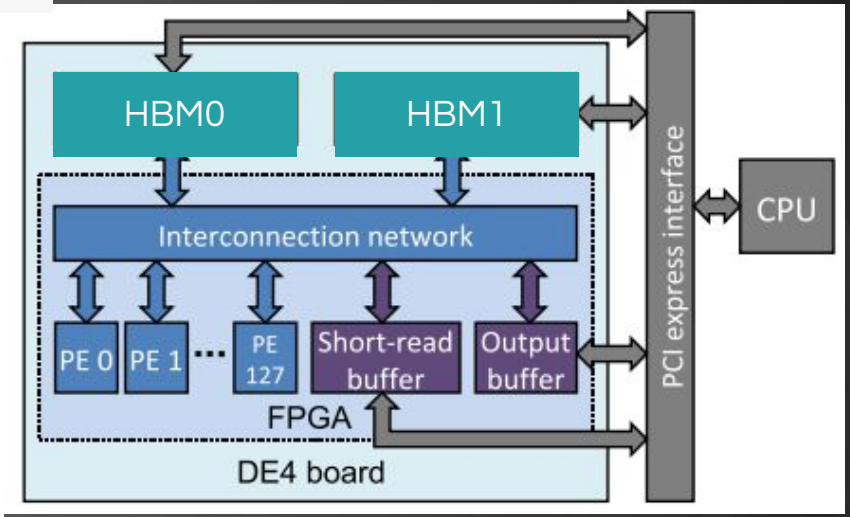
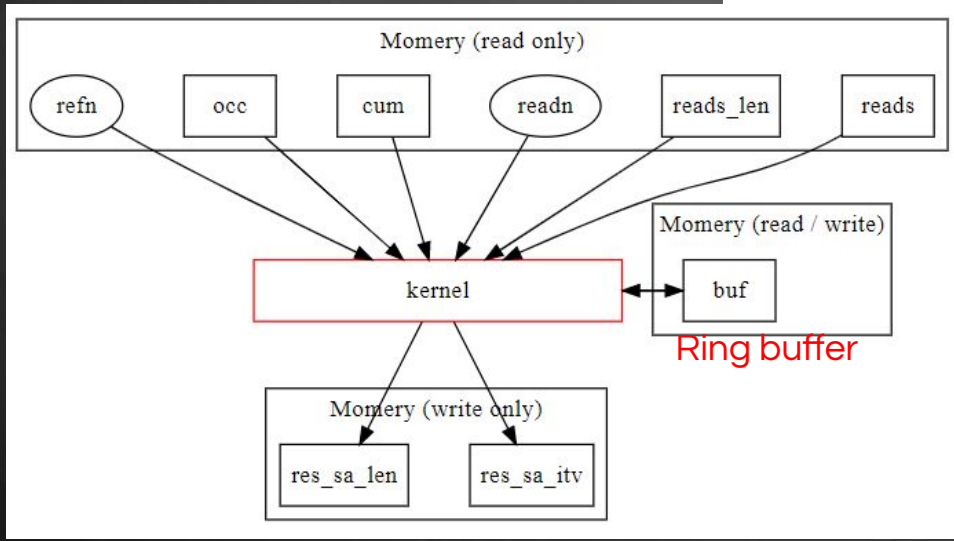


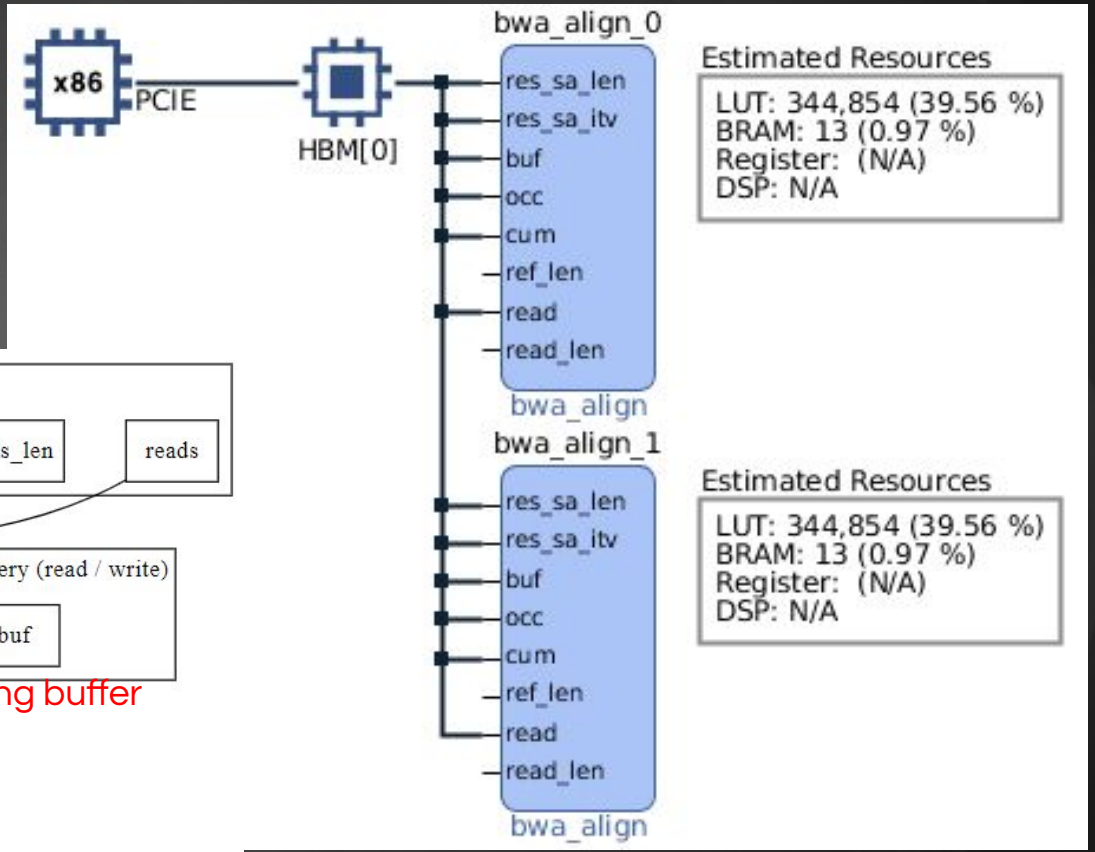
Fig. 6: Structure of a PE



System Diagram

- Platform: Xilinx Alveo U50
- Tool: Vitis with OpenCL

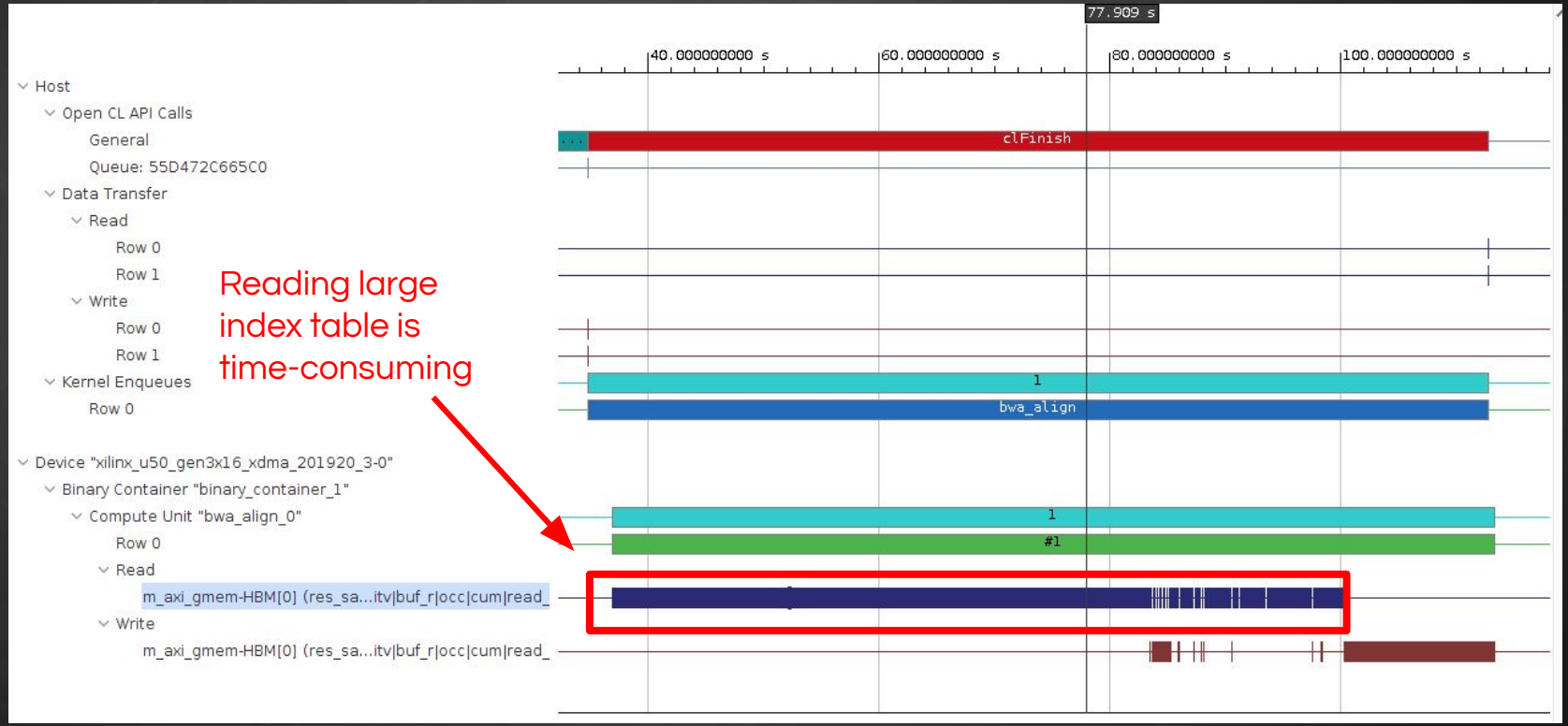




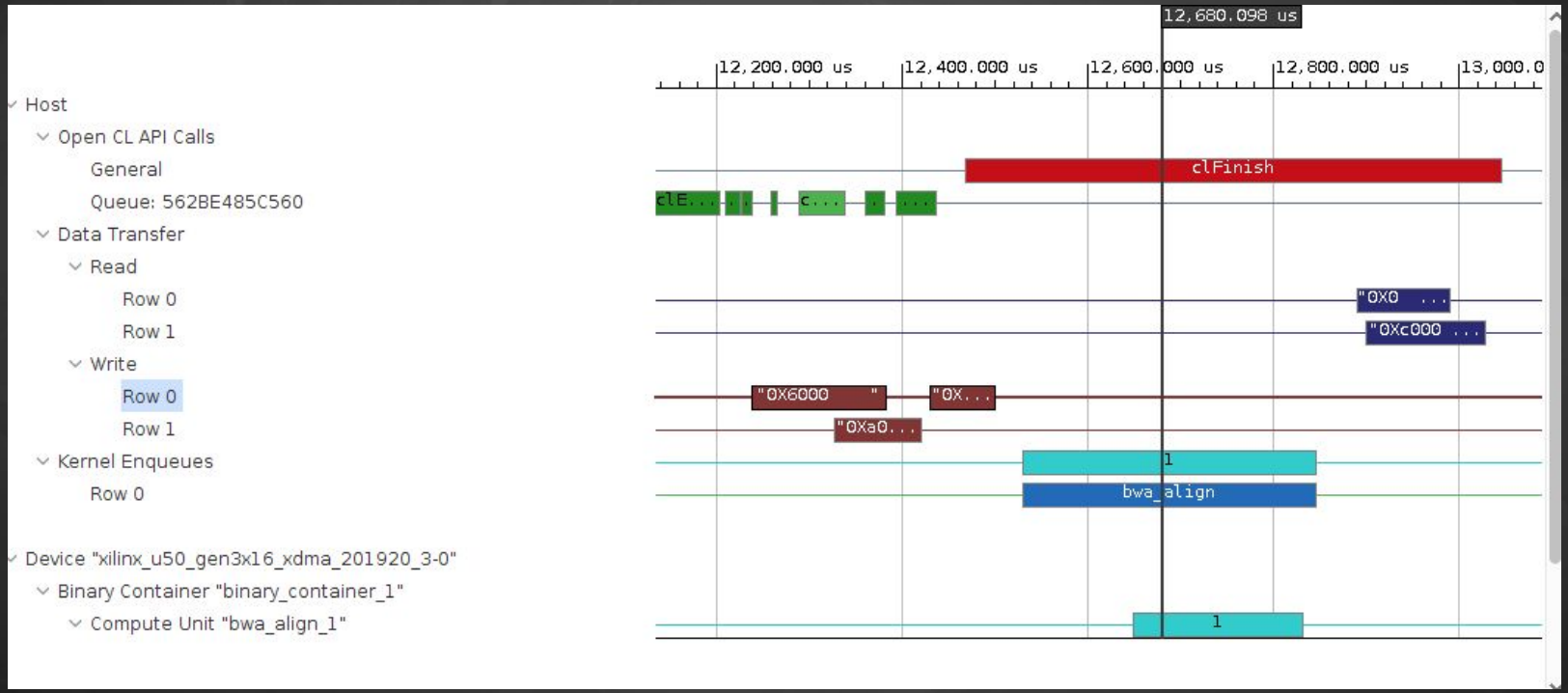
Interface

Name	Type	Size	Offset	Host Offset	Host Size	Port	Port Mode	Port Range	Port Data Width	Port Base
bwa_align										
res_sa_len	int*	0x8	0x10	0x0	0x8	M_AXI_GMEM	master	0xFFFF_FFFF	32	0x0
res_sa_itv	int*	0x8	0x1C	0x0	0x8	M_AXI_GMEM	master	0xFFFF_FFFF	32	0x0
buf	int*	0x8	0x28	0x0	0x8	M_AXI_GMEM	master	0xFFFF_FFFF	32	0x0
occ	int const *	0x8	0x34	0x0	0x8	M_AXI_GMEM	master	0xFFFF_FFFF	32	0x0
cum	int const *	0x8	0x40	0x0	0x8	M_AXI_GMEM	master	0xFFFF_FFFF	32	0x0
ref_len	int	0x4	0x4C	0x0	0x4	S_AXI_CONTROL	slave	0x1000	32	0x0
read	char const *	0x8	0x54	0x0	0x8	M_AXI_GMEM	master	0xFFFF_FFFF	32	0x0
read_len	int	0x4	0x60	0x0	0x4	S_AXI_CONTROL	slave	0x1000	32	0x0

Single Kernel, Single Read

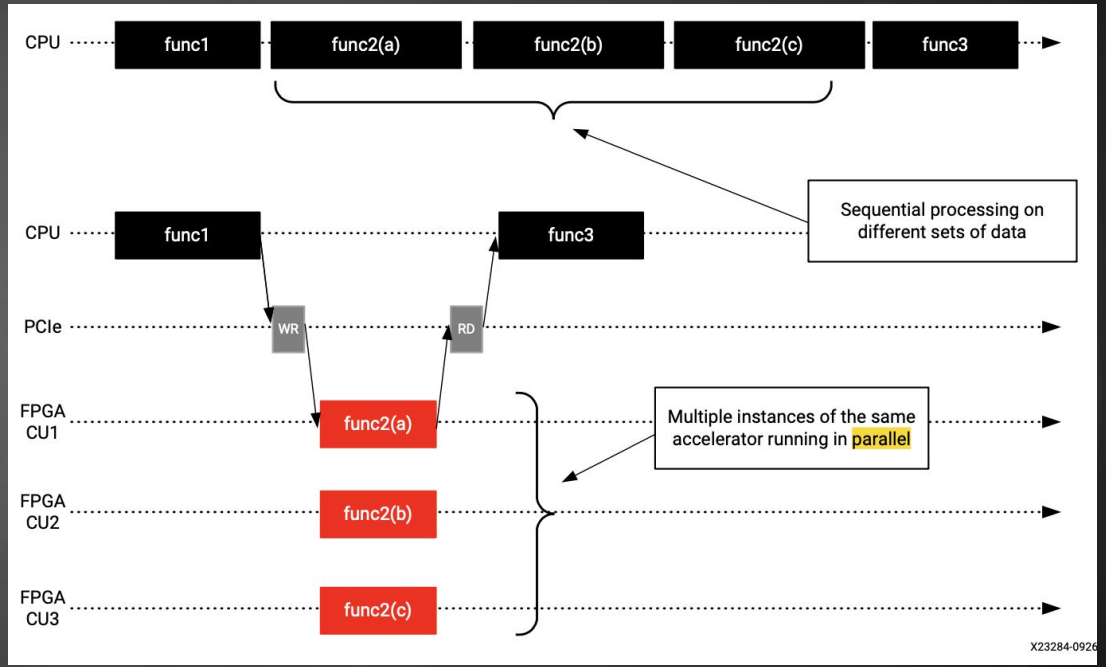


Single Kernel, Single Read



Multiple Compute Units

- Improve throughput by dispatching reads to separate CU
- Correctly specifiy dependencies, then XRT will schedule them



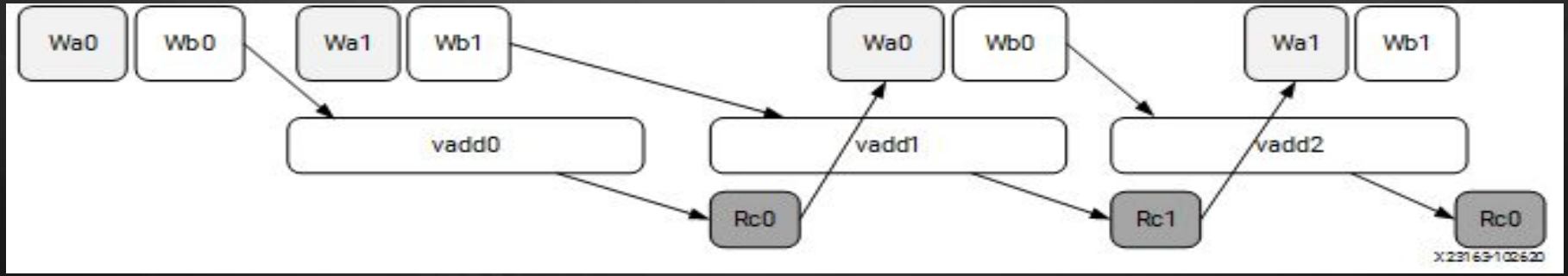
Multiple Compute Units

Compute Units: Running Time and Stalls

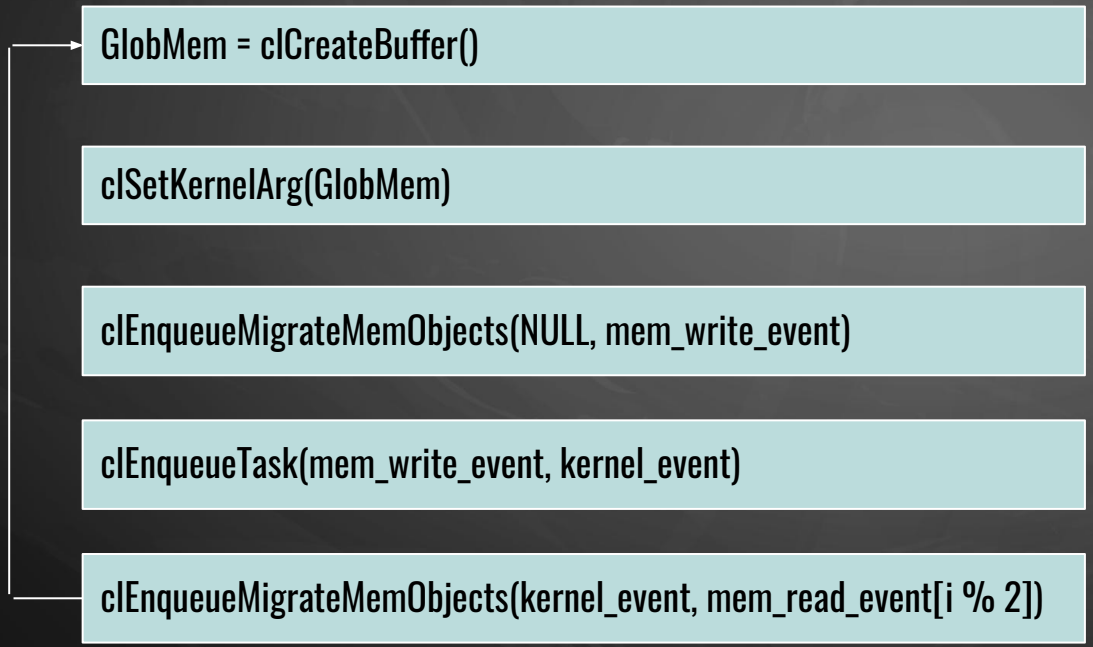
Compute Unit	Running Time (us)	Intra-Kernel Stream Stalls (%)	External Memory Stalls (%)	External Stream Stalls (%)
bwa_align_0	85.014	0.0	-45875.924	0.0
bwa_align_1	123.736	0.0	0.0	0.0

Hiding Data Transfer

- Kernel consumes one read per run
- Make use of out-of-order execution to hide data transfer inside kernel execution
- `clEnqueueMigrateMemObjects(), clEnqueueTask()`



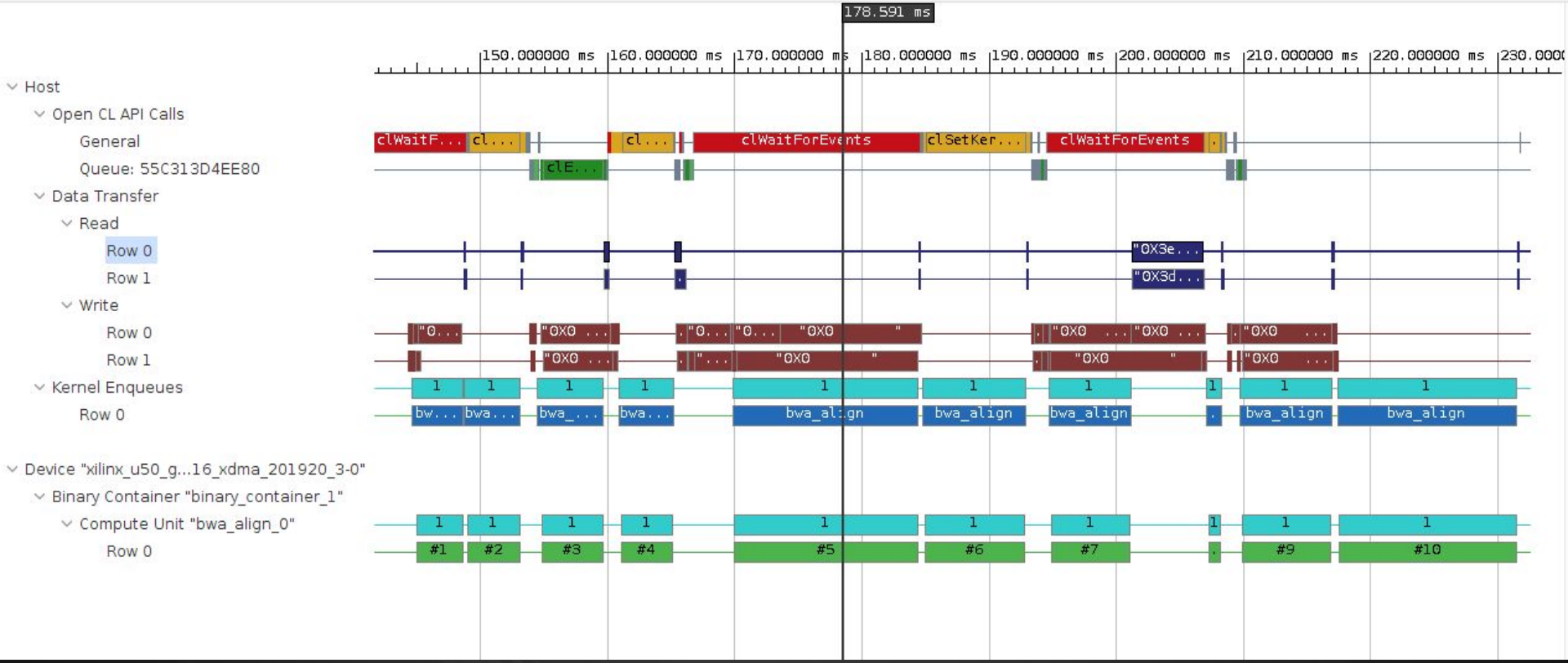
Hiding Data Transfer



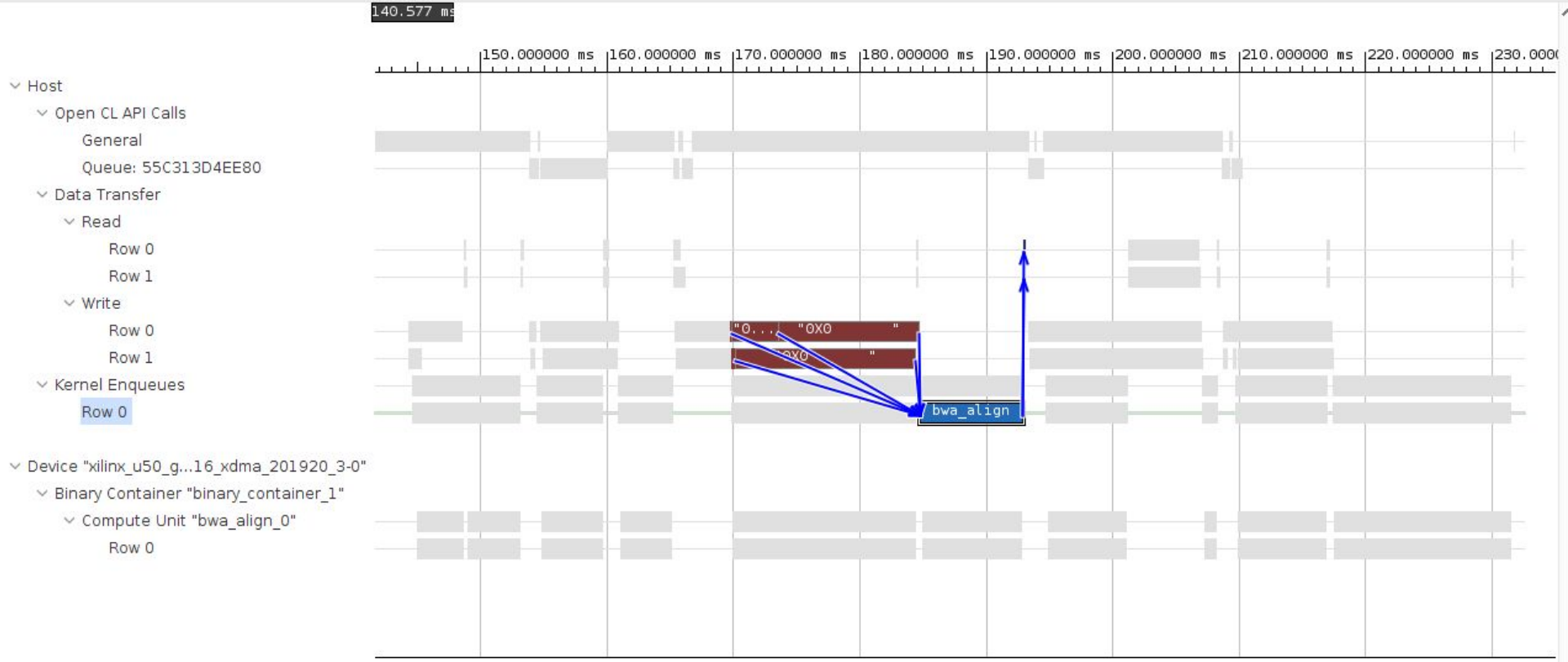
Problem: Stuck at memory write so that kernel won't start without any error message

Solution: Make sure host code specifies correct buffer size

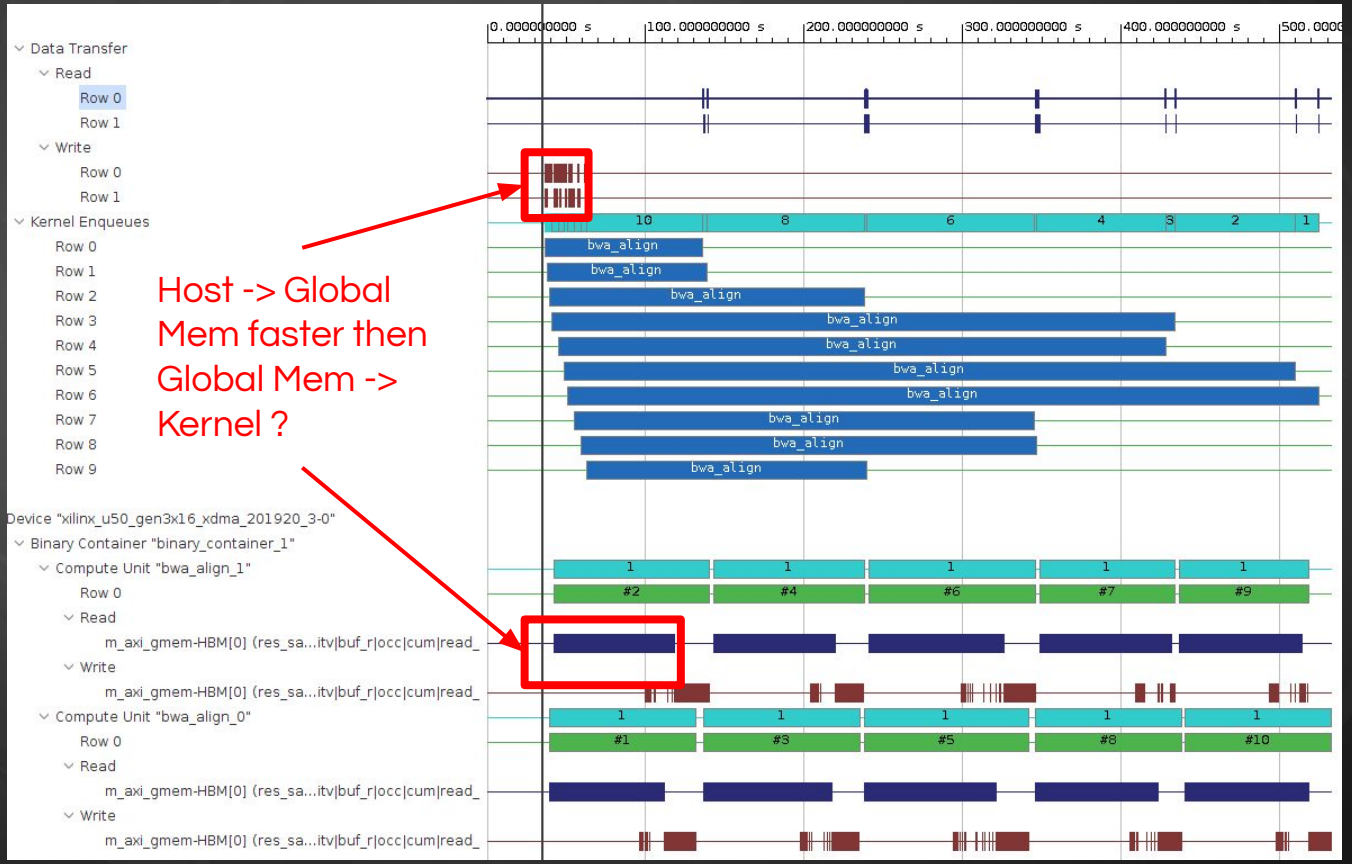
Hiding Data Transfer - Timeline (SW)



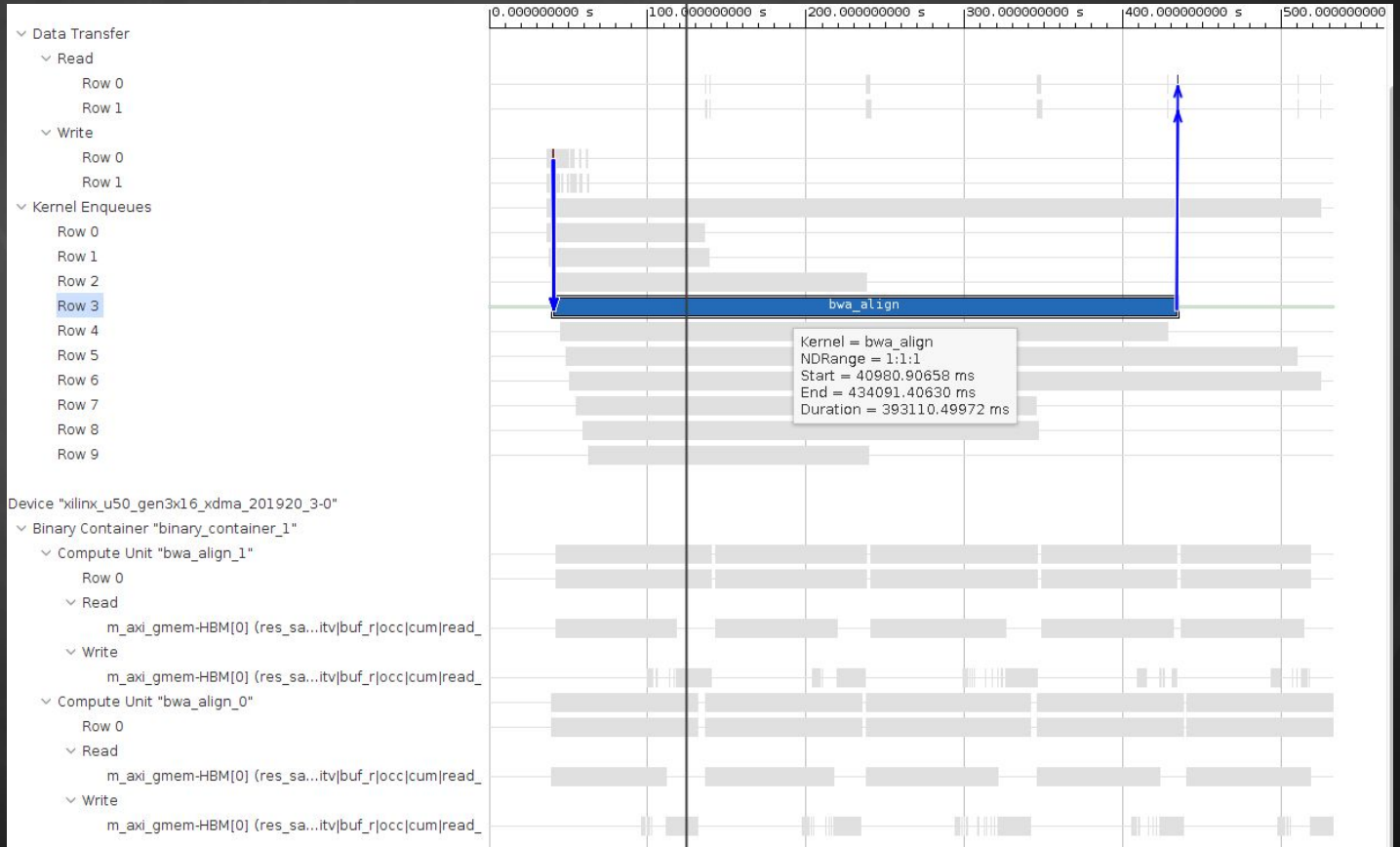
Hiding Data Transfer - Timeline (SW)



Hiding Data Transfer - Timeline (HW-emulation)



Hiding Data Transfer - Timeline (HW-emulation)



Bandwidth Usage (HW-emulation)

- Why Host -> Glabal_Mem significantly faster than Global_Mem -> Kernel?
  - # transfer of the ring buffer is large and irregular, and does not pass host
  - Maybe we should increase the data burst size
- Why the time of Host -> Global_Mem cannot be measured?

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	20	N/A	N/A	1.026	N/A	N/A
context0:1	WRITE	50	N/A	N/A	1.236	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
xilinx_u50_gen3x16_xdma_201920_3-0	bwa_align_0/m_axi_gmem	res_sa_len|res_sa_itv|buf_r|occ|cum|read_r	HBM[0]	READ	5744	4800.000	41.667	0.016	3.333
xilinx_u50_gen3x16_xdma_201920_3-0	bwa_align_1/m_axi_gmem	res_sa_len|res_sa_itv|buf_r|occ|cum|read_r	HBM[0]	READ	5724	4800.000	41.667	0.016	3.333
xilinx_u50_gen3x16_xdma_201920_3-0	bwa_align_0/m_axi_gmem	res_sa_len|res_sa_itv|buf_r|occ|cum|read_r	HBM[0]	WRITE	1435	3211.190	27.875	0.016	6.667
xilinx_u50_gen3x16_xdma_201920_3-0	bwa_align_1/m_axi_gmem	res_sa_len|res_sa_itv|buf_r|occ|cum|read_r	HBM[0]	WRITE	1568	3052.070	26.494	0.016	6.667

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	bwa_align_1	7292	16.000	0.391	0.117	0.025	0.092	4273.700
xilinx_u50_gen3x16_xdma_201920_3-0	bwa_align_0	7179	16.000	0.391	0.115	0.023	0.092	4368.010

Top Kernel Execution

Kernel Instance Address	Kernel	Context ID	Command Queue ID	Device	Start Time (ms)	Duration (ms)	Global Work Size	Local Work Size
0x55c640d85cc0	bwa_align	0	0	xilinx_u50_gen3x16_xdma_201920_3-0	0.262	0.090	1:1:1	1:1:1
0x55c640d85cc0	bwa_align	0	0	xilinx_u50_gen3x16_xdma_201920_3-0	0.256	0.081	1:1:1	1:1:1
0x55c640d85cc0	bwa_align	0	0	xilinx_u50_gen3x16_xdma_201920_3-0	0.031	0.076	1:1:1	1:1:1
0x55c640d85cc0	bwa_align	0	0	xilinx_u50_gen3x16_xdma_201920_3-0	0.109	0.075	1:1:1	1:1:1
0x55c640d85cc0	bwa_align	0	0	xilinx_u50_gen3x16_xdma_201920_3-0	0.103	0.075	1:1:1	1:1:1
0x55c640d85cc0	bwa_align	0	0	xilinx_u50_gen3x16_xdma_201920_3-0	0.179	0.074	1:1:1	1:1:1
0x55c640d85cc0	bwa_align	0	0	xilinx_u50_gen3x16_xdma_201920_3-0	0.186	0.074	1:1:1	1:1:1
0x55c640d85cc0	bwa_align	0	0	xilinx_u50_gen3x16_xdma_201920_3-0	0.353	0.074	1:1:1	1:1:1
0x55c640d85cc0	bwa_align	0	0	xilinx_u50_gen3x16_xdma_201920_3-0	0.338	0.073	1:1:1	1:1:1
0x55c640d85cc0	bwa_align	0	0	xilinx_u50_gen3x16_xdma_201920_3-0	0.028	0.073	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)
0x9000	0	0	33294.400	N/A	4.096	N/A
0xa000	0	0	33388.600	N/A	0.016	N/A
0x7000	0	0	33112.600	N/A	2.048	N/A
0x27000	0	0	233123.000	N/A	4.096	N/A
0x29000	0	0	233370.000	N/A	0.016	N/A
0x28000	0	0	233244.000	N/A	0.016	N/A
0x25000	0	0	233010.000	N/A	2.048	N/A
0x33000	0	0	312075.000	N/A	4.096	N/A
0x1b000	0	0	167018.000	N/A	4.096	N/A
0x1c000	0	0	167132.000	N/A	0.016	N/A

Kernel Execution (includes estimated device times)

Kernel	Number Of Enqueues	Total Time (ms)	Minimum Time (ms)	Average Time (ms)	Maximum Time (ms)
bwa_align	10	0.765	0.073	0.076	0.090

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 (ms)
xilinx_u50_gen3x16_xdma_201920_3-0	bwa_align_0	bwa_align	1:1:1	1:1:1	5	No	1	1.000000x	0.357	0.069	0.071	0.077
xilinx_u50_gen3x16_xdma_201920_3-0	bwa_align_1	bwa_align	1:1:1	1:1:1	5	No	1	1.000000x	0.369	0.070	0.074	0.086

Local Buffering

- Reduce memory access to large index table O by local buffering
  - O is Ref_Genome_Size x 4
- Index of access to O is **increasing**
  - k, l are increasing during runtime
  - Easy to buffer

```
INEXRECUR(W, i, z, k, l)
  if z < D(i) then
    return ∅
  if i < 0 then
    return {[k, l]}
  I ← ∅
  I ← I ∪ INEXRECUR(W, i − 1, z − 1, k, l)
  for each b ∈ {A, C, G, T} do
    k ← C(b) + O(b, k − 1) + 1
    l ← C(b) + O(b, l)
    if k ≤ l then
      I ← I ∪ INEXRECUR(W, i, z − 1, k, l)
      if b = W[i] then
        I ← I ∪ INEXRECUR(W, i − 1, z, k, l)
      else
        I ← I ∪ INEXRECUR(W, i − 1, z − 1, k, l)
  return I
```

Local Buffering

- Reduce R/W turnaround of ring buffer

```
INEXRECUR(W, i, z, k, l)
  if z < D(i) then
    return ∅
  if i < 0 then
    return {[k, l]}
  I ← ∅
  I ← I ∪ INEXRECUR(W, i − 1, z − 1, k, l)
  for each b ∈ {A, C, G, T} do
    k ← C(b) + O(b, k − 1) + 1
    l ← C(b) + O(b, l)
    if k ≤ l then
      I ← I ∪ INEXRECUR(W, i, z − 1, k, l)
      if b = W[i] then
        I ← I ∪ INEXRECUR(W, i − 1, z, k, l)
      else
        I ← I ∪ INEXRECUR(W, i − 1, z − 1, k, l)
  return I
```

Local Buffering

- How to determine local buffer size?
- Consider data burst size and available memory
  - Max data burst size = 256 bits

Specification	U50 Production[1]
Product SKU	A-U50-P00G-PQ-G
Total electrical card load[2]	75W
Thermal cooling solution	Passive
Weight	300g – 325g
Form factor	Half height, half length
Network interface	1x QSFP28 (100 GbE)
Network clock precision	IEEE 1588
PCIe interface[3, 4]	Gen3 x16, Gen4 x8, CCIX
HBM2 total capacity	8 GB
HBM2 bandwidth	316 GB/s[7]
Look-up tables (LUTs)	872K
Registers	1,743K
DSP slices	5,952
Max. Dist. RAM	24.6 Mb
36 Kb block RAM	1344 (47.3 Mb)
288 Kb UltraRAM	640 (180.0 Mb)
GTY transceivers	20
V_{CCINT} supported	V_{NOM} (0.85V)
Vitis™ Development Enviroment	Yes
Vitis platform	Gen3 x16 XDMA, Gen3 x4 XDMA[8]
Vivado Design Suite	Yes
Target workloads	Fintech, video, database, and computational storage

Local Buffering

```
void bwa_align(
    int* res_sa_len, int res_sa_itv[BUF_SIZE * 2],
    int buf[BUF_SIZE * 4],
    const int occ[BUF_SIZE * 4],
    const int cum[4],
    int ref_len,
    const char read[READ_MAX_LEN], int read_len)
{
  static int res_sa_itv_w[LOCAL_BUF_SIZE][2];
  static int buf_w[9][4];

  static int buf_r[4];
  static int occ_r[LOCAL_BUF_SIZE][4];
  static int cum_r[4];
  static char read_r[READ_MAX_LEN];
```

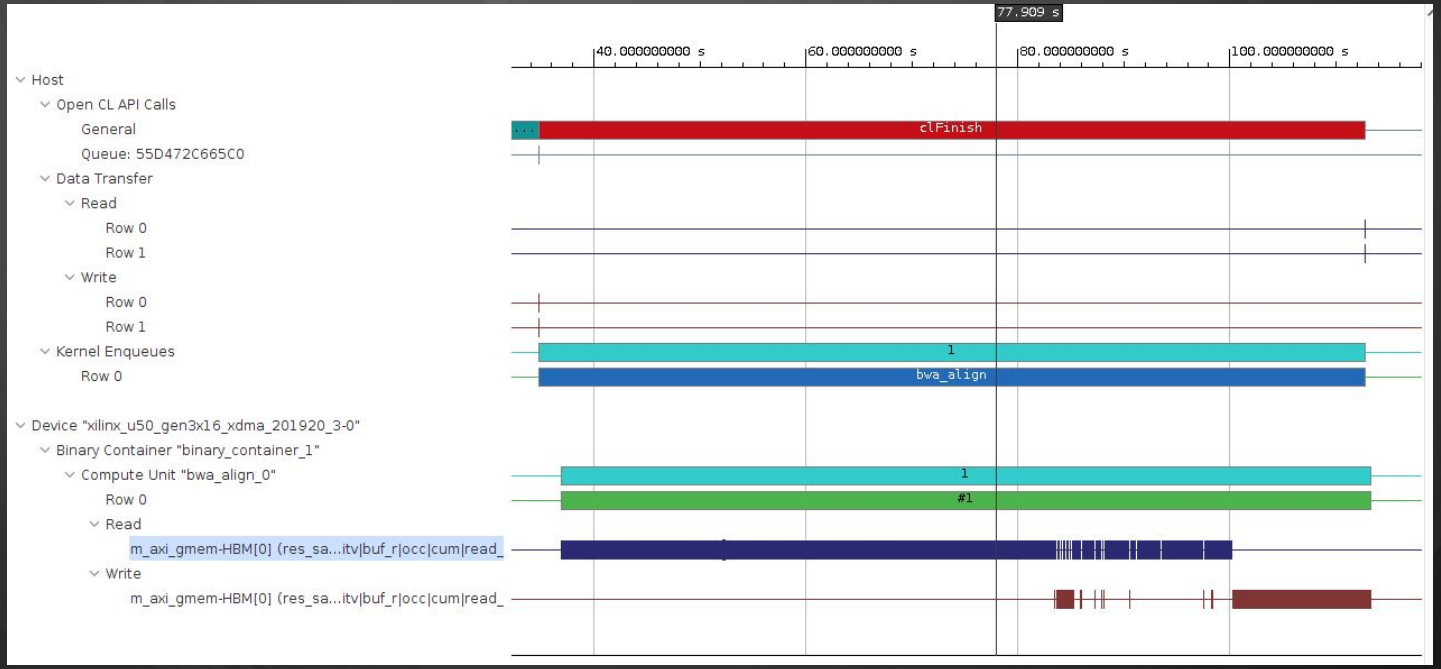
```
  //// read input ////
  for (int i=0; i<LOCAL_BUF_SIZE; i++) {
#pragma HLS PIPELINE II=1
    for (int j=0; j<4; j++) {
      occ_r[i][j] = occ[i * 4 + j];
    }
  }
  for (int i=0; i<4; i++) {
#pragma HLS PIPELINE II=1
    cum_r[i] = cum[i];
  }
  for (int i=0; i<MAX_READ_LEN; i++) {
#pragma HLS PIPELINE II=1
    read_r[i] = read[i];
  }

  // process...

  // write result
  for (int j=0; j<LOCAL_BUF_SIZE; j++) {
#pragma HLS PIPELINE II=1
    if (j < tail * 4) {
      res_sa_itv[j * 2] = res_sa_itv_w[j][0];
      res_sa_itv[j * 2 + 1] = res_sa_itv_w[j][1];
    }
  }
```

Local Buffering

- Local buffer size = 256 x 4 int



Local Buffering

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	2	N/A	N/A	1.026	N/A	N/A
context0:1	WRITE	5	N/A	N/A	1.236	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)
xilinx_u50_gen3x16_xdma_201920_3-0	bwa_align_0/m_axi_gmem	res_sa_len|res_sa_itv|buf_r|occ|cum|read_r	HBM[0]	READ	1177	4800.000	41.667	0.016
xilinx_u50_gen3x16_xdma_201920_3-0	bwa_align_0/m_axi_gmem	res_sa_len|res_sa_itv|buf_r|occ|cum|read_r	HBM[0]	WRITE	411	2676.800	23.236	0.016

Compute Units: Running Time and Stalls

Compute Unit	Running Time (us)	Intra-Kernel Stream Stalls (%)	External Memory Stalls (%)	External Stream Stalls (%)
bwa_align_0	85.014	0.0	-45875.924	0.0
bwa_align_1	123.736	0.0	0.0	0.0

Local Buffering

-
-

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	bwa_align_0	1588	16.000	0.391	0.025	0.007	0.019	3982.450
xilinx_u50_gen3x16_xdma_201920_3-0	bwa_align_1	0	0.0	0.0	0.0	0.0	0.0	0.0

Top Kernel Execution

Kernel Instance Address	Kernel	Context ID	Command Queue ID	Device	Start Time (ms)	Duration (ms)	Global Work Size	Local Work Size
0x55d472c89ac0	bwa_align	0	0	xilinx_u50_gen3x16_xdma_201920_3-0	0.027	0.088	1:1:1	1:1:1

Top Memory Writes: Host to Global Memory

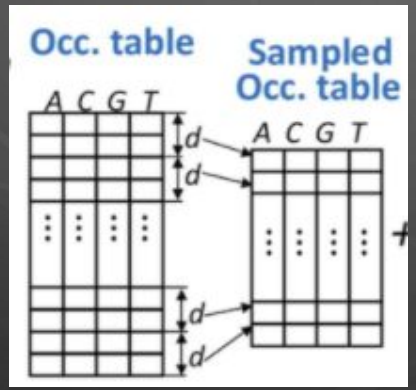
Buffer Address	Context ID	Command Queue ID	Start Time (ms)	Duration (ms)	Buffer Size (KB)	Writing Rate (MB/s)
0x3000	0	0	34787.800	N/A	4.096	N/A
0x1000	0	0	34787.600	N/A	2.048	N/A
0x4000	0	0	34791.300	N/A	0.016	N/A
0x0	0	0	34787.500	N/A	0.004	N/A
0x5000	0	0	34792.800	N/A	0.016	N/A

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)
0x0	0	0	112800.000	N/A	0.004	N/A
0x1000	0	0	112800.000	N/A	2.048	N/A

Reducing Index Table Size

- Index table are generated off-line
- To reduce index table size, accumulate an interval of the index table O, and recalculate on demand
- Require BWT string for reconstruction



Position	0	1	2	3	4	5	6
Ref. sequence (X)	C	C	T	G	A	G	$

Position	0	1	2
Short read (W)	C	G	A

a	A	C	G	T
$C(a)$	1	2	4	6

(a) Reference sequence X, short-read W and $C(a)$ of X

Position	0	1	2	3	4	5	6
0	C	C	T	G	A	G	$
1	C	T	G	A	G	$	C
2	T	G	A	G	$	C	C
3	G	A	G	$	C	C	T
4	A	G	$	C	C	T	G
5	G	$	C	C	T	G	A
6	$	C	C	T	G	A	G

(b) Rotation of X

BWT string (B)

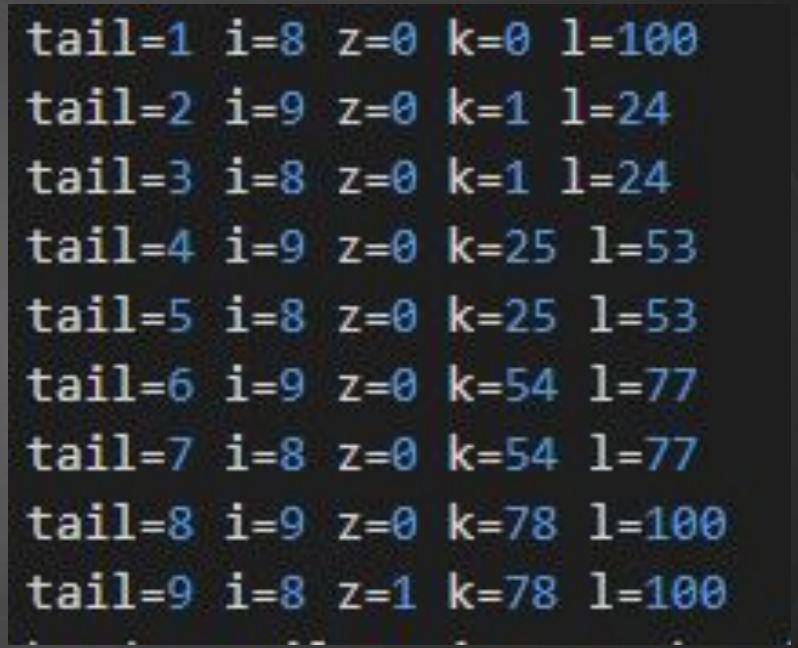
Occurrence array O(. , .)

SA	Position	0	1	2	3	4	5	6	$	A	C	G	T
0	6	$	C	C	T	G	A	G	0	0	0	1	0
1	4	A	G	$	C	C	T	G	0	0	0	2	0
2	0	C	C	T	G	A	G	$	1	0	0	2	0
3	1	C	T	G	A	G	$	C	1	0	1	2	0
4	5	G	$	C	C	T	G	A	1	1	1	2	0
5	3	G	A	G	$	C	C	T	1	1	1	2	1
6	2	T	G	A	G	$	C	C	1	1	2	2	1

(c) Sorting result

Streaming Interface from Host to Kernel?

- We thought streaming interface may not be helpful
- Since index of O (k and l) is increasing by variable step size
- Streaming may be beneficial when many kernels form a dataflow, but we have only one kernel



More Optimization for Memory Access

- Separate HBM banks
  - More bandwidth to utilize, may increase local buffer size
  - Note: Kernel instances that shares the same host memory should be on same bank
- Data burst

```
#pragma HLS INTERFACE m_axi port=res_sa_len offset=slave bundle=gmem0 max_write_burst_length=256
#pragma HLS INTERFACE m_axi port=res_sa_itv offset=slave bundle=gmem0 max_write_burst_length=256
#pragma HLS INTERFACE m_axi port=buf offset=slave bundle=gmem1 max_read_burst_length=256 max_write_burst_length=256
#pragma HLS INTERFACE m_axi port=occ offset=slave bundle=gmem2 max_read_burst_length=256
#pragma HLS INTERFACE m_axi port=cum offset=slave bundle=gmem3 max_read_burst_length=256
#pragma HLS INTERFACE m_axi port=read offset=slave bundle=gmem3 max_read_burst_length=256
```

Synthesis Timing

```
+ Timing:
    * Summary:
    +--------+---------+----------+------------+
    |  Clock |  Target | Estimated| Uncertainty|
    +--------+---------+----------+------------+
    |ap_clk  | 3.33 ns | 3.033 ns |   0.90 ns  |
    +--------+---------+----------+------------+

+ Latency:
    * Summary:
    +---------+---------+----------+----------+-----+-----+---------+
    |  Latency (cycles) |  Latency (absolute) |  Interval | Pipeline|
    |   min   |   max   |    min   |    max   | min | max |   Type  |
    +---------+---------+----------+----------+-----+-----+---------+
    |        ?|        ?|         ?|         ?|    ?|    ?|   none  |
    +---------+---------+----------+----------+-----+-----+---------+
```

Hardware Utilization

* Summary:

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	329704	-
FIFO	-	-	-	-	-
Instance	8	-	1145	1723	-
Memory	5	-	0	0	0
Multiplexer	-	-	-	13427	-
Register	-	-	77250	-	-
Total	13	0	78395	344854	0
Available SLR	1344	2976	871680	435840	320
Utilization SLR (%)	~0	0	8	79	0
Available	2688	5952	1743360	871680	640
Utilization (%)	~0	0	4	39	0

Hardware Utilization

* Memory:

Memory	Module	BRAM_18K	FF	LUT	URAM	Words	Bits	Banks	W*Bits*Banks
occ_r_U	bwa_align_occ_r	2	0	0	0	1024	32	1	32768
read_r_U	bwa_align_read_r	1	0	0	0	16	8	1	128
res_sa_itv_w_0_U	bwa_align_res_sa_itv_w_0	1	0	0	0	256	32	1	8192
res_sa_itv_w_1_U	bwa_align_res_sa_itv_w_0	1	0	0	0	256	32	1	8192
Total		5	0	0	0	1552	104	4	49280

Implementation Issues

- Our kernel at first runs correctly in Emulation-SW, but produced incorrect result in Emulation-HW
- Kernel had better use 1D array as argument
- Input/Output buffering is necessary
- Recurrece may have variable branch size but we make it fixed

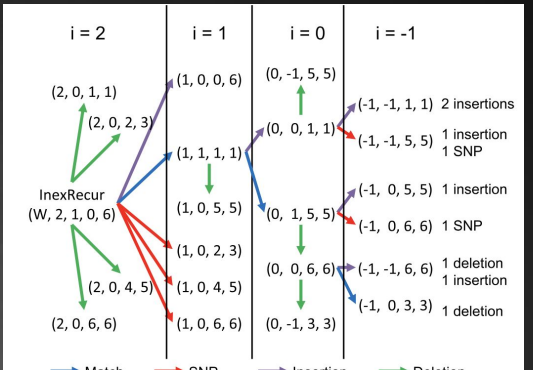


Fig. 3: Mapping of short-read W

```
for (int head = 0, tail = 1; head < tail; head++) {
  // pop head
  i = buf_r[0];  z = buf_r[1];
  k = buf_r[2];  l = buf_r[3];

  for (int s = 0; s < 4; s++) {
    int k_nxt = cum_r[s] + occ_r[k - 1][s];
    int l_nxt = cum_r[s] + occ_r[l][s] - 1;

    if (k_nxt <= l_nxt) {
      // SNP (substitute) alpha[s]
      buf_w[2 * s + 1][0] = i;
      buf_w[2 * s + 1][1] = z - 1;
      buf_w[2 * s + 1][2] = k_nxt;
      buf_w[2 * s + 1][3] = l_nxt;

      if (match_symbol(s, read_r[i])) {
        // match alpha[s]
        buf_w[2 * s + 2][0] = i - 1;
        buf_w[2 * s + 2][1] = z;
        buf_w[2 * s + 2][2] = k_nxt;
        buf_w[2 * s + 2][3] = l_nxt;
```

Conclusion

- To use larger dataset -> Closer to practice
  - Data transfer policy becomes critical due to limited memory
- Fully utilize memory bandwidth to obtain high throughput
- Cache locally to reduce memory access

What's Learnt

- Emulation has few error message, hard to debug
- Though a quite simple kernel, building is time-consuming
  - Emulation-HW: 20+ minutes; Hardware: 2+ hours
  - (May there be incremental build?)
- Justify clearly before actually implement any optimization
- Strictly follow: Emulation-SW -> Emulation-HW -> Hardware

Reference

- Waidyasooriya, H. M., et al. (2013). Implementation of a custom hardware-accelerator for short-read mapping using Burrows-Wheeler alignment, IEEE.
- https://github.com/Xilinx/Vitis_Accel_Examples/tree/master/host/overlap
- https://github.com/Xilinx/SDAccel-Tutorials/blob/master/docs/using-multiple-cu/
- Vitis Unified Software Platform Documentation - Application Acceleration Development (UG1393), Xilinx
- Lab3 Slides of High-Level Synthesis, Jiin Lai
- OpenBWT, 2008-2009 Yuta Mori