XILINX

Host Code Optimization

Introduction to Vitis

Methodology for Host cost Optimization

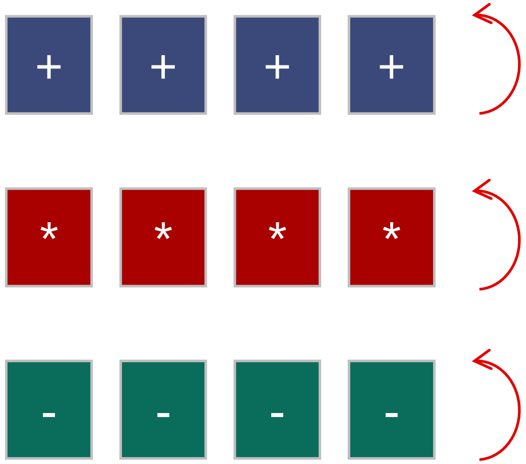
- Optimizing system performance
  - Host optimization
  - Kernel optimization
- Three main areas:
  - Reducing the overhead of kernel enqueing
  - Optimizing data movement
  - Scheduling of the compute units

XILINX

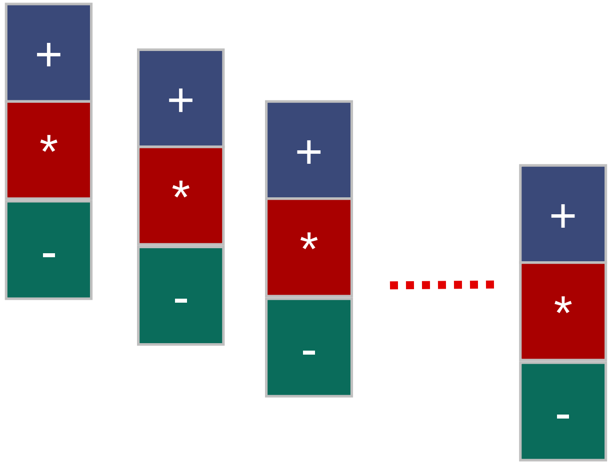
Data parallelism vs Task parallelism

- OpenCL supports Data parallelism and task parallelism

- Data parallelism
  - Same operations are performed on different subsets of data



- Task parallelism
  - Different operations or tasks scheduled on the same or different data

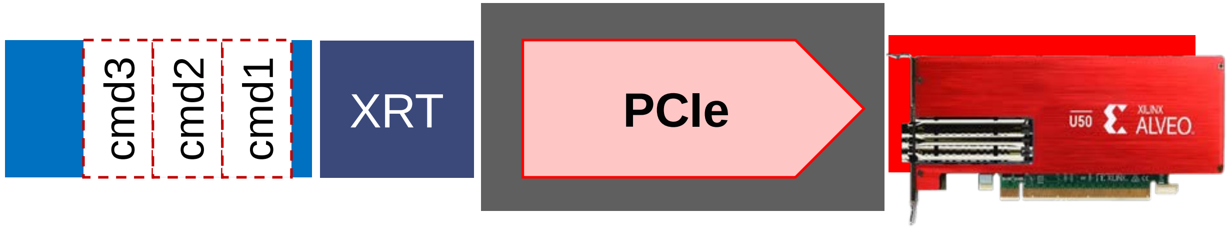


XILINX.

Reducing the Overhead of Kernel Enqueing

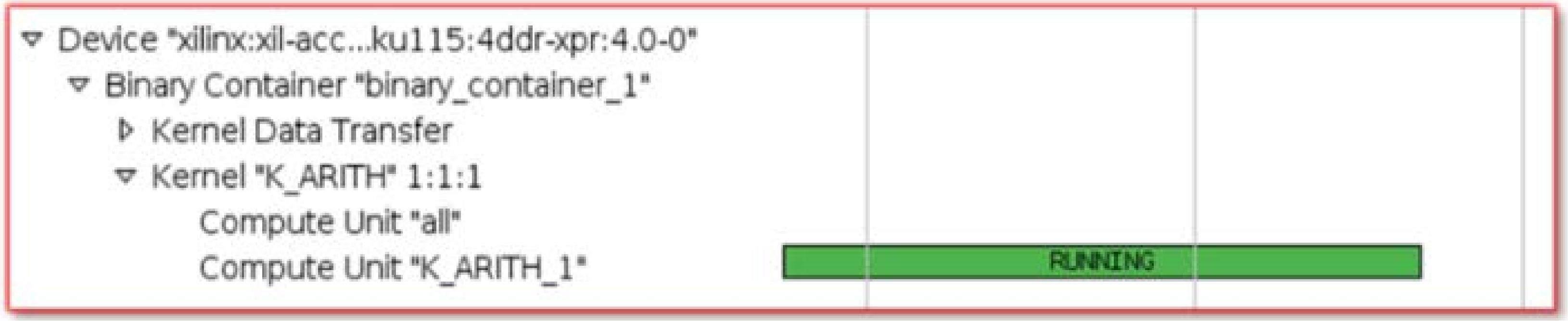
<u>`clEnqueueTask`</u>:
Task parallel workload to kernel

`clEnqueueNDRangeKernel`:
Data parallel workload to kernel

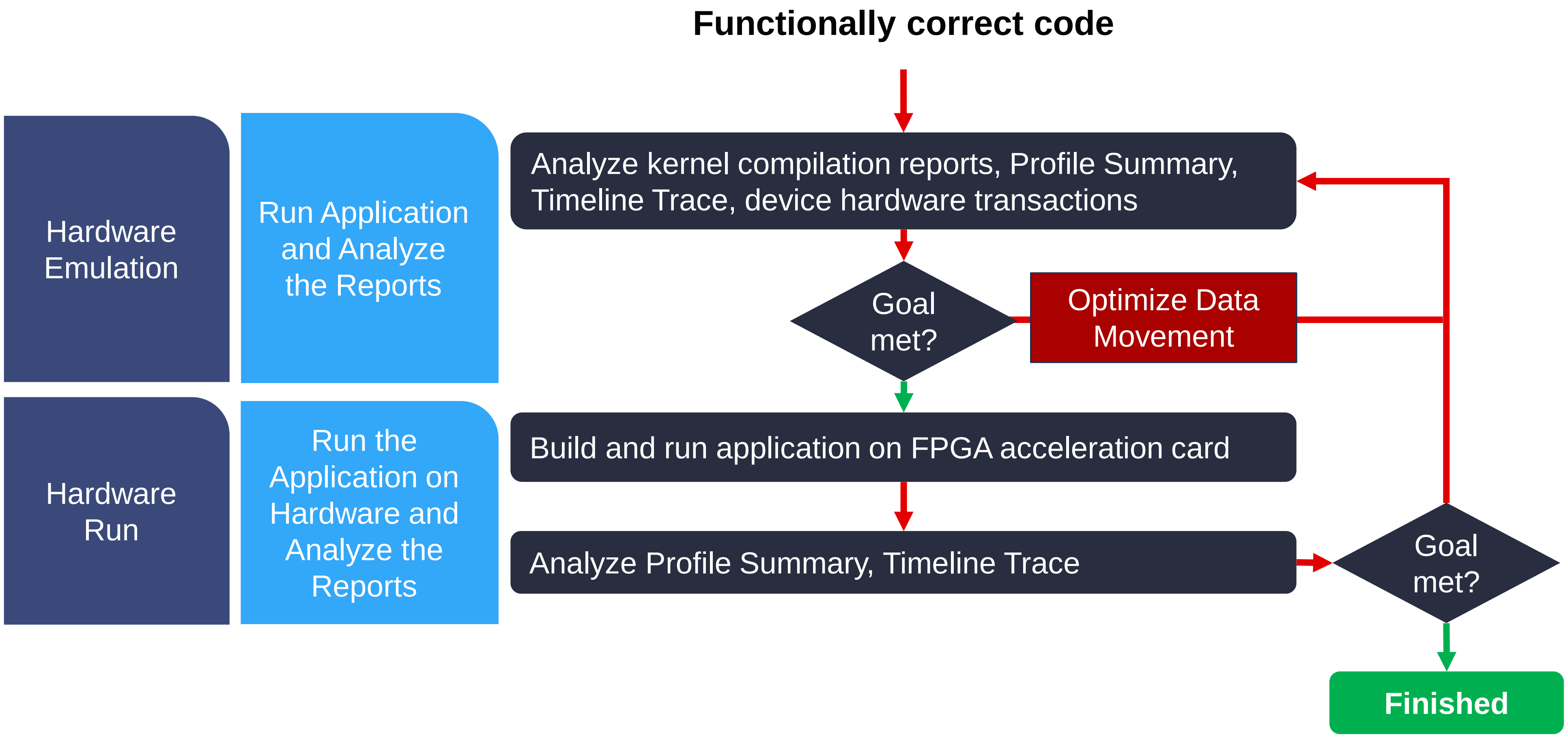


Global Size	4096
Local Size	512
Work Groups	8

Global Size	1
Local Size	1
Work Groups	1

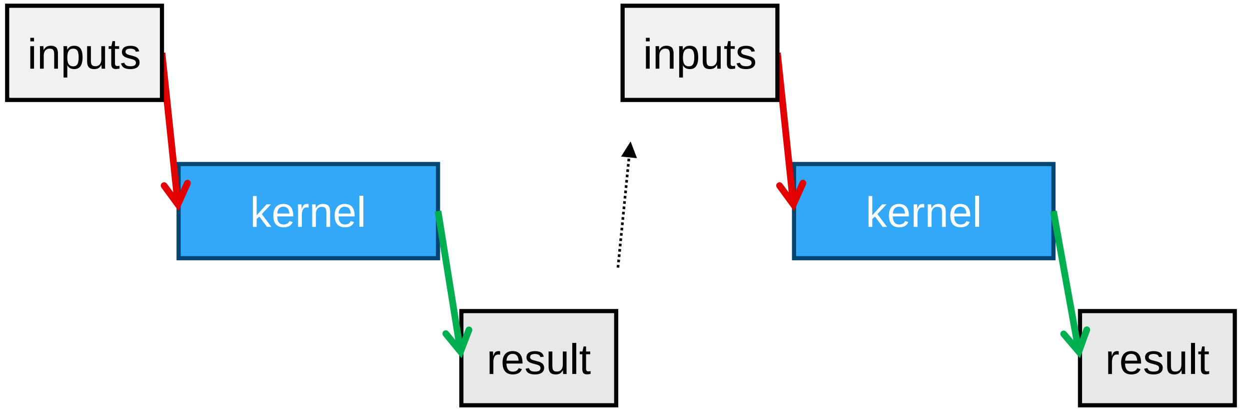


Optimizing Data Movement

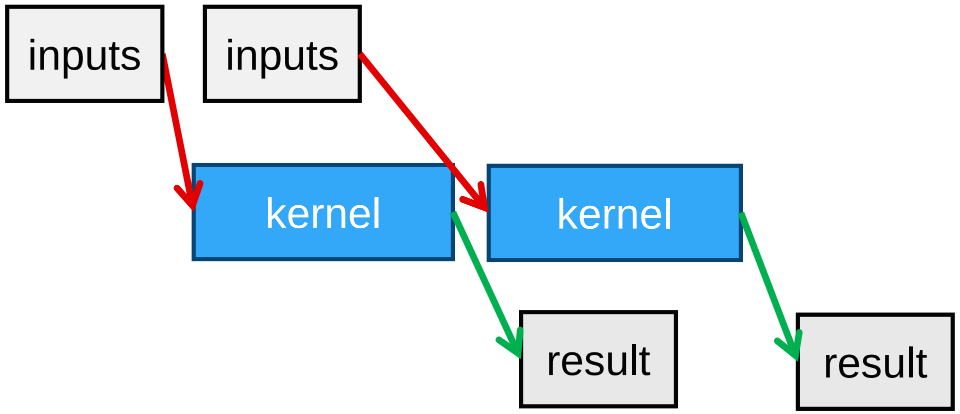


Optimizing Data Movement – Overlapping Data Transfers with Kernel Computation

- Large datasets need to be transferred to the target in smaller blocks
  - Use techniques to overlap the data transfers with the computation to optimize performance
- Using an out-of-order command queue, data transfer and kernel execution can overlap
  - OpenCL EVENT object can be used to setup and synchronize dependencies



Sequential commands



Out-of-order overlapping commands

Optimizing Data Movement – Buffer Memory Segmentation

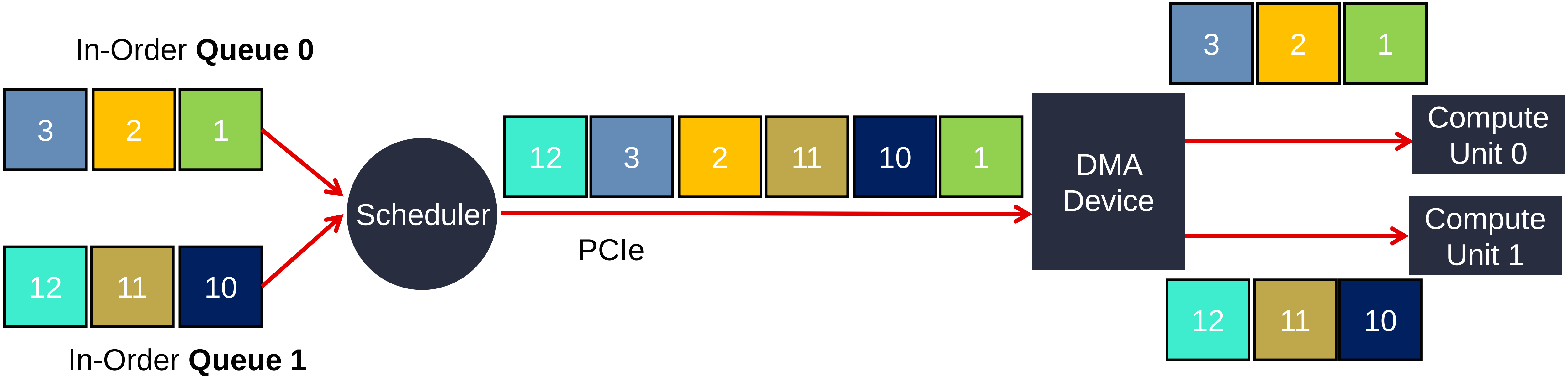
- Allocation/deallocation of buffers can lead to memory segmentation
  - May occur when multiple pthreads for different compute units are used
    - Threads allocate and release many buffers every time they enqueue the kernels
  - May result in sub-optimal performance

- Buffers should be continuous
  - May take time for space to be freed when many buffers are allocated and deallocated
  - <u>Allocate device buffer and reuse between different enqueues of a kernel</u>

Scheduling of Compute Units

- Important when implementing multiple compute units
- There are two ways of executing the kernel
  - Multiple in-order command queues
  - Single out-of-order command queues

Scheduling of Compute Units – In-order Command Queue



- **Commands from queue 0, 1 can be scheduled in any order**
- **You must manage synchronization between queues if required**

XILINX

Scheduling of Compute Units – Out-of-order Command Queue

- **The scheduler can dispatch commands from the queue in any order**
- **You must manually define event dependencies and synchronizations as required**

Thank You