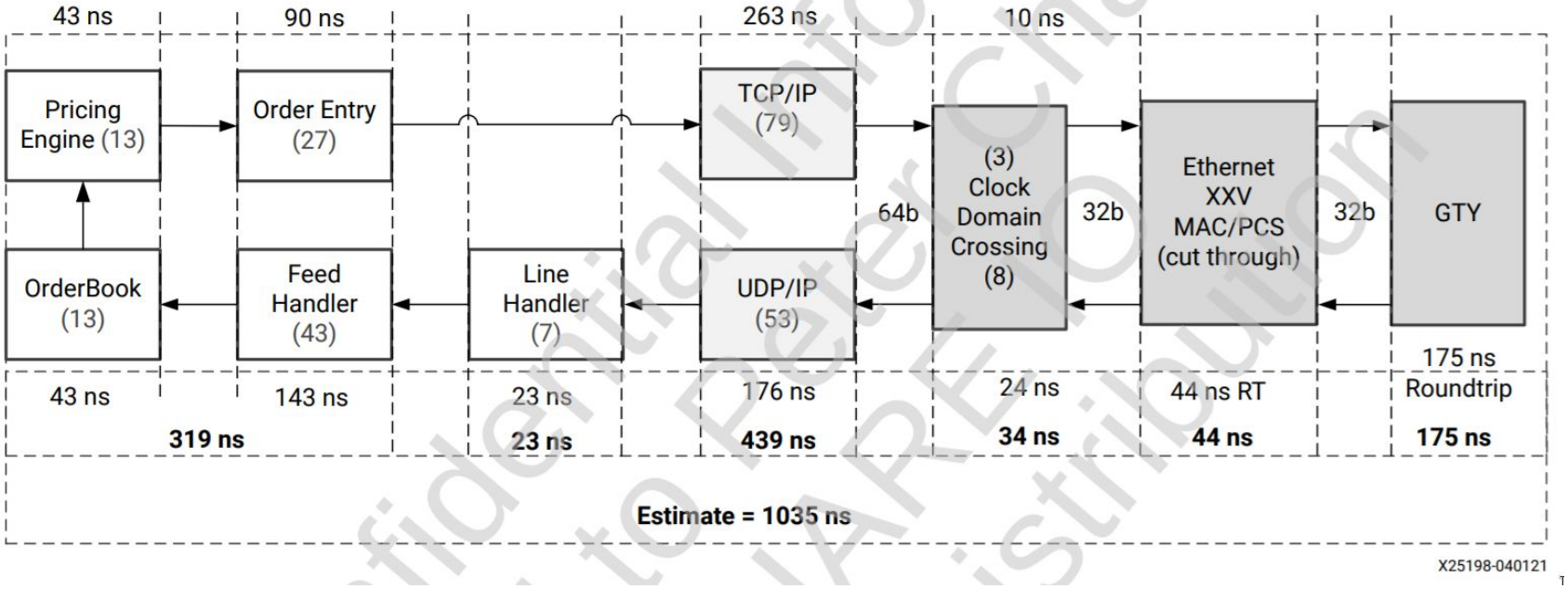
HLS Final Project

Team10

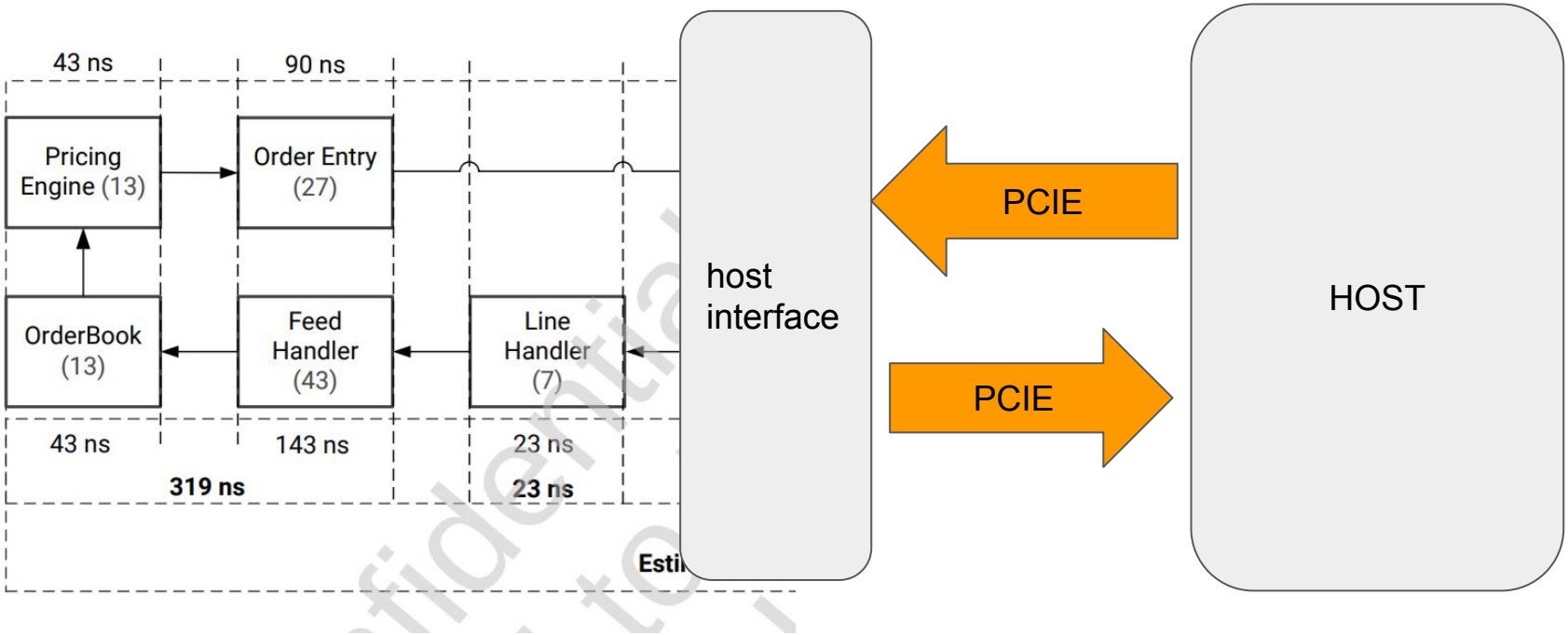
Outline

- Introduction
  - Introduction to AAT platform
  - Building system and OpenCL workflow
- Implementation Details
  - SDAccel™ stream interface
  - Multi-instence connection
- Single-component hardware testbench

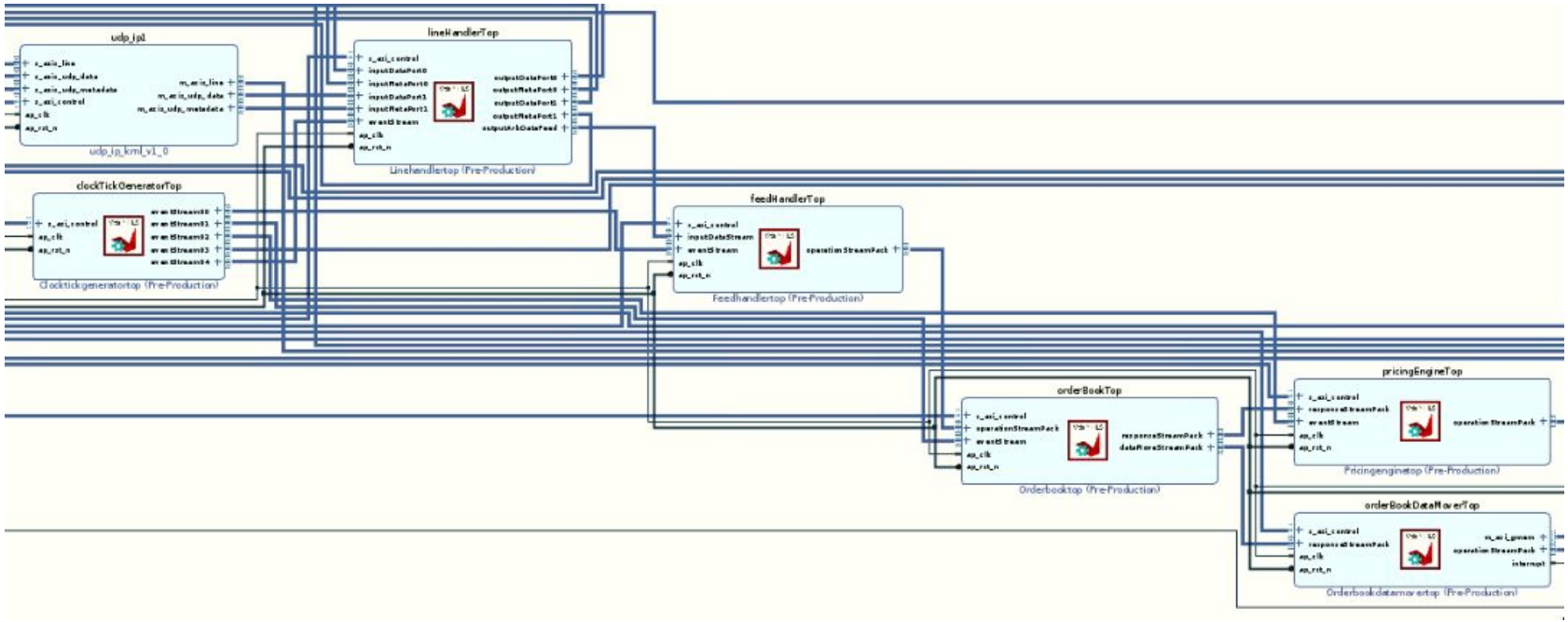
Introduction to AAT platform



Introduction to AAT platform



AAT platform block design

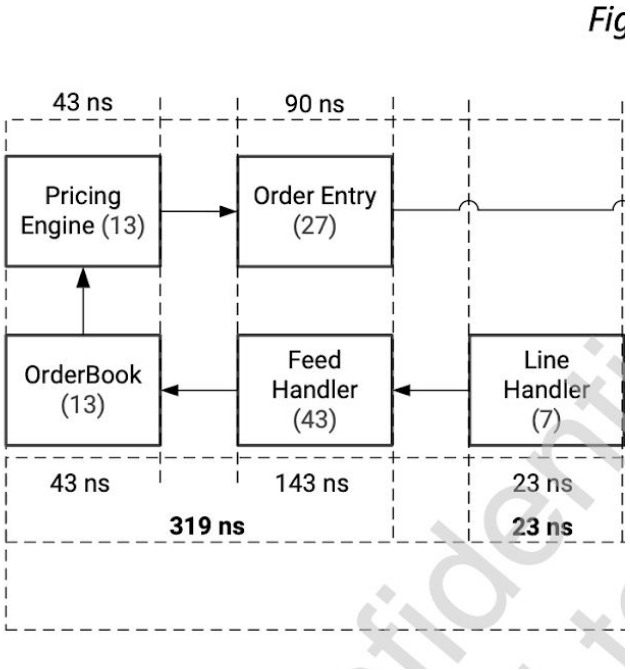


Try to run build script with lots of versions of vitis

- Ubuntu 18.04 (浩杰's machine)
  - Vitis 2019
    - Failed due to unsupported ardument
  - Vitis 2020.2
    - Failed due to lack of some shared library in vivado (wierd?)
    - Some error during vivado installation
  - Vitis 2020.2 + update
  - Vitis 2020.3
- Ubuntu 20.04 (宥儒's machine)
  - Vitis 2020.2
    - Move U50 card to the machine and reinstall the drivers, etc…
    - Successfully run build script, while error during bitstream generation stage
      - all xo file was generated

Componentwise cosimulation

- Feed Handler
  - csynth
- OrderBook
  - cosim
- OrderBook-data-mover
  - csynth
- Pricing Engine
  - cosim
- Order Entry
  - cosim

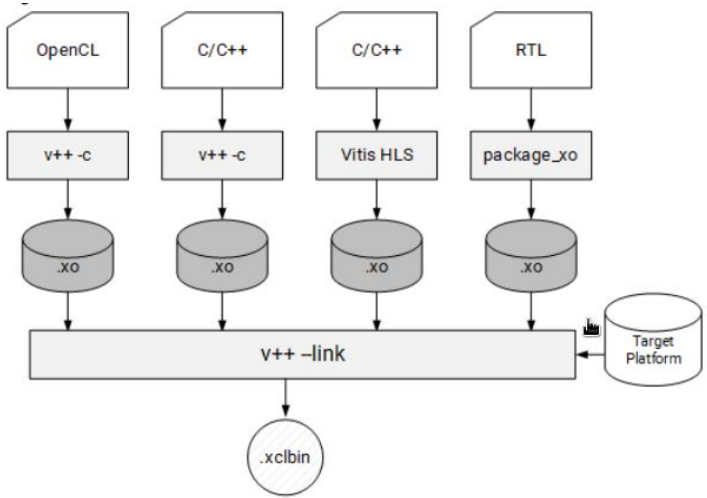


Xilinx Build System

- Build through v++
- Build through vitis TCL script

Xilinx Build System

- V++ building stage
  - Compile c/c++ into `.xo` files
    - v++ -c
  - Link xo files into `.link.xclbin`
    - v++ --link
  - Paskage into `.xclbin`
    - v++ --package



Xilinx Build System

- Build through tcl script
  - add_files
  - set_top
  - open_solution
  - create_clock
  - config_compile
  - csynth_design
  - export_design
    - export into xo files

OpenCL workflow

- setting up kernel
  - cl::Context(...)
  - cl::CommandQueue(...)
  - cl::Program(...)
  - cl::Kernel(...)
- Migrate buffer and launch kernel
  - kernel.setArg
  - enqueueWriteBuffer
  - enqueueMigrateMemObjects
  - enqueueNDRangeKernel

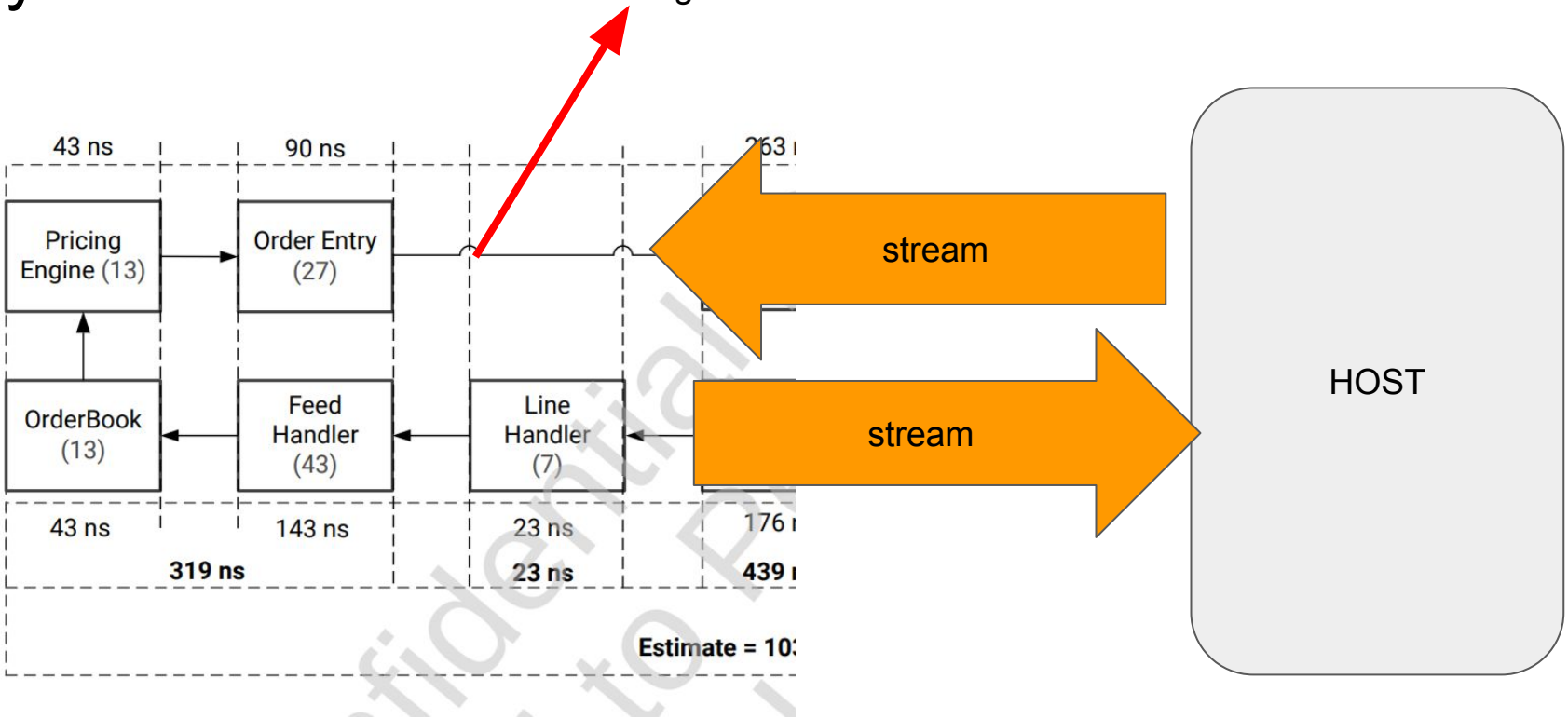
Running Emulation and Hardware

- SW_EMU
  - emconfigutil --platform xilinx_u200_xdma_201830_2
  - XCL_EMULATION_MODE=sw_emu ./host.exe kernel.xclbin
- HW_EMU
  - XCL_EMULATION_MODE=hw_emu ./host.exe kernel.xclbin
- HW
  - ./host.exe kernel.xclbin

Implementation Details

- SDAccel™ stream interface
- Multi-instence connection

System overview



SDAccel™ stream interface

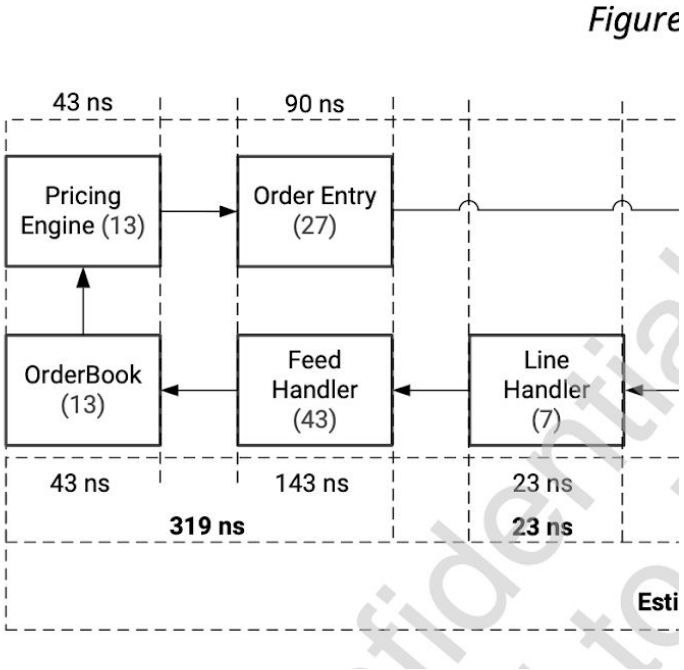
- Xilinx SDAccel™ 2019.1 release a set of OpenCL extensions
- QDMA
- Direct streaming of data from host to kernel and kernel to host without having to go through global memory
    - The host application does not necessarily need to know the size of the data coming from the kernel.
    - Data resides on the host memory can be transferred to the kernel as soon as it is needed. Similarly, the processed data can be transferred back when it is required.

SDAccel™ stream interface

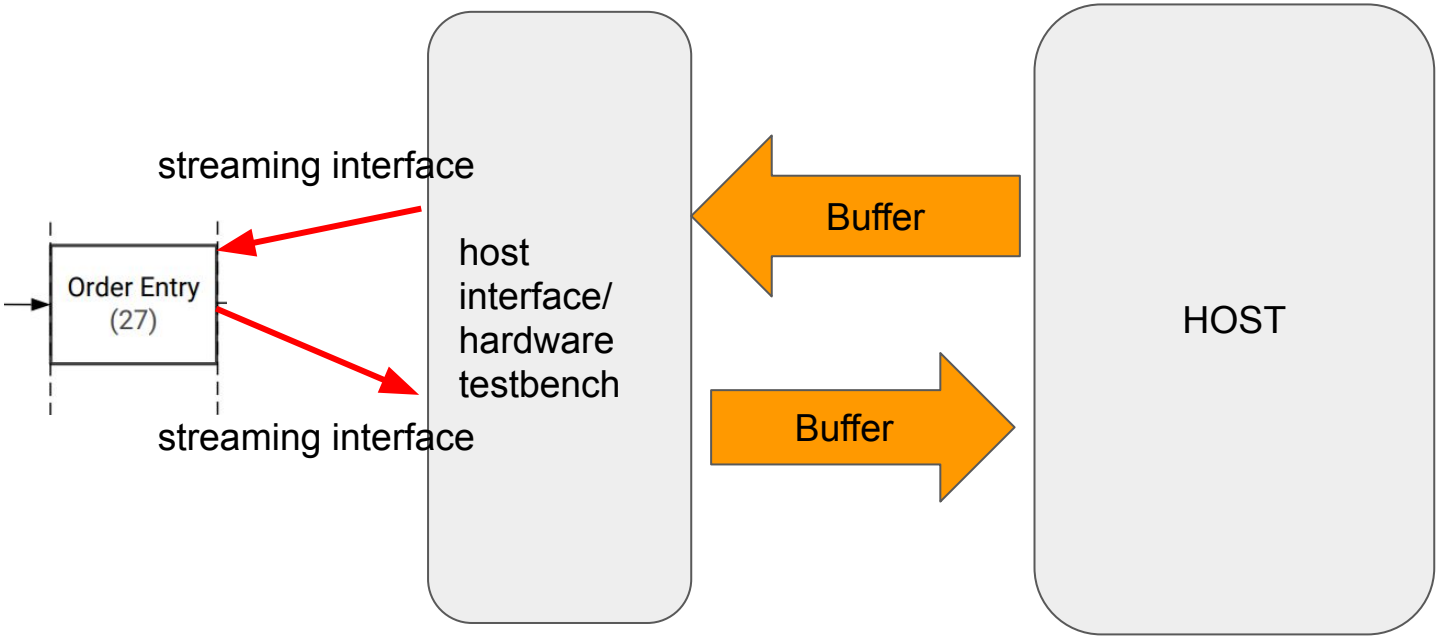
- clCreateStream()
- clReleaseStream()
- clWriteStream()
- clReadStream()
- clPollStreams()

Componentwise cosimulation

- Feed Handler
  - csynth
- OrderBook
  - cosim
- OrderBook-data-mover
  - csynth
- Pricing Engine
  - cosim
- **Order Entry**
  - **cosim**



System Overview



Multiple Instances Kernel

- customize the kernel linking stage to instantiate multiple hardware compute units (CUs) from a single kernel.
- Specify the connection during linking stage `connect.cfg`

Creating Multiple Instances of a Kernel

- Customize the kernel linking stage to instantiate multiple hardware compute units (CUs) from a single kernel.
- #nk=<kernel name>:<number>:<cu_name>.<cu_name>...
- e.g., nk=vadd:3:vadd_X.vadd_Y.vadd_Z

Mapping Kernel Ports to Memory

- We can manually specify which global memory bank each kernel port (or interface) is connected to. Proper configuration of kernel to memory connectivity is important to maximize bandwidth, optimize data transfers, and improve overall performance.
- #sp=<cu_name>.<interface_name>:HOST[0]
- e.g., sp=cnn_1.m_axi_gmem:HOST[0]

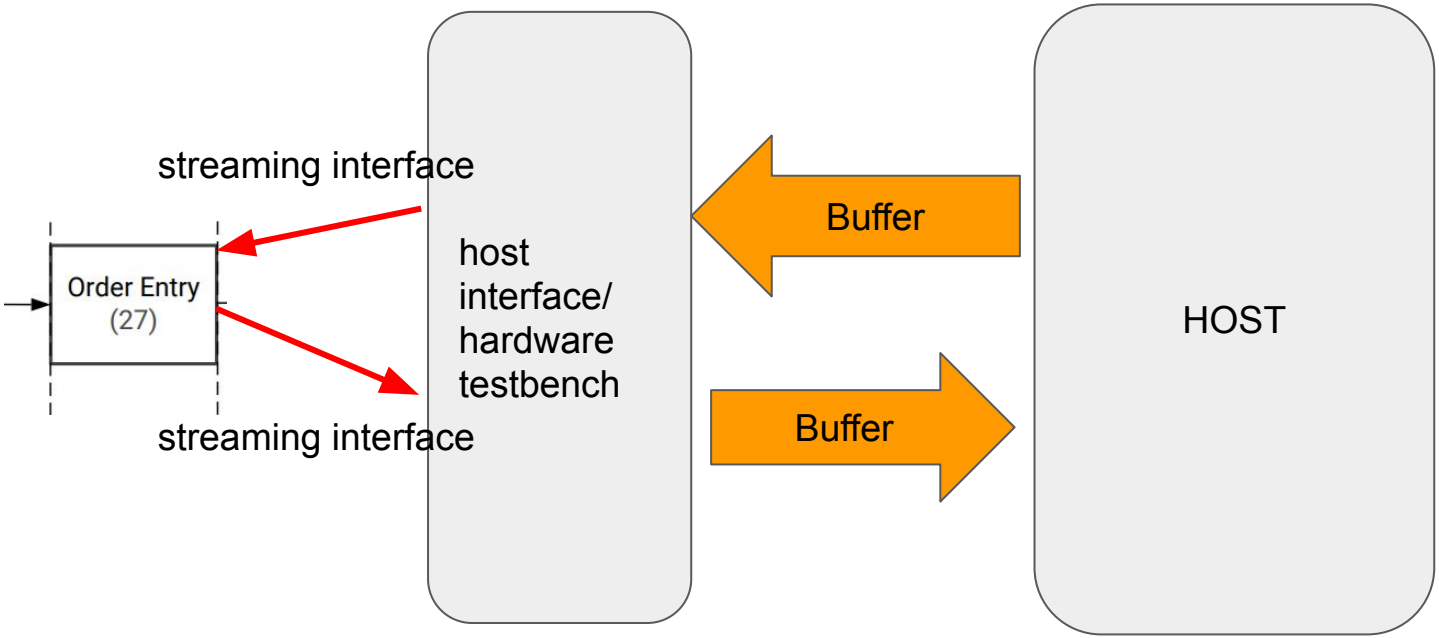
Specifying Streaming Connections between Compute Units

- The Vitis core development kit supports streaming data transfer between two kernels, allowing data to move directly from one kernel to another without having to transmit back through global memory.
- #sc=<cu_name>.<output_port>:<cu_name>.<input_port>:[<fifo_depth>]
- e.g., sc=vadd_1.stream_out:vadd_2.stream_in

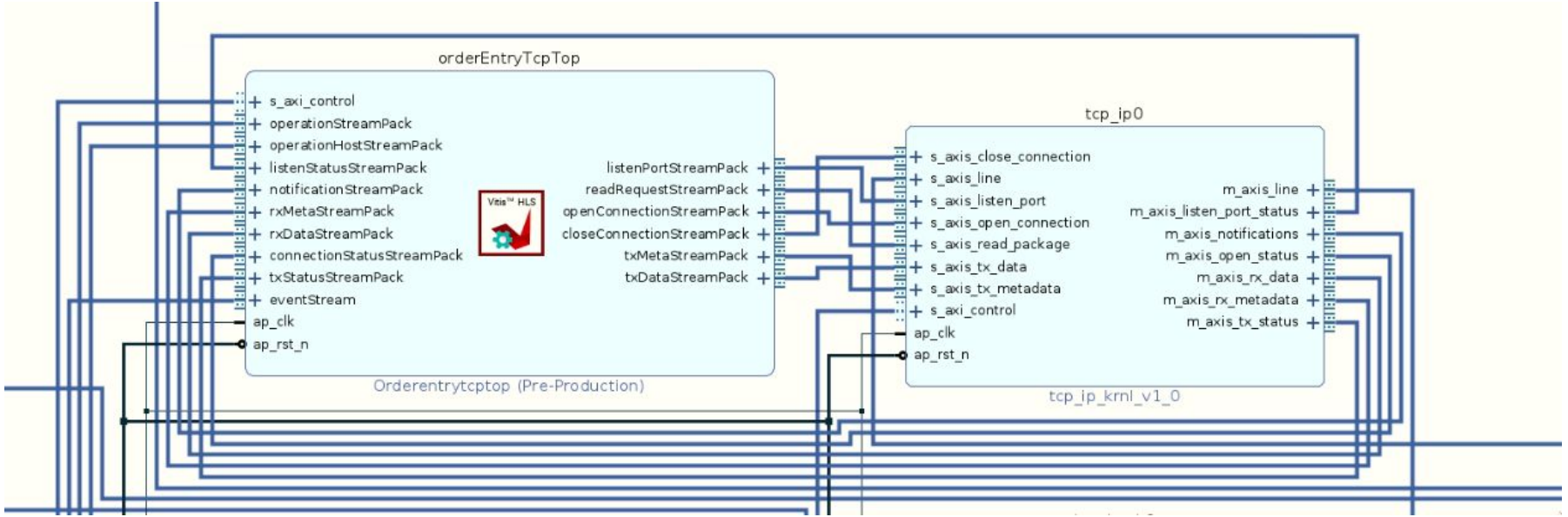
Specifying Streaming Connections between Compute Units

- Xilinx devices on Data Center accelerator cards use stacked silicon consisting of several Super Logic Regions (SLRs) to provide device resources, including global memory. We need to manually assign the kernel instance, or CU into the same SLR as the global memory to ensure the best performance.
- #slr=<compute_unit_name>:<slr_ID>
- e.g., slr=vadd_1:SLR2

System Overview



order entry block design



Hardware testbench simulation

- listenStatus
- txStatusPack
  - sessionID
  - length
  - space
- operationStreamPackFIFO
- ...

Components

- FeedHandler
  - Receives market data(in Simple Binary Encoding) from previous block
  - Create book update messages, e.g. add, modify, delete
- OrderBook
  - Receive and parse book update instructions from FeedHandler
  - Update orderbook
- PricingEngine
  - Receive notification of change in orderbook
  - Check trading strategy and output a reqeust (bid/ask messages)
- OrderEntry
  - Receive request from PricingEngine
  - Construct TCP/IP message

OrderBook::operationProcess

- II Violation
- Unable to enforce a carried dependence constraint (II = 1, distance =1, offset = 1) between 'store' operation ('kernel_orderBookAskCount_y_addr_13_write_ln391') of constant <constant:_ssdm_op_Write.bram.i160> on array 'kernel_orderBookAskCount_V' and 'load' operation ('_Val2_') on array 'kernel_orderBookAskCount_V'.

The II Violation in module 'operationProcess' (function 'operationProcess'): Unable to enforce a carried dependence constraint (II = 1, distance = 1, offset = 1) between 'store' operation ('kernel_orderBookAskCount_V_addr_13_write_ln391') of constant <constant:_ssdm_op_Write.bram.i160> on array 'kernel_orderBookAskCount_V' and 'load' operation ('__Val2__') on array 'kernel_orderBookAskCount_V'.

Github

- Single-componet hardware testbench
  - https://github.com/eee4017/HLS_FINAL

Reference

- the direct streaming of data from host to kernel and kernel to host without having to go through global memory
  - https://www.xilinx.com/html_docs/xilinx2019_1/sdaccel_doc/qwz1555342848145.html
  - https://github.com/Xilinx/Vitis_Accel_Examples/tree/2020.2/host/streaming_reg_access
- Multi-instance connection
  - https://www.xilinx.com/html_docs/xilinx2020_2/vitis_doc/vitiscommandcompiler.html#ariaid-title5
  - https://www.xilinx.com/html_docs/xilinx2020_2/vitis_doc/buildingdevicebinary.html#:~:text=these%20build%20options.-,Creating%20Multiple%20Instances%20of%20a%20Kernel,-By%20default%2C%20the
  - https://github.com/Xilinx/Vitis_Accel_Examples/tree/master/host/mult_compute_units