EVEREST

DESIGN ENVIRONMENT
FOR EXTREME-SCALE BIG DATA ANALYTICS
ON HETEROGENEOUS PLATFORMS

CPS Summer School – Sep 19, 2023

Tutorial

Olympus: How to use HLS for building customized memory architectures

CHRISTIAN PILATO

EVEREST Scientific Coordinator
Assistant Professor, Politecnico di Milano

christian.pilato@polimi.it

STEPHANIE SOLDAVINI

PhD Student, Politecnico di Milano

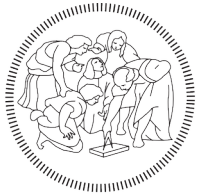
stephanie.soldavini@polimi.it

STEP ONE

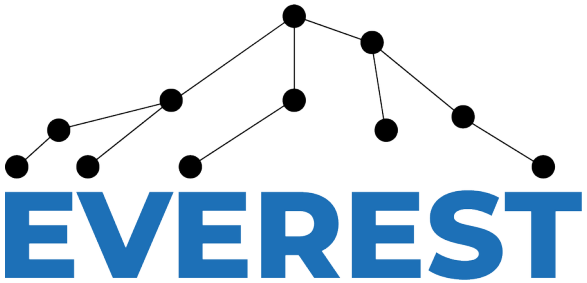
- Download Olympus: https://tinyurl.com/OlympusCPS23
- Decompress the .tar.gz inside the VM
- Inside olympus/ install libs by running:
  - **`make init`**
  - Your password is “password”

About Me – Stephanie Soldavini

- BS & MS in Computer Engineering from Rochester Institute of Technology (2014-2019)
- PhD Student at Politecnico di Milano (2020-Now)



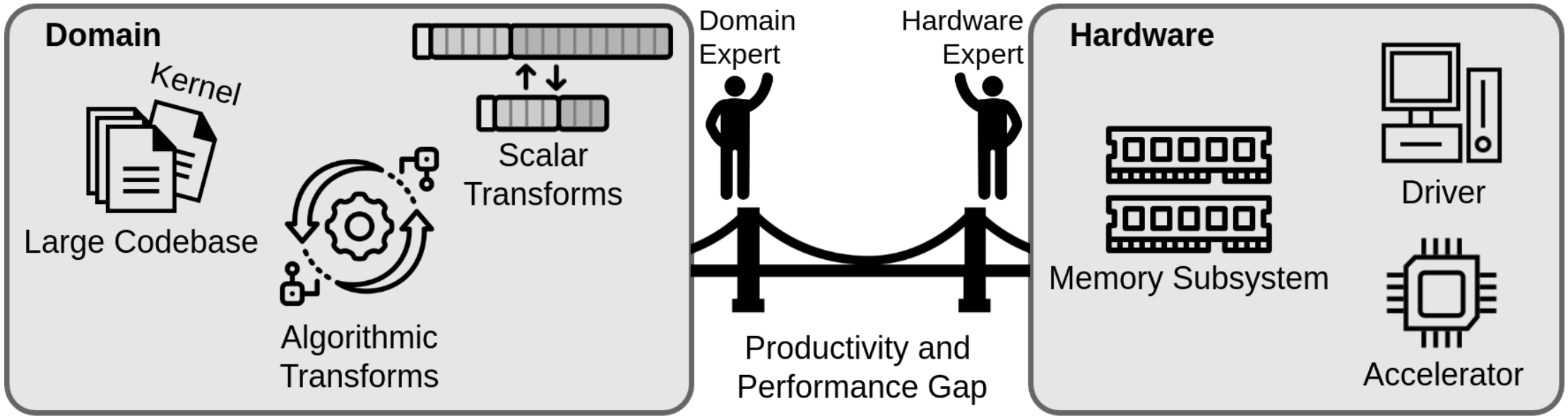
POLITECNICO
MILANO 1863



https://tinyurl.com/CPS2022olympus

Problem

- **Productivity**: Application designers usually do not have the necessary hardware design knowledge to create efficient hardware architectures
- **Performance**: Fine-tuned hardware descriptions are required to efficiently coordinate data transfers and execution



https://tinyurl.com/CPS2022olympus

EVEREST

Target Boards:
Alveo Data Center Accelerator Cards with HBM

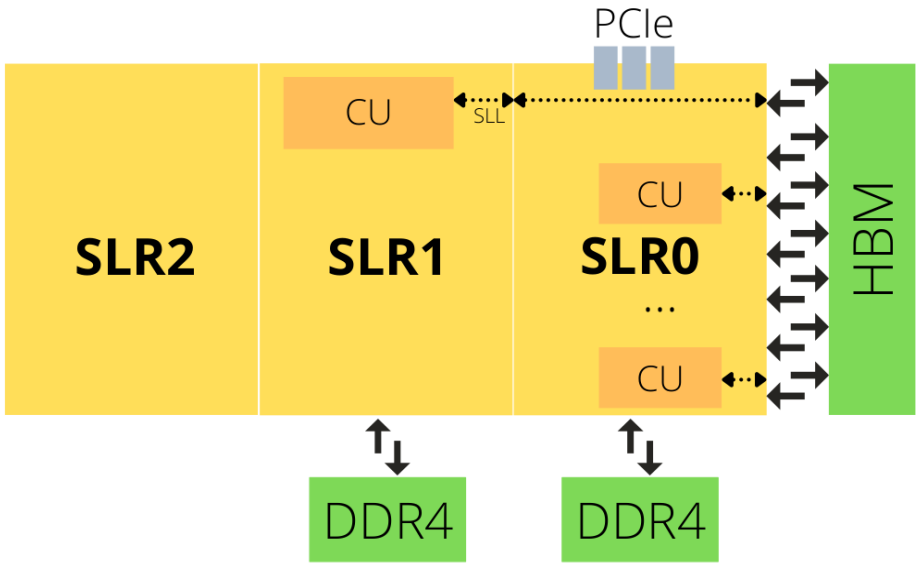
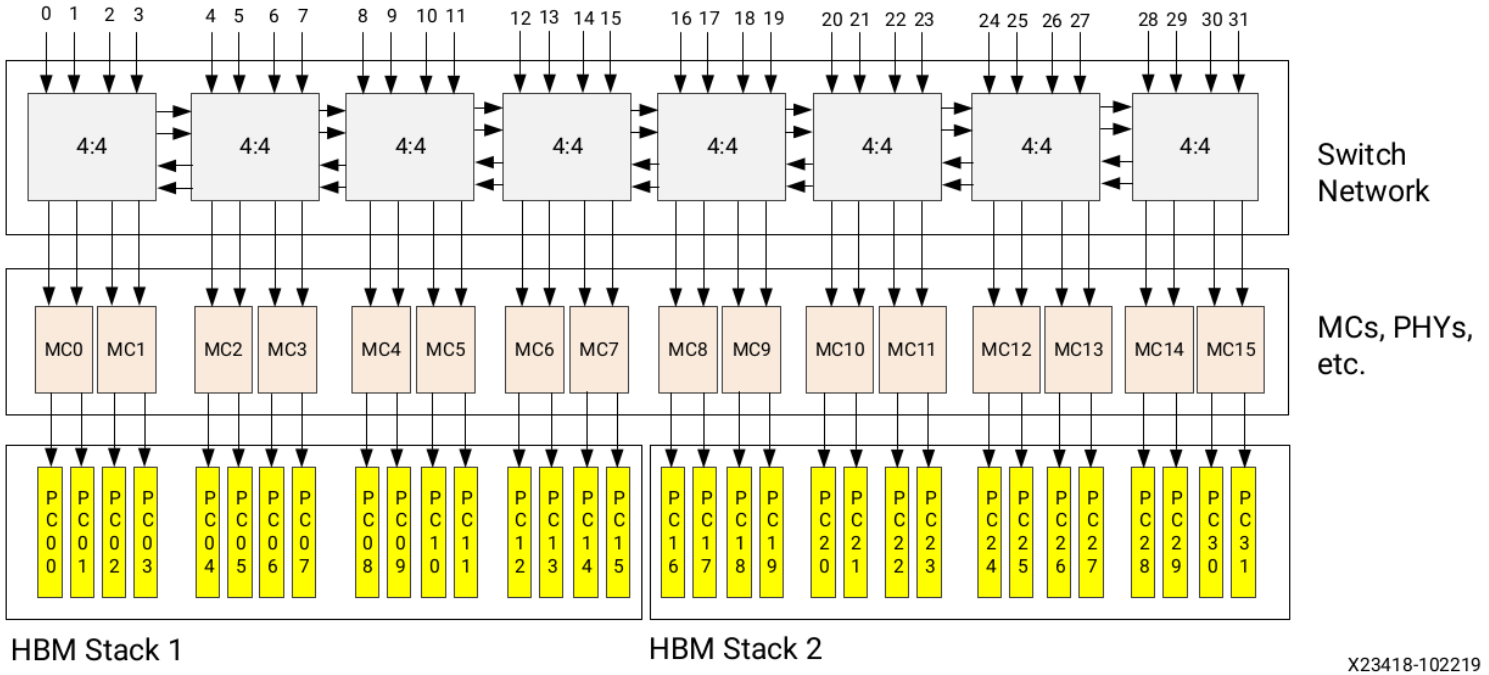


Table 1. Alveo U280 SLR resources.

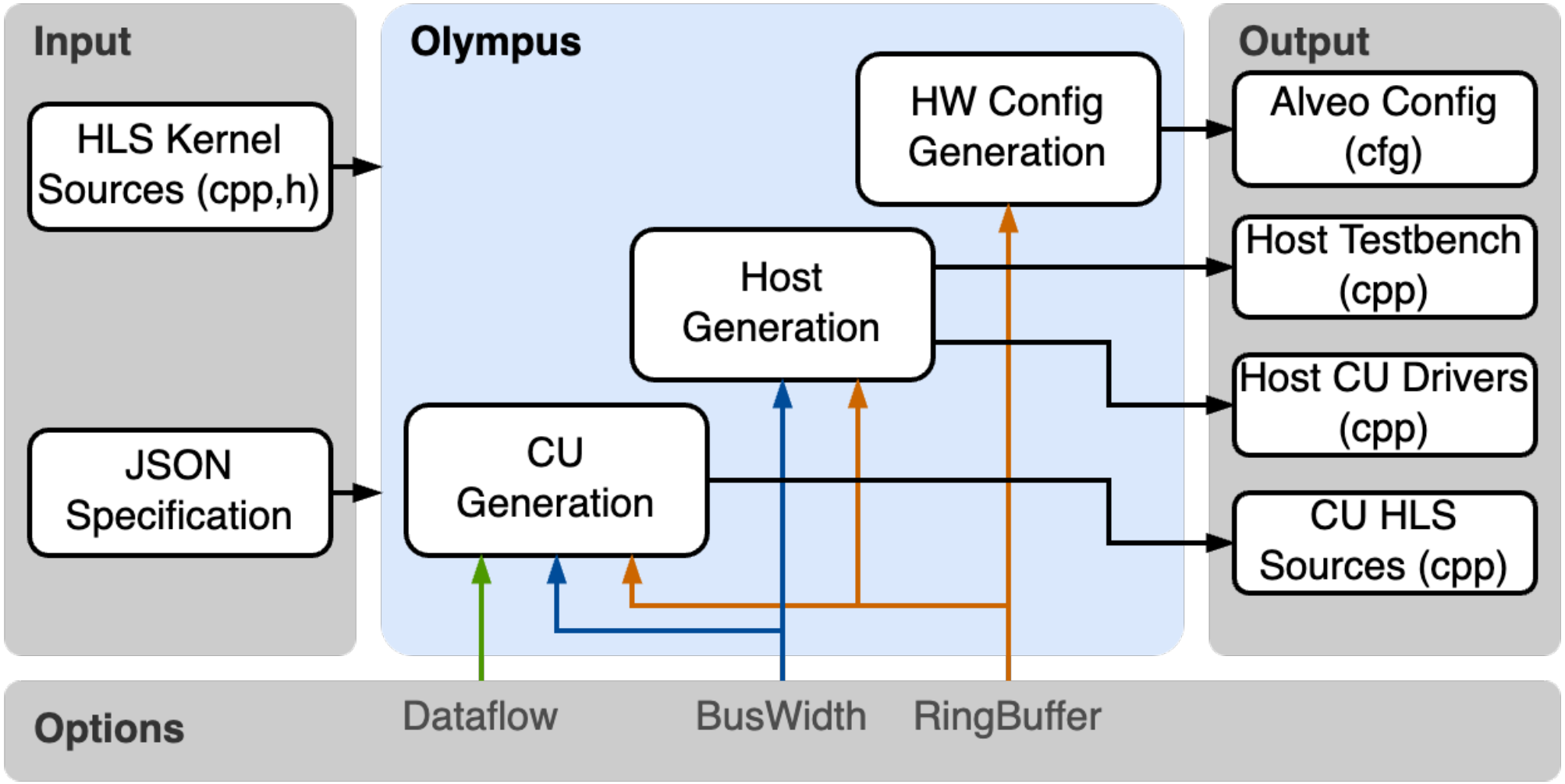
Resources	SLR0	SLR1	SLR2
HBM	32×256MB	-	-
DDR4	16GB	16GB	-
PLRAM	2×4MB	2×4MB	2×4MB
CLB LUT	369K	333K	367K
CLB Register	746K	675K	729K
Block RAM tile	507	468	512
UltraRAM	320	320	320
DSP	2,733	2,877	2,880



https://tinyurl.com/CPS2022olympus

EVEREST

Olympus Flow Diagram



https://tinyurl.com/CPS2022olympus

EVEREST

Inputs: C HLS Kernel

- C code of application kernel to be accelerated
- Standard C array interfaces

input/kernel_body.cpp

```
#include "kernel_body.h"

void kernel_body(double S[121], double D[1331], double u[1331], double v[1331],
        double t[1331], double r[1331], double t1[1331],
        double t3[1331], double t0[1331], double t2[1331])
{
    for (int c1 = 0; c1 <= 10; c1 += 1)
        for (int c2 = 0; c2 <= 10; c2 += 1)
            for (int c3 = 0; c3 <= 10; c3 += 1) {
                // stmt0
                t1[121 * c1 + 11 * c2 + c3] = 0;
                for (int c8 = 0; c8 <= 10; c8 += 1)
                    // stmt0
                    t1[121 * c1 + 11 * c2 + c3] = t1[121 * c1 + 11 * c2 + c3] + S[11 *
            }
    for (int c1 = 0; c1 <= 10; c1 += 1)
        for (int c2 = 0; c2 <= 10; c2 += 1)
            for (int c3 = 0; c3 <= 10; c3 += 1) {
                // stmt1
                t0[121 * c1 + 11 * c2 + c3] = 0;
                for (int c8 = 0; c8 <= 10; c8 += 1)
                    // stmt1
                    t0[121 * c1 + 11 * c2 + c3] = t0[121 * c1 + 11 * c2 + c3] + S[11 *
            }
    for (int c1 = 0; c1 <= 10; c1 += 1)
        for (int c2 = 0; c2 <= 10; c2 += 1)
            for (int c3 = 0; c3 <= 10; c3 += 1) {
                // stmt2
                t[121 * c1 + 11 * c2 + c3] = 0;
                for (int c8 = 0; c8 <= 10; c8 += 1)
                    // stmt2
                    t[121 * c1 + 11 * c2 + c3] = t[121 * c1 + 11 * c2 + c3] + S[11 * c1
            }
```

Inputs: JSON Kernel Specification

- Format derived from the JSON required for the Vitis RTL Blackbox flow[1]
- Required info:
  - c_function_name
    - the name of the kernel function AND filename (cpp & h)
  - param_type
    - “mm” : memory mapped C arrays
    - “stream” : Xilinx hls::stream<>
  - c_parameters
    - Details on each port interface (must be in the same order as in the C source
    - c_name : Name of the array
    - c_type : Data type of a single element
    - c_port_direction : in, out, or inout
    - depth : Number of elements in the array

input/helmholtz.json

```
{
    "c_function_name": "kernel_body",
    "param_type": "mm",
    "c_parameters": [
        {
            "c_name": "S",
            "c_type": "double",
            "c_port_direction": "in",
            "depth": 121,
        },
        {
            "c_name": "D",
            "c_type": "double",
            "c_port_direction": "in",
            "depth": 1331,
        },
        {
            "c_name": "u",
            "c_type": "double",
            "c_port_direction": "in",
            "depth": 1331,
        },
        {
            "c_name": "v",
            "c_type": "double",
            "c_port_direction": "out",
            "depth": 1331,
        },
        {
            "c_name": "t",
            "c_type": "double",
            "c_port_direction": "inout",
            "depth": 1331
        },
        {
            "c_name": "r",
            "c_type": "double",
            "c_port_direction": "inout",
            "depth": 1331
        },
        {
            "c_name": "t1",
            "c_type": "double",
            "c_port_direction": "inout",
            "depth": 1331
        },
        {
            "c_name": "t3",
            "c_type": "double",
            "c_port_direction": "inout",
            "depth": 1331
        },
        {
            "c_name": "t0",
            "c_type": "double",
            "c_port_direction": "inout",
            "depth": 1331
        },
        {
            "c_name": "t2",
            "c_type": "double",
            "c_port_direction": "inout",
            "depth": 1331
        }
    ]
}
```

[1] https://docs.xilinx.com/r/2021.1-English/ug1399-vitis-hls/JSON-File-for-RTL-Blackbox

Makefile

- KERNEL_DIR
  - Directory where the kernel HLS sources are
- KERNEL_JSON
  - The JSON spec file path
- RING_BUF
  - The degree of ring buffering
- BUS_WIDTH
  - The bitwidth of the bus to global memory
- STREAMS
  - Whether or not to use a dataflow streaming architecture
  - 1=dataflow, 0=not dataflow
- N_CU
  - Number of CUs to instantiate in the FPGA

```
14 # Olympus options
15 KERNEL_DIR  ?= input/
16 KERNEL_JSON ?= input/helmholtz.json
17 RING_BUF   ?= 2
18 BUS_WIDTH  ?= 64
19 STREAMS    ?= 1
20 N_CU       ?= 1
```

```
15 KERNEL_DIR  ?= input/
16 KERNEL_JSON ?= input/helmholtz.json
17 RING_BUF   ?= 2
18 BUS_WIDTH  ?= 64
19 STREAMS    ?= 1
20 N_CU       ?= 1
21
22 TEST      ?= CLEAN
23
24 #host code
25 HOST_SRC   ?= HostSampleTop.gen.cpp AlveoHost.cpp HostImpl.gen.cpp ../src/$(KERNEL_MODEL).cpp
26 EXEC_FLAGS ?=
27 #kernel code
28 CU_SRC      ?= CU.cpp $(KERNEL_BODY).cpp
29 CU_NAME     ?= krnl_helm
30 KFLAGS      ?=
31 #run flags
32 RUN_FLAGS   ?= $(N_CU) $(POINTS)
33
34 #design configuration
35 CFG         ?= Design.cfg
36
37 include $(BOARD_TARGET_PATH)/targets.Makefile
```

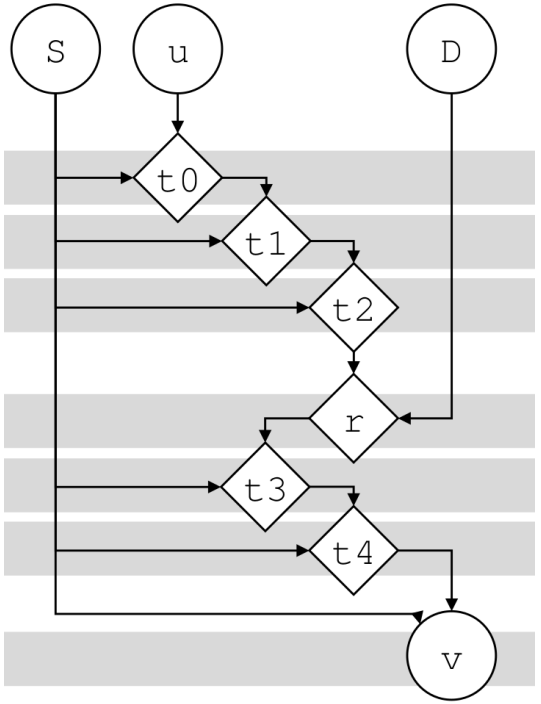
Test Kernel: Inverse Helmholtz

- Tensor operator commonly used in computational fluid dynamics (CFD) applications
- Implemented in C as 7 loop nests of depth 3-4
  - `input/kernel_body.cpp`

$$t_{ijk,e} = \sum_{l=0}^{p}\sum_{m=0}^{p}\sum_{n=0}^{p} S^T_{li}\cdot S^T_{mj}\cdot S^T_{nk}\cdot u_{lmn,e} = (S\otimes S\otimes S\otimes u_e)_{iljmkn}{}^{lmn} \quad (1a)$$

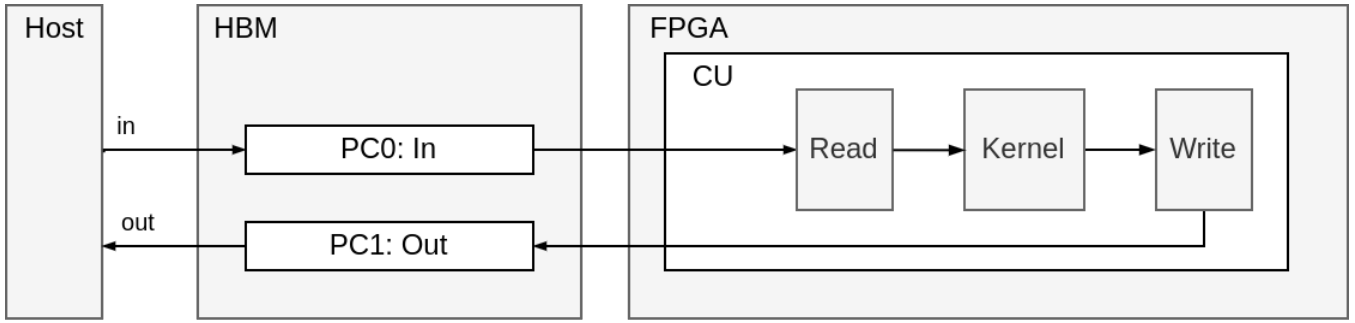
$$r_{ijk,e} = D_{ijk,e}\cdot t_{ijk,e} \quad (1b)$$

$$v_{ijk,e} = \sum_{l=0}^{p}\sum_{m=0}^{p}\sum_{n=0}^{p} S_{li}\cdot S_{mj}\cdot S_{nk}\cdot r_{lmn,e} = (S\otimes S\otimes S\otimes r_e)_{limjnk}{}^{lmn} \quad (1c)$$



EVEREST

Basic Implementation



- `make olympus`
  - Generate the code (default values in Makefile yield basic implementation)
  - Sources: `~/alveo_tests/helmholtz_autogen/RB1_BW64_S0-student/krnl_helm/CLEAN/`
    - `src/` : kernel sources, `CU.cpp` is the generated Compute Unit wrapper
    - `host/` : host sources
      - `HostImpl.gen.cpp`: Implementation of driver functions (moving data to global mem, invoking CU)
      - `HostSampleTop.gen.cpp`: A sample test bench `main` file
- `make hls TARGET=hw`
  - Run HLS

Basic - Results

`source /tools/Xilinx/Vitis/2021.1/settings64.sh`

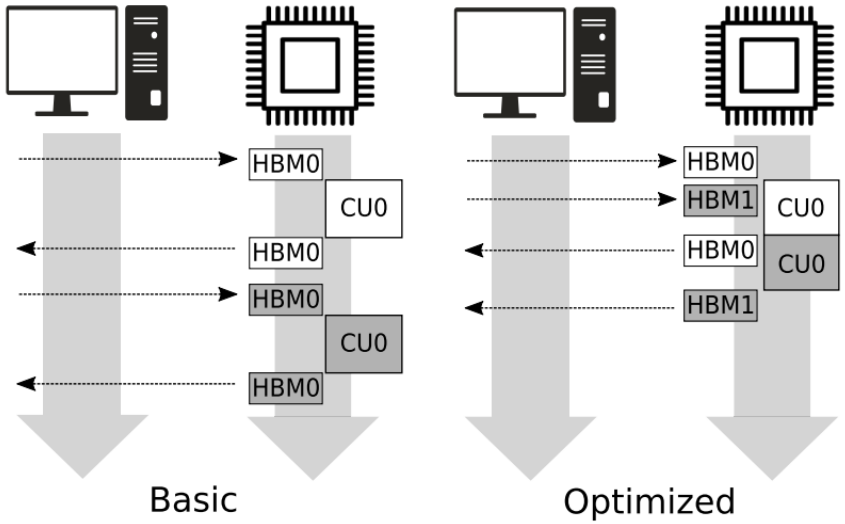
3rd-to-last line of "`make hls`" output: `vitis_analyzer [path]/[kernel name].xo.compile_summary`

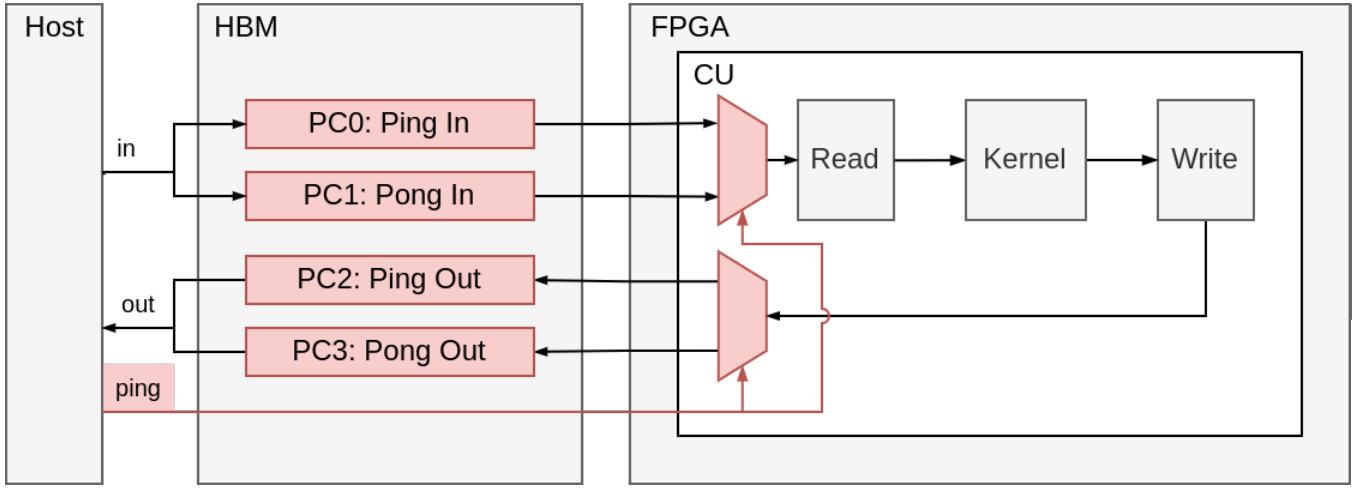
Name	Latency (cycles)	Iteration Latency	Interval
krnl_helm			
krnl_helm_Pipeline_1	124		124
krnl_helm_Pipeline_2	1334		1334
krnl_helm_Pipeline_3	1334		1334
compute_1	9744		9744
compute_1_Pipeline_VITIS_LOOP_7_1_VITIS_LOOP_8_2_VITIS_LOOP_9_3	1398		1398
compute_1_Pipeline_VITIS_LOOP_16_5_VITIS_LOOP_17_6_VITIS_LOOP_18_7	1398		1398
compute_1_Pipeline_VITIS_LOOP_25_9_VITIS_LOOP_26_10_VITIS_LOOP_27_11	1398		1398
compute_1_Pipeline_VITIS_LOOP_34_13_VITIS_LOOP_35_14_VITIS_LOOP_36_15	1343		1343
compute_1_Pipeline_VITIS_LOOP_39_16_VITIS_LOOP_40_17_VITIS_LOOP_41_18	1398		1398
compute_1_Pipeline_VITIS_LOOP_48_20_VITIS_LOOP_49_21_VITIS_LOOP_50_22	1398		1398
compute_1_Pipeline_VITIS_LOOP_57_24_VITIS_LOOP_58_25_VITIS_LOOP_59_26	1398		1398
krnl_helm_Pipeline_4	1334		1334
VITIS_LOOP_41_1		14159	

BRAM	BRAM (%)	DSP	DSP (%)	FF	FF (%)	LUT	LUT (%)
10	~0	135	1	40078	1	27879	2

EVEREST

Ping Pong Buffers





- `make olympus RING_BUF=2`
  - RING_BUF sets the degree of ring buffering, 2 => ping pong
  - Sources: `~/alveo_tests/helmholtz_autogen/RB2_BW64_S0-student/krnl_helm/CLEAN/`
- `make hls RING_BUF=2 TARGET=hw`
  - Run HLS

EVEREST

Ping Pong Buffers - Results

Name	Latency (cycles)	Iteration Latency	Interval
krnl_helm			
krnl_helm_Pipeline_4	124		124
krnl_helm_Pipeline_5	1334		1334
krnl_helm_Pipeline_6	1334		1334
krnl_helm_Pipeline_1	124		124
krnl_helm_Pipeline_2	1334		1334
krnl_helm_Pipeline_3	1334		1334
compute_1	9744		9744
compute_1_Pipeline_VITIS_LOOP_7_1_VITIS_LOOP_8_2_VITIS_LOOP_9_3	1398		1398
compute_1_Pipeline_VITIS_LOOP_16_5_VITIS_LOOP_17_6_VITIS_LOOP_18_7	1398		1398
compute_1_Pipeline_VITIS_LOOP_25_9_VITIS_LOOP_26_10_VITIS_LOOP_27_11	1398		1398
compute_1_Pipeline_VITIS_LOOP_34_13_VITIS_LOOP_35_14_VITIS_LOOP_36_15	1343		1343
compute_1_Pipeline_VITIS_LOOP_39_16_VITIS_LOOP_40_17_VITIS_LOOP_41_18	1398		1398
compute_1_Pipeline_VITIS_LOOP_48_20_VITIS_LOOP_49_21_VITIS_LOOP_50_22	1398		1398
compute_1_Pipeline_VITIS_LOOP_57_24_VITIS_LOOP_58_25_VITIS_LOOP_59_26	1398		1398
krnl_helm_Pipeline_8	1334		1334
krnl_helm_Pipeline_7	1334		1334
VITIS_LOOP_62_1		14159	

Previous:

BRAM	BRAM (%)	DSP	DSP (%)	FF	FF (%)	LUT	LUT (%)
10	~0	135	1	40078	1	27879	2

Now:

BRAM	BRAM (%)	DSP	DSP (%)	FF	FF (%)	LUT	LUT (%)
18	~0	135	1	42747	1	32108	2

Interface	Data Width (SW->HW)
m_axi_gmem0	64 -> 64
m_axi_gmem1	64 -> 64
m_axi_gmem2	64 -> 64
m_axi_gmem3	64 -> 64

Ping Pong Buffers - Results

Name	Latency (cycles)	Iteration Latency	Interval
krnl_helm			
krnl_helm_Pipeline_4	124		124
krnl_helm_Pipeline_5	1334		1334
krnl_helm_Pipeline_6	1334		1334
krnl_helm_Pipeline_1	124		124
krnl_helm_Pipeline_2	1334		1334
krnl_helm_Pipeline_3	1334		1334
compute_1	9744		9744
compute_1_Pipeline_VITIS_LOOP_7_1_VITIS_LOOP_8_2_VITIS_LOOP_9_3	1398		1398
compute_1_Pipeline_VITIS_LOOP_16_5_VITIS_LOOP_17_6_VITIS_LOOP_18_7	1398		1398
compute_1_Pipeline_VITIS_LOOP_25_9_VITIS_LOOP_26_10_VITIS_LOOP_27_11	1398		1398
compute_1_Pipeline_VITIS_LOOP_34_13_VITIS_LOOP_35_14_VITIS_LOOP_36_15	1343		1343
compute_1_Pipeline_VITIS_LOOP_39_16_VITIS_LOOP_40_17_VITIS_LOOP_41_18	1398		1398
compute_1_Pipeline_VITIS_LOOP_48_20_VITIS_LOOP_49_21_VITIS_LOOP_50_22	1398		1398
compute_1_Pipeline_VITIS_LOOP_57_24_VITIS_LOOP_58_25_VITIS_LOOP_59_26	1398		1398
krnl_helm_Pipeline_8	1334		1334
krnl_helm_Pipeline_7	1334		1334
VITIS_LOOP_62_1		14159	

Previous:

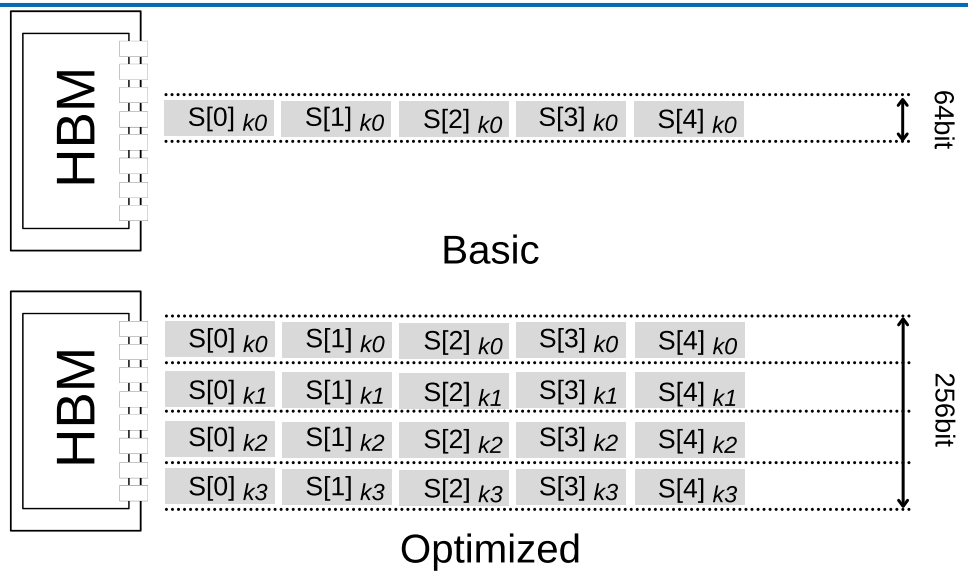
BRAM	BRAM (%)	DSP	DSP (%)	FF	FF (%)	LUT	LUT (%)
10	~0	135	1	40078	1	27879	2

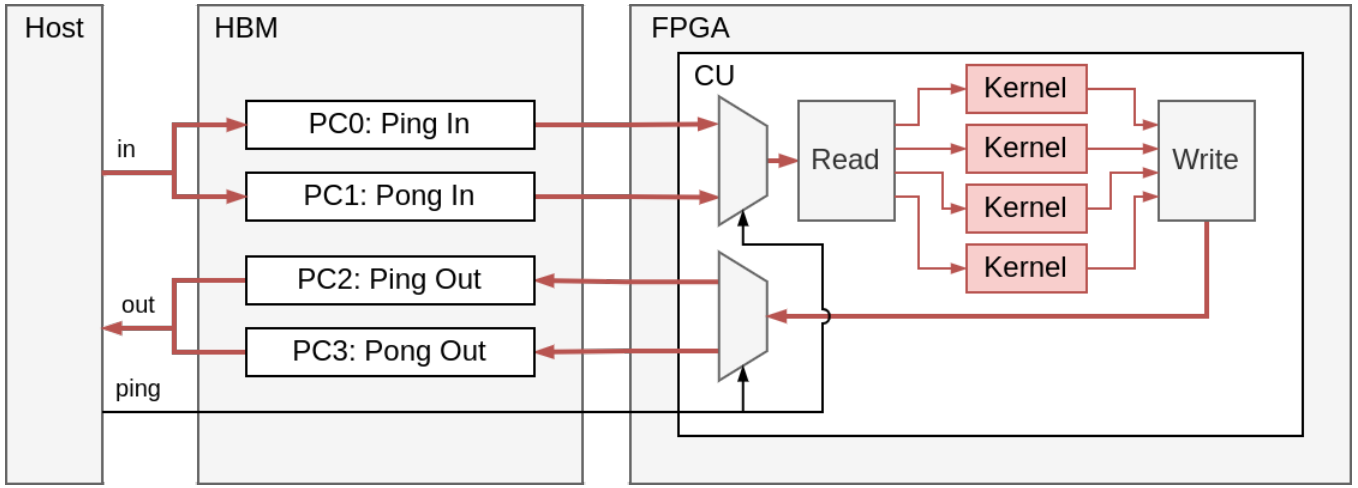
Now:

BRAM	BRAM (%)	DSP	DSP (%)	FF	FF (%)	LUT	LUT (%)
18	~0	135	1	42747	1	32108	2

Interface	Data Width (SW->HW)
m_axi_gmem0	64 -> 64
m_axi_gmem1	64 -> 64
m_axi_gmem2	64 -> 64
m_axi_gmem3	64 -> 64

Wide Bus





- `make olympus RING_BUF=2 BUS_WIDTH=256`
  - BUS_WIDTH sets the width of the bus to global memory. 256/sizeof(double) => 4 "lanes"
  - Sources: `~/alveo_tests/helmholtz_autogen/RB2_BW256_S0-student/krnl_helm/CLEAN/`
- `make hls RING_BUF=2 BUS_WIDTH=256 TARGET=hw`
  - Run HLS

Wide Bus - Results

Name	Issue Type	Latency (cycles)	Iteration Latency	Interval
krnl_helm				
read_data_1		3009		3009
compute		49659		49659
compute_Pipeline_VITIS_LOOP_7_1_VITIS_LOOP_8_2_VITIS_LOOP_9_3	II Violation	8051		8051
compute_Pipeline_VITIS_LOOP_16_5_VITIS_LOOP_17_6_VITIS_LOOP_18_7	II Violation	8051		8051
compute_Pipeline_VITIS_LOOP_25_9_VITIS_LOOP_26_10_VITIS_LOOP_27_11	II Violation	8051		8051
compute_Pipeline_VITIS_LOOP_34_13_VITIS_LOOP_35_14_VITIS_LOOP_36_15		1343		1343
compute_Pipeline_VITIS_LOOP_39_16_VITIS_LOOP_40_17_VITIS_LOOP_41_18	II Violation	8050		8050
compute_Pipeline_VITIS_LOOP_48_20_VITIS_LOOP_49_21_VITIS_LOOP_50_22	II Violation	8050		8050
compute_Pipeline_VITIS_LOOP_57_24_VITIS_LOOP_58_25_VITIS_LOOP_59_26	II Violation	8050		8050
krnl_helm_Pipeline_VITIS_LOOP_133_2		1334		1334
krnl_helm_Pipeline_VITIS_LOOP_117_1		1334		1334
VITIS_LOOP_188_1			54077	

Previous:

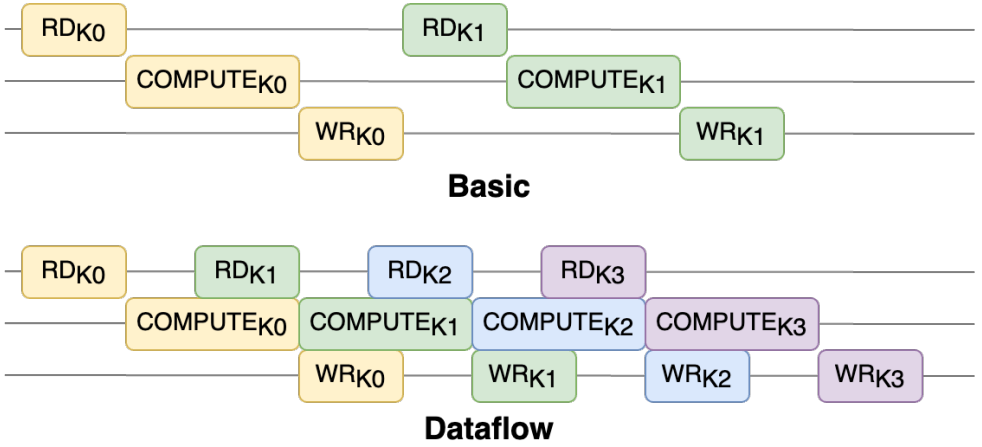
BRAM	BRAM (%)	DSP	DSP (%)	FF	FF (%)	LUT	LUT (%)
18	~0	135	1	42747	1	32108	2

Now:

BRAM	BRAM (%)	DSP	DSP (%)	FF	FF (%)	LUT	LUT (%)
72	1	142	1	109117	4	66257	5

Interface	Data Width (SW->HW)
m_axi_gmem0	256 -> 256
m_axi_gmem1	256 -> 256
m_axi_gmem2	256 -> 256
m_axi_gmem3	256 -> 256

Streaming (1 compute)



- `make olympus RING_BUF=2 BUS_WIDTH=256 STREAMS=1`
  - STREAMS=1 turns on the HLS dataflow pragma and uses the hls::stream data type
  - Sources: `~/alveo_tests/helmholtz_autogen/RB2_BW256_S1-student/krnl_helm/CLEAN/`
- `make hls RING_BUF=2 BUS_WIDTH=256 STREAMS=1 TARGET=hw`
  - Run HLS

Streaming (1 compute) - Results

Name	Latency (cycles)	Iteration Latency	Interval
krnl_helm			
dataflow_in_loop_VITIS_LOOP_229_1	14088		13877
read_data_1	3009		3009
compute_11	13876		13876
compute_12	13876		13876
compute_13	13876		13876
compute_1	13876		13876
compute_1_Pipeline_VITIS_LOOP_162_1	123		123
compute_1_Pipeline_VITIS_LOOP_167_2	1333		1333
compute_1_Pipeline_VITIS_LOOP_172_3	1333		1333
compute_1_Pipeline_VITIS_LOOP_7_1_VITIS_LOOP_8_2_VITIS_LOOP_9_3	1398		1398
compute_1_Pipeline_VITIS_LOOP_16_5_VITIS_LOOP_17_6_VITIS_LOOP_18_7	1398		1398
compute_1_Pipeline_VITIS_LOOP_25_9_VITIS_LOOP_26_10_VITIS_LOOP_27_11	1398		1398
compute_1_Pipeline_VITIS_LOOP_34_13_VITIS_LOOP_35_14_VITIS_LOOP_36_15	1343		1343
compute_1_Pipeline_VITIS_LOOP_39_16_VITIS_LOOP_40_17_VITIS_LOOP_41_18	1398		1398
compute_1_Pipeline_VITIS_LOOP_48_20_VITIS_LOOP_49_21_VITIS_LOOP_50_22	1398		1398
compute_1_Pipeline_VITIS_LOOP_57_24_VITIS_LOOP_58_25_VITIS_LOOP_59_26	1398		1398
compute_1_Pipeline_VITIS_LOOP_178_4	1333		1333
entry_proc	0		0
write_data_1	1407		1407
VITIS_LOOP_229_1			

Previous "Compute" Latency: 49659

Previous Iteration Latency: 54077

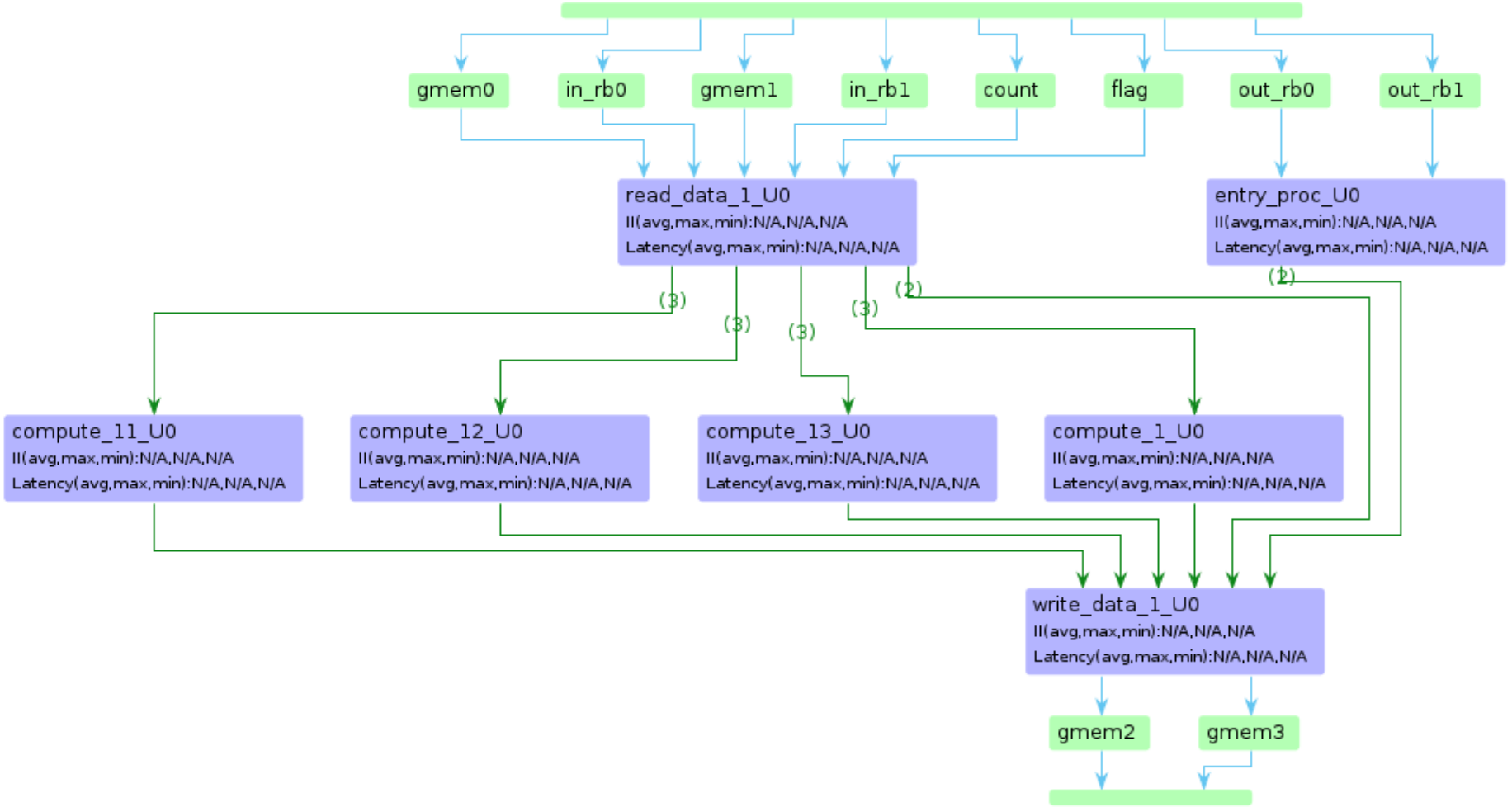
Previous:

BRAM	BRAM (%)	DSP	DSP (%)	FF	FF (%)	LUT	LUT (%)
72	1	142	1	109117	4	66257	5

Now:

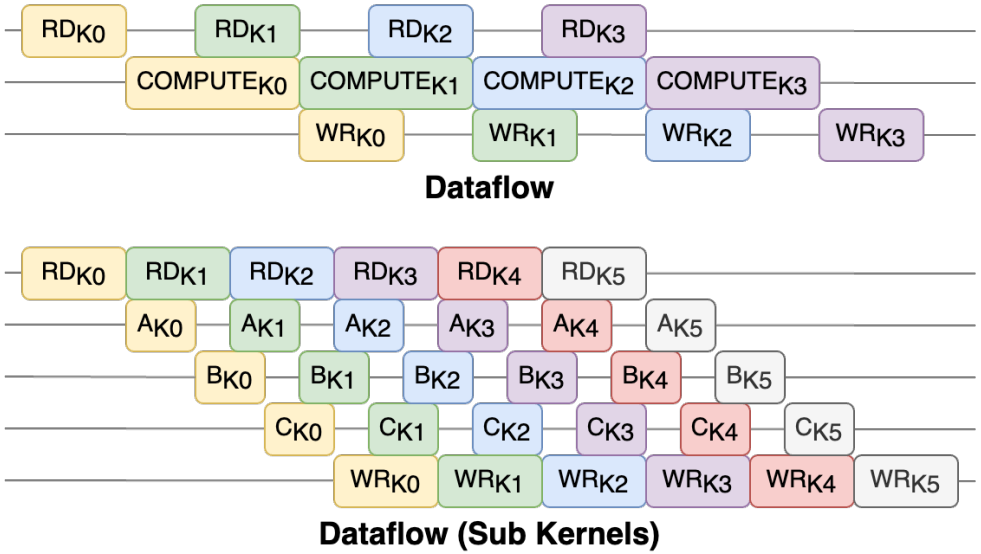
BRAM	BRAM (%)	DSP	DSP (%)	FF	FF (%)	LUT	LUT (%)
104	2	540	5	158002	6	105005	8

Streaming (1 compute) - Results



EVEREST

Streaming (multi compute)



- Multi compute is not controlled by Olympus, but using the INLINE pragma any submodules in the original HLS kernel will be used in the pipeline
- Edit the Makefile:
  - `KERNEL_BODY ?= kernel_body_df`
  - `KERNEL_MODEL ?= kernel_body_sw`
  - `KERNEL_JSON ?= input/helmholtz_df.json`
  - `TEST ?= DF`

- Same commands as before:
  - `make olympus RING_BUF=2 BUS_WIDTH=256 STREAMS=1`
    - Sources: `~/alveo_tests/helmholtz_autogen/RB2_BW256_S1-student/krnl_helm/DF/`
  - `make hls RING_BUF=2 BUS_WIDTH=256 STREAMS=1 TARGET=hw`

Streaming (multi compute) - Results

Name	Latency (cycles)	Iteration Latency	Interval
krnl_helm			
dataflow_in_loop_VITIS_LOOP_199_1	3429		3010
read_data_1	3009		3009
gemm5	2861		2861
...			
gemm	2861		2861
mmult8	1340		1340
mmult15	1340		1340
mmult22	1340		1340
mmult	1340		1340
gemm_inv9	2861		2861
...			
gemm_inv_last	2861		2861
entry_proc	0		0
write_data_1	1407		1407
VITIS_LOOP_199_1			

Previous Dataflow Interval: 13877

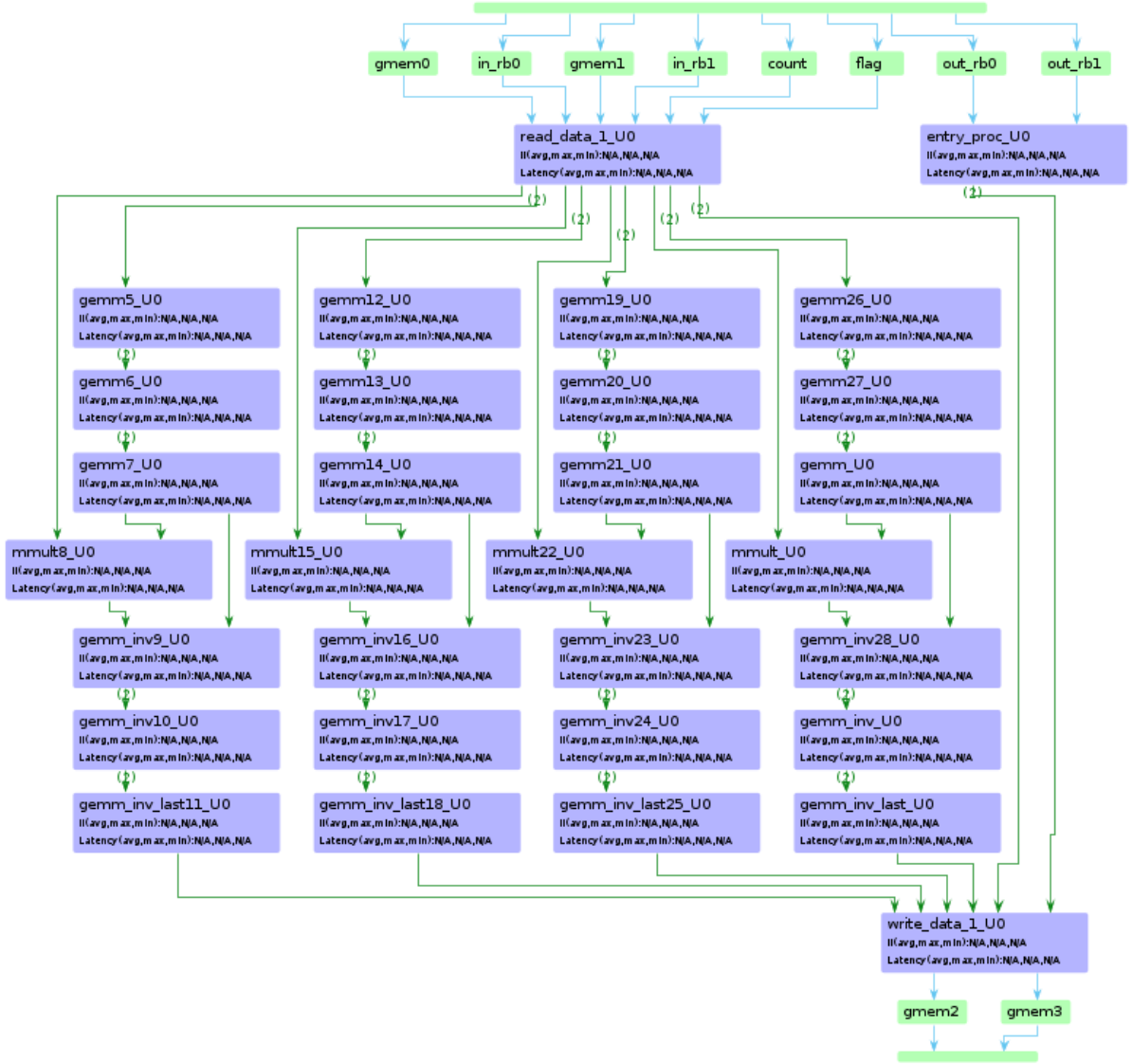
Previous:

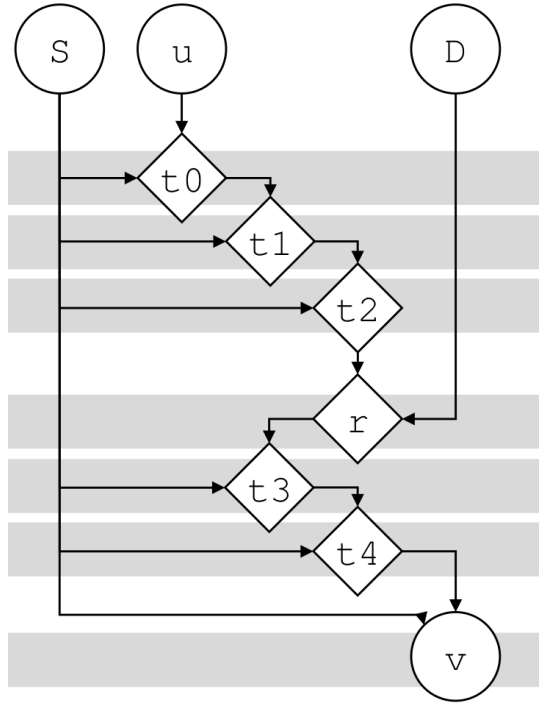
BRAM	BRAM (%)	DSP	DSP (%)	FF	FF (%)	LUT	LUT (%)
104	2	540	5	158002	6	105005	8

Now:

BRAM	BRAM (%)	DSP	DSP (%)	FF	FF (%)	LUT	LUT (%)
144	3	2964	32	344814	13	279029	21

Streaming (multi compute) - Results





EVEREST

EVEREST

DESIGN ENVIRONMENT
FOR EXTREME-SCALE BIG DATA ANALYTICS
ON HETEROGENEOUS PLATFORMS

Thank you!

CHRISTIAN PILATO

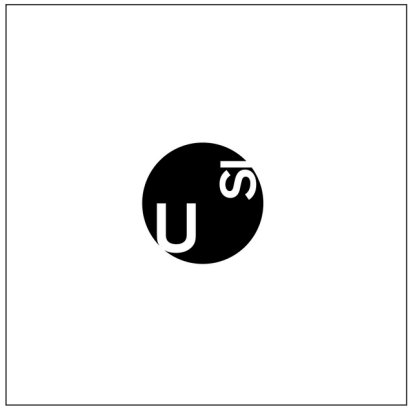
EVEREST Scientific Coordinator
Assistant Professor, Politecnico di Milano

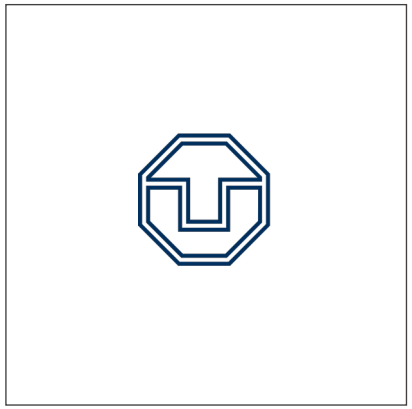
christian.pilato@polimi.it

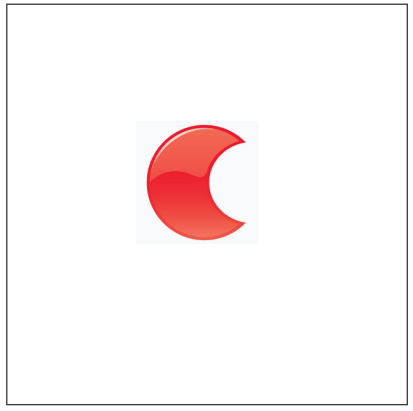
STEPHANIE SOLDAVINI

PhD Student, Politecnico di Milano

stephanie.soldavini@polimi.it







This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957269