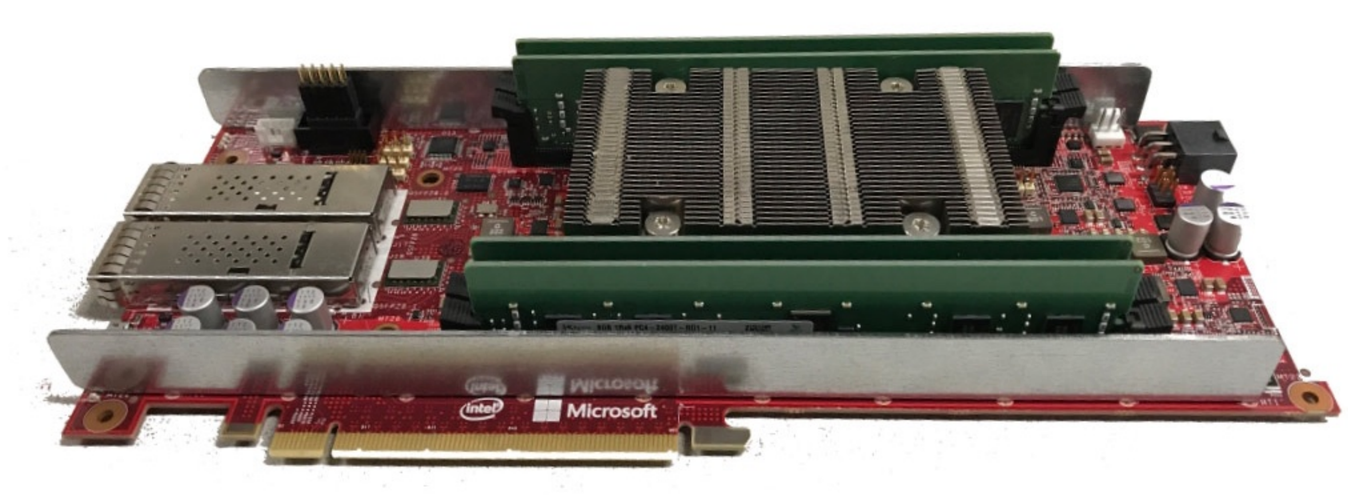
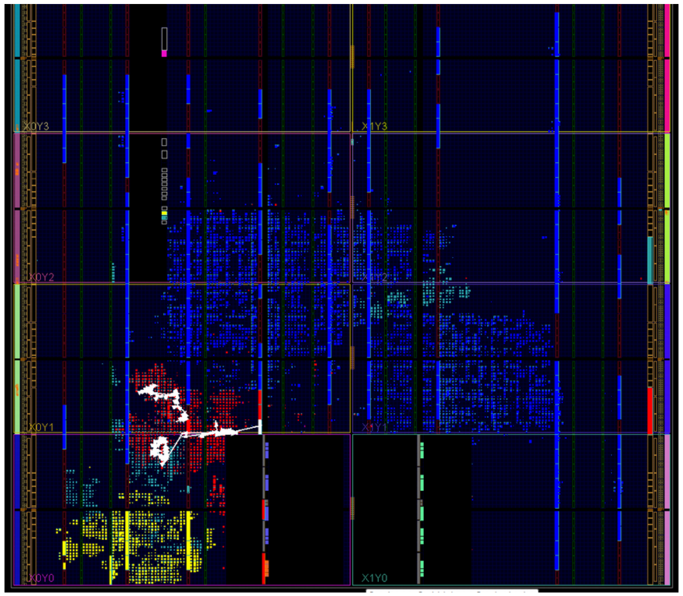


Firmware Implementation and SDAccel

Dylan Rankin (MIT)

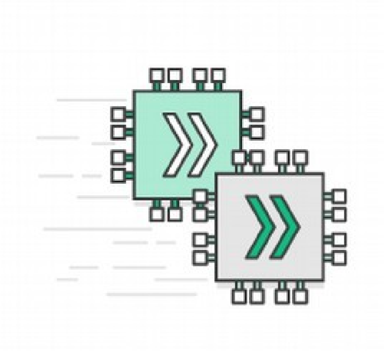


FPGA4HEP

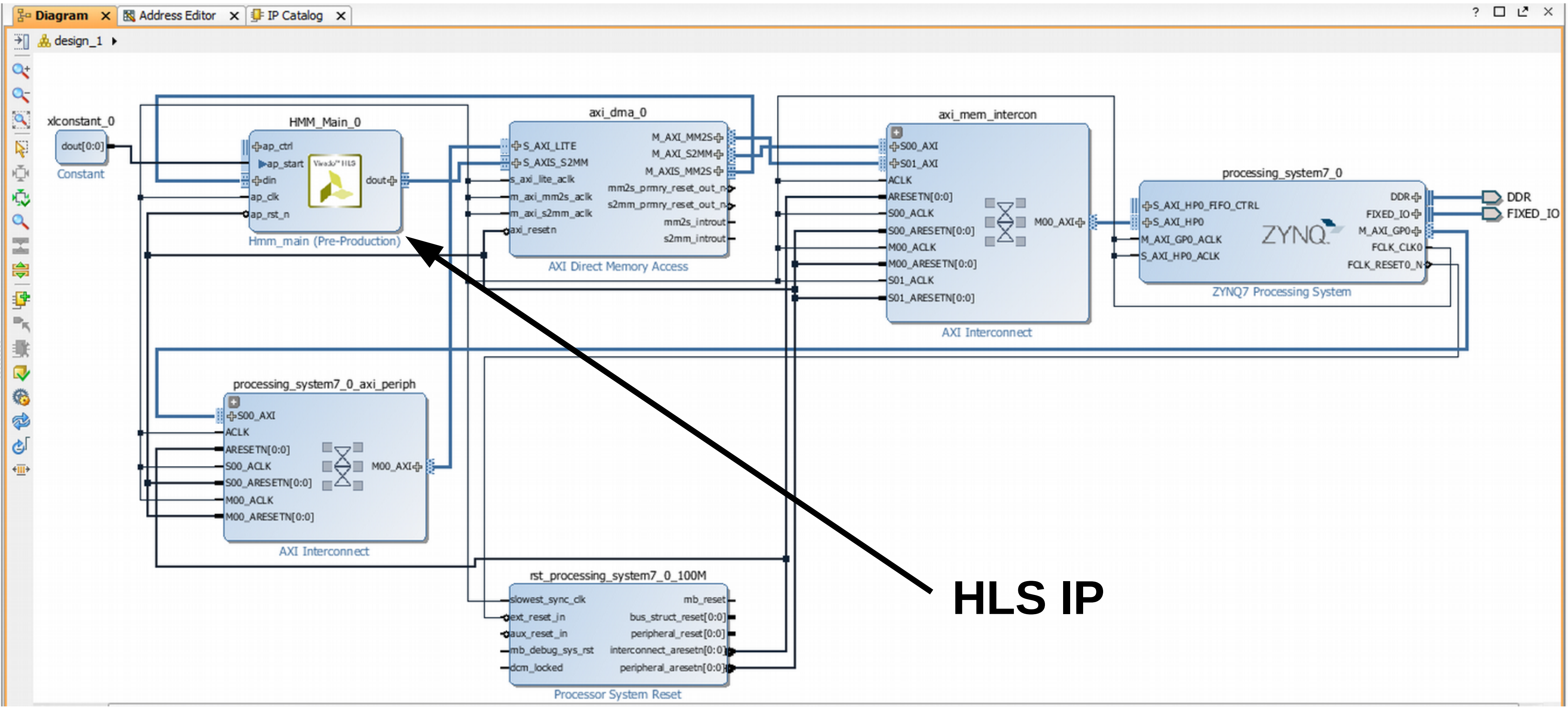
February 6th, 2018

Introduction

- HLS project is just one part of actually running an application on an FPGA
  - Need to handle data in to/out of FPGA, how to actually route signals through FPGA, etc.
- SDAccel is a tool that helps in development/implementing designs on FPGAs specifically for acceleration (CPU ↔ FPGA)
- Will show today how to use SDAccel to accelerate an hls4ml project

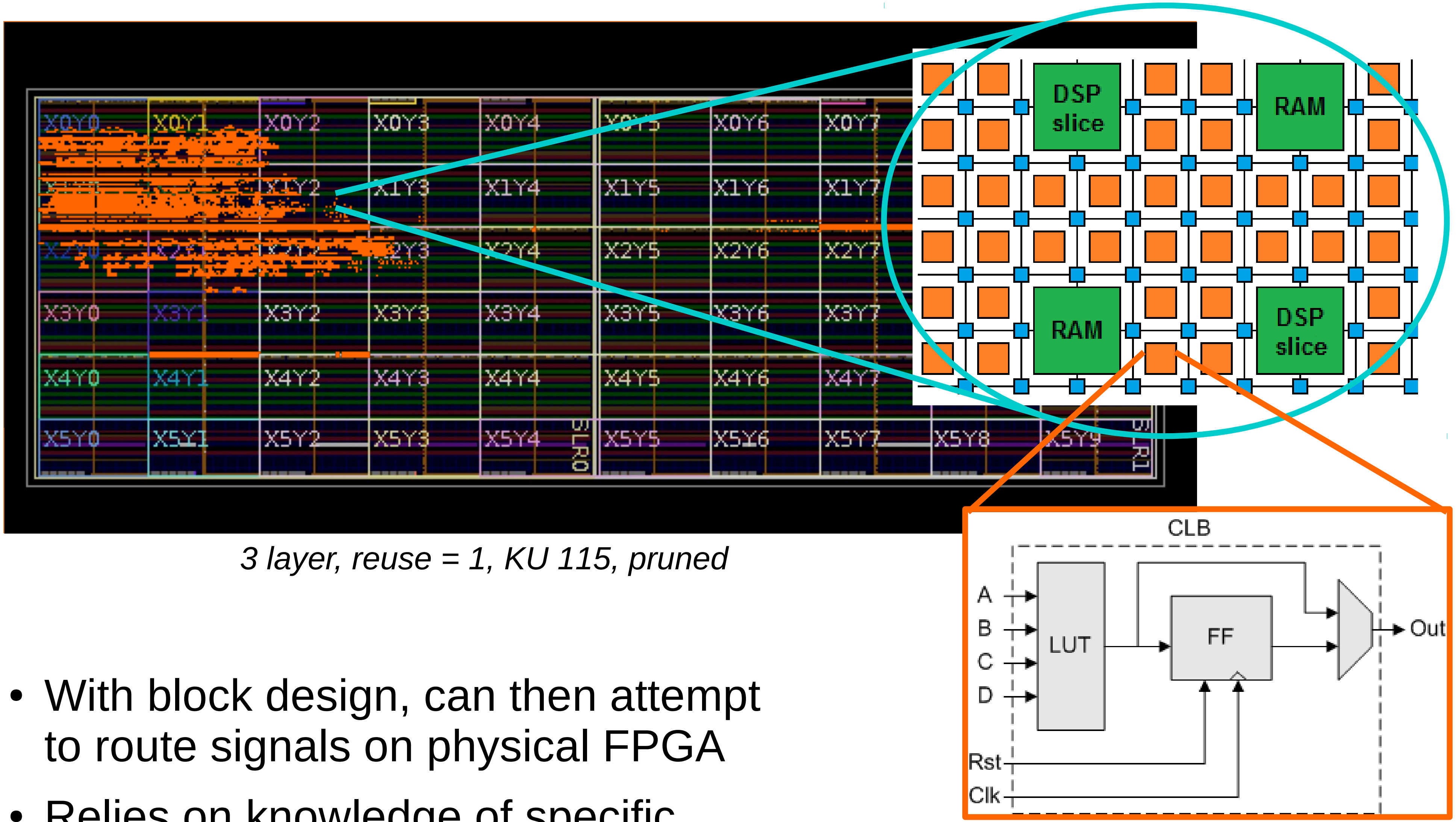


Programming an FPGA



- Typically need to specify how signals will be sent through FPGA

Programming an FPGA



3 layer, reuse = 1, KU 115, pruned

- With block design, can then attempt to route signals on physical FPGA
- Relies on knowledge of specific device, layout of components

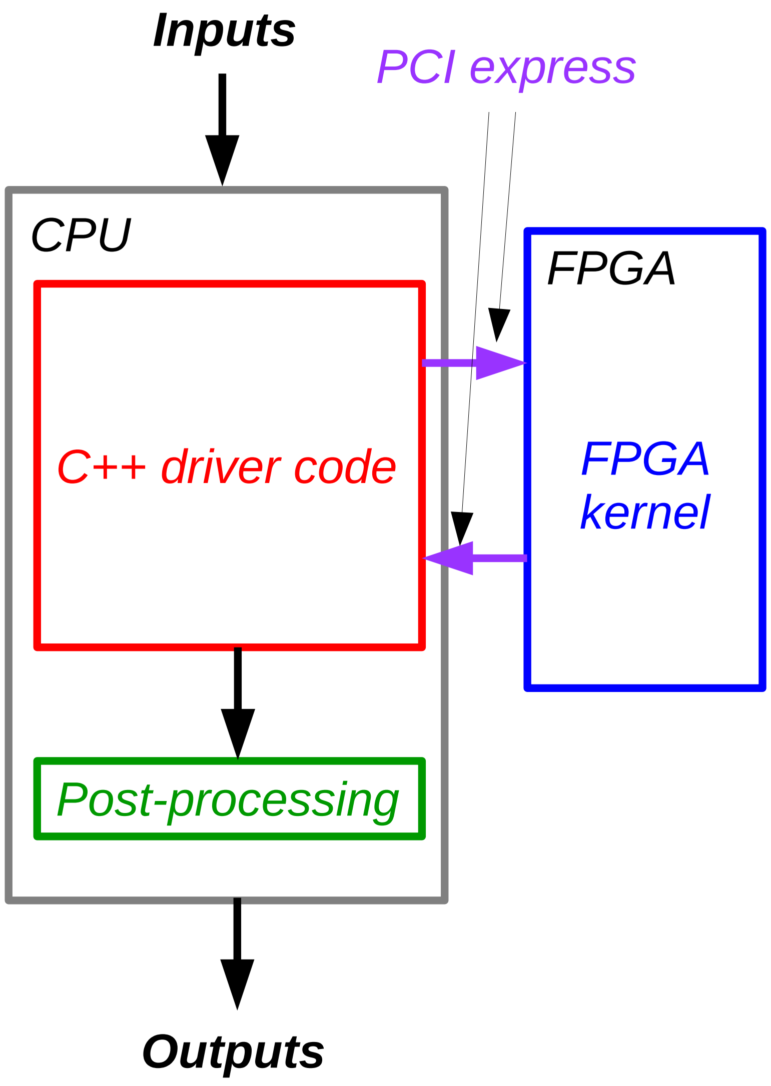
aws Overview

- AWS F1 instances are machines connected directly to a Xilinx Virtex UltraScale+ FPGA (VU9P) using PCI-express
- Will use cheaper general computing AWS instance (T2) instance to develop applications
  - F1 cost: $1.65/hr
  - T2 cost: $0.37/hr
- General application development on AWS done using SDAccel

Virtex Ultrascale+ VU9P
6800 DSPs
1M LUTs
2M FFs
75 Mb BRAM

SDAccel™ Environment

- SDAccel Development Environment allows the development/running of connected FPGA kernels and CPU processes
- Define FPGA kernel using HLS, OpenCL, or VHDL/Verilog
  - Need to meet certain design constraints regarding control, input/output protocol
- Any FPGA application defined in one of these languages can be easily accelerated using SDAccel
- Once FPGA kernel is available, write host code to run on CPU and manage data transfer, FPGA execution
  - Examples: https://github.com/Xilinx/SDAccel_Examples



STOP

Start T2 instances

Set up: **README**
Do: *Check out SDAccel and setup environment*

Setup

- What have we done?
- SDAccel has been set up:
  `source setup_sdaccel.sh`
- hls4ml wrapper (`hls4ml_c` directory) was already created
- This area will allow us to accelerate any hls4ml project

```
[centos@ip-172-31-36-147 hls4ml_c]$ ls *
Makefile  README.md  awsver.txt  check.sh  config.tcl  create.sh
description.json

src:
aws_hls4ml.cpp  host.cpp  kernel_params.h
```

SDAccel Makefile

- SDAccel uses make commands to build executables
  - xcpp for the CPU host code (Xilinx C++ compiler)
  - xocc for the FPGA kernel (Xilinx OpenCL Compiler)
    - xocc will run Vivado HLS under the hood

```
# Host Application
host_SRCS=./src/host.cpp $(xcl2_SRCS)
host_HDRS=$(xcl2_HDRS)
host_CXXFLAGS=-I./src/ -I$(HLS4ML_BASE)/nnet_utils/ -I$(HLS4ML_BASE)/keras-to-hls/$
(HLS4ML_PROJECT)/firmware/ $(xcl2_CXXFLAGS) $(opencl_CXXFLAGS) -DIS_$
(HLS4ML_PROJ_TYPE) -DHLS4ML_DATA_DIR=$(HLS4ML_BASE)/keras-to-hls/$
(HLS4ML_PROJECT)/tb_data/ -std=c++11
host_LDFLAGS=$(opencl_LDFLAGS) -I$(XILINX_VIVADO)/include/ -I$
(XILINX_SDACCEL)/include/ -Wno-unknown-pragmas

# aws_hls4ml Kernels
aws_hls4ml_SRCS=./src/aws_hls4ml.cpp $(HLS4ML_BASE)/keras-to-hls/$
(HLS4ML_PROJECT)/firmware/$(HLS4ML_NAME).cpp
aws_hls4ml_CLFLAGS=-k aws_hls4ml -DMYPROJ=$(HLS4ML_NAME) -DIS_$(HLS4ML_PROJ_TYPE)
-I./src/ -I$(HLS4ML_BASE)/keras-to-hls/$(HLS4ML_PROJECT)/firmware/ -I$
(HLS4ML_BASE)/keras-to-hls/$(HLS4ML_PROJECT)/firmware/weights -I$
(HLS4ML_BASE)/nnet_utils/ --xp "prop:solution.hls_pre_tcl=./config.tcl"
```

SDAccel Makefile

- SDAccel uses make commands to build executables
  - xcpp for the CPU host code (Xilinx C++ compiler)
  - xocc for the FPGA kernel (Xilinx OpenCL Compiler)
    - xocc will run Vivado HLS under the hood
- Three main running modes (all compile host code):
- Software Emulation : **`sw_emu`**
  - Emulate kernel in software (checks for C errors, ~csim)
- Hardware Emulation : **`hw_emu`**
  - Emulate kernel in hardware (builds kernel, ~cosim)
- Hardware : **`hw`**
  - Create xclbin (system image file, contains bitstream with kernel for programming FPGA, ~implementation/routing/bitstream)

STOP

Makefile: edit to point to your favorite model

```
#--v--v--
#these need to be set by the user for their specific installation
HLS4ML_BASE := /home/centos/fpga4hep/hls4ml
HLS4ML_PROJECT := my-hls-test-3layer
HLS4ML_NAME := myproject
HLS4ML_PROJ_TYPE := DENSE
#possible options are: DENSE, CONV1D
#--^--^--
```

Run:

```
make clean
make check TARGETS=sw_emu DEVICES=$AWS_PLATFORM all
```

STOP

Makefile: edit to point to the 1 layer MLP

```
#--v--v--
#these need to be set by the user for their specific installation
HLS4ML_BASE := /home/centos/fpga4hep/hls4ml
HLS4ML_PROJECT := my-hls-test-1layer
HLS4ML_NAME := myproject
HLS4ML_PROJ_TYPE := DENSE
#possible options are: DENSE, CONV1D
#--^--^--
```

Run:

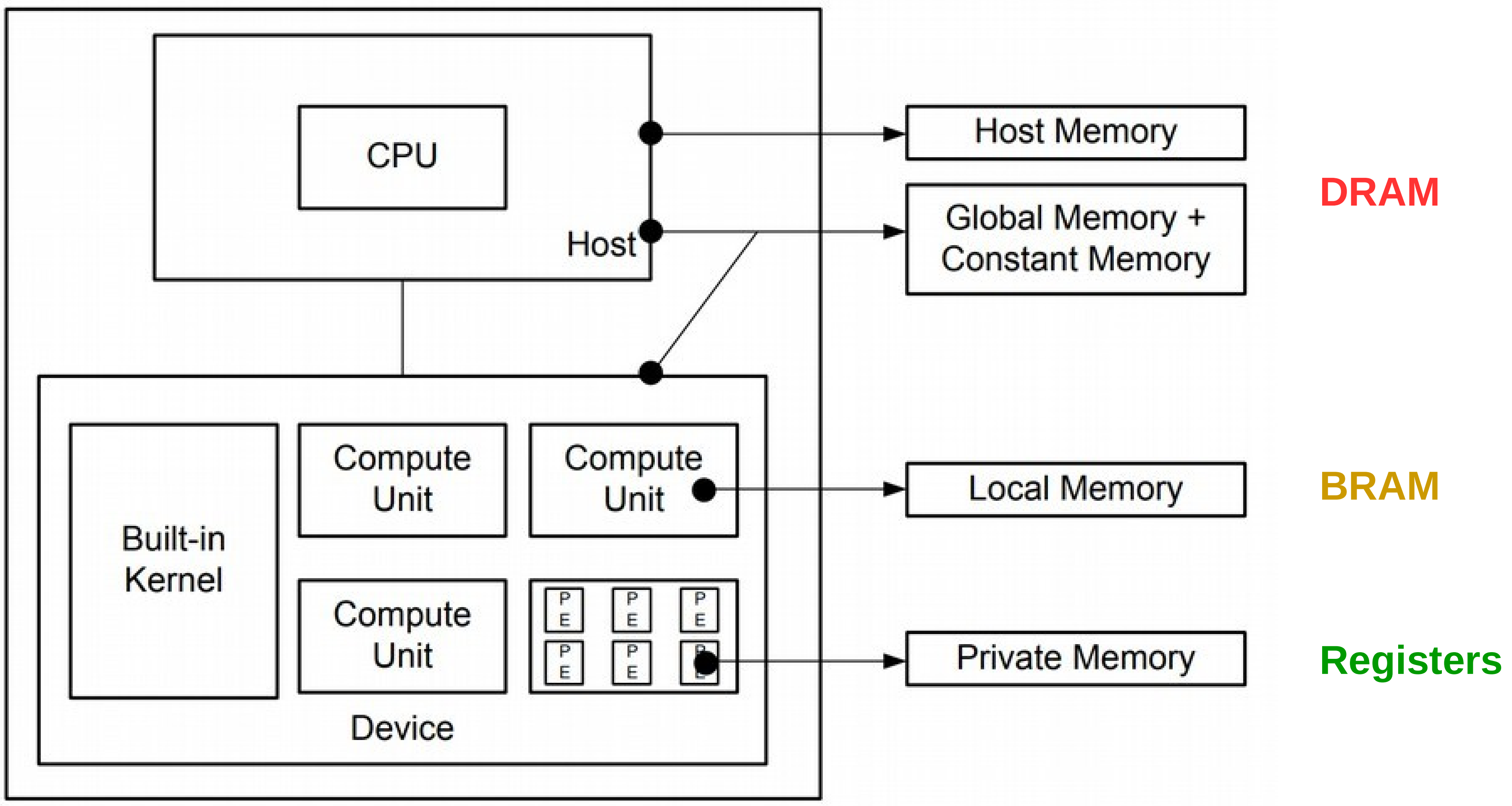
```
make clean
nohup make check TARGETS=hw_emu DEVICES=$AWS_PLATFORM all >& out.log &
```

FPGA Kernel

- SDAccel kernels require inputs and outputs to be passed in certain manner
  - Must be AXI-stream
  - Must be mapped to global memory
  - Specific control

```
extern "C" {
void aws_hls4ml(
        const data_t *in, // Read-Only Vector
        data_t *out       // Output Result
        )
{
#pragma HLS INTERFACE m_axi port=in  offset=slave bundle=gmem
#pragma HLS INTERFACE m_axi port=out offset=slave bundle=gmem
#pragma HLS INTERFACE s_axilite port=in   bundle=control
#pragma HLS INTERFACE s_axilite port=out  bundle=control
#pragma HLS INTERFACE s_axilite port=return bundle=control
```

SDAccel Data Management



- Transfer between CPU and FPGA *must* use DRAM

FPGA Kernel

```
    unsigned short insize, outsize; //necessary for hls4ml kernel, not used

    input_t in_buf[STREAMSIZE][N_INPUTS];
    result_t out_buf[STREAMSIZE][N_OUTPUTS]; //these will get partitioned properly in
the hls4ml code

//getting data from axi stream and formatting properly
    for (int i = 0; i < STREAMSIZE; i++) {
#pragma HLS LOOP UNROLL
        for (int j = 0; j < N_INPUTS; j++) {
#pragma HLS LOOP UNROLL
            in_buf[i][j] = (input_t)in[i*N_INPUTS+j];
        }
    }

//run inference
    for (int i = 0; i < STREAMSIZE; i++) {
#pragma HLS dataflow
        Hls4ml: myproject(in_buf[i],out_buf[i],insize,outsize);
    }

//place output into axi stream output
    for (int i = 0; i < STREAMSIZE; i++) {
#pragma HLS LOOP UNROLL
        for (int j = 0; j < N_OUTPUTS; j++) {
#pragma HLS LOOP UNROLL
            out[i*N_OUTPUTS+j] = (data_t)out_buf[i][j];
        }
    }
```

Host Code

- Need to allocate block in memory for data
- Pass information to device with OpenCL buffer objects
  - `cl::Buffer`

```
size_t vector_size_in_bytes = sizeof(data_t) * DATA_SIZE_IN * STREAMSIZE;
size_t vector_size_out_bytes = sizeof(data_t) * DATA_SIZE_OUT * STREAMSIZE;
std::vector<data_t,aligned_allocator<data_t>> source_in(DATA_SIZE_IN*STREAMSIZE);
std::vector<data_t,aligned_allocator<data_t>> source_hw_results(DATA_SIZE_OUT*STREAMSIZE);

// OPENCL HOST CODE AREA START
// get_xil_devices() is a utility API which will find the xilinx
// platforms and will return list of devices connected to Xilinx platform
std::vector<cl::Device> devices = xcl::get_xil_devices();
cl::Device device = devices[0];
cl::Context context(device);
cl::CommandQueue q(context, device, CL_QUEUE_PROFILING_ENABLE);

// Allocate Buffer in Global Memory
// Buffers are allocated using CL_MEM_USE_HOST_PTR for efficient memory and
// Device-to-host communication
cl::Buffer buffer_in   (context,CL_MEM_USE_HOST_PTR | CL_MEM_READ_ONLY,
        vector_size_in_bytes, source_in.data());
cl::Buffer buffer_output(context,CL_MEM_USE_HOST_PTR | CL_MEM_WRITE_ONLY,
        vector_size_out_bytes, source_hw_results.data());
```

Host Code

- Specify binary file to use for programming FPGA
- Connect global memory buffers with kernel arguments

```
// find_binary_file() is a utility API which will search the xclbin file for
// targeted mode (sw_emu/hw_emu/hw) and for targeted platforms.
std::string binaryFile = xcl::find_binary_file(device_name,"aws_hls4ml");
// import_binary_file() ia a utility API which will load the binaryFile
// and will return Binaries.
cl::Program::Binaries bins = xcl::import_binary_file(binaryFile);
devices.resize(1);
cl::Program program(context, devices, bins);
std::vector<cl::Memory> inBufVec, outBufVec;
inBufVec.push_back(buffer_in);
outBufVec.push_back(buffer_output);
cl::Kernel krnl_aws_hls4ml(program,"aws_hls4ml");
int narg = 0;
krnl_aws_hls4ml.setArg(narg++, buffer_in);
krnl_aws_hls4ml.setArg(narg++, buffer_output);
```

Host Code

- To run:
- 1) Place inputs in allocated global memory
- 2) Launch kernel
- 3) Move outputs back from global memory when available

```
    // Copy input data to device global memory
    q.enqueueMigrateMemObjects(inBufVec,0/* 0 means from host*/);
    // Launch the Kernel
    // For HLS kernels global and local size is always (1,1,1). So, it
is recommended
    // to always use enqueueTask() for invoking HLS kernel
    q.enqueueTask(krnl_aws_hls4ml);
    // Copy Result from Device Global Memory to Host Local Memory
    q.enqueueMigrateMemObjects(outBufVec,CL_MIGRATE_MEM_OBJECT_HOST);
    // Check for any errors from the command queue
    q.finish();
```

STOP

Vivado HLS Synthesis Reports

- Can access standard Vivado HLS reports after `hw_emu/hw`
  - Exact location changes with SDAccel/AMI version
  - *FPGA HDK 1.5.0:* `_x/aws_hls4ml.hw_emu.xilinx_aws-vu9p-f1-04261818_dynamic_5_0/aws_hls4ml/aws_hls4ml/solution/syn/report/`**`myproject_csynth.rpt`**

```
================================================================
== Performance Estimates
================================================================
+ Timing (ns):
    * Summary:
    +--------+-------+---------+------------+
    |  Clock | Target| Estimated| Uncertainty|
    +--------+-------+---------+------------+
    |ap_clk  |   4.00|     2.920|        1.08|
    +--------+-------+---------+------------+


+ Latency (clock cycles):
    * Summary:
    +-----+-----+-----+-----+----------+
    |  Latency  |  Interval | Pipeline |
    | min | max | min | max |   Type   |
    +-----+-----+-----+-----+----------+
    |   24|   24|    1|    1| function |
    +-----+-----+-----+-----+----------+
```

Vivado HLS Synthesis Reports

- Can access standard Vivado HLS reports after `hw_emu/hw`
  - Exact location changes with SDAccel/AMI version
  - *FPGA HDK 1.5.0:* `_x/aws_hls4ml.hw_emu.xilinx_aws-vu9p-f1-04261818_dynamic_5_0/aws_hls4ml/aws_hls4ml/solution/syn/report/`**`myproject_csynth.rpt`**

```
================================================================
== Utilization Estimates
================================================================
* Summary:
+-----------------+---------+-------+---------+---------+-----+
|       Name      | BRAM_18K| DSP48E|    FF   |   LUT   | URAM|
+-----------------+---------+-------+---------+---------+-----+
|DSP              |        -|      -|        -|        -|    -|
|Expression       |        -|      -|        0|        6|    -|
|FIFO             |        -|      -|        -|        -|    -|
|Instance         |        -|    123|    14829|    65502|    -|
|Memory           |        -|      -|        -|        -|    -|
|Multiplexer      |        -|      -|        -|        9|    -|
|Register         |        -|      -|     6249|        -|    -|
+-----------------+---------+-------+---------+---------+-----+
|Total            |        0|    123|    21078|    65517|    0|
+-----------------+---------+-------+---------+---------+-----+
|Available        |     4320|   6840|  2364480|  1182240|  960|
+-----------------+---------+-------+---------+---------+-----+
|Utilization (%)  |        0|      1|    ~0   |        5|    0|
+-----------------+---------+-------+---------+---------+-----+
```

Vivado HLS Synthesis Reports

- Can access standard Vivado HLS reports after `hw_emu/hw`
  - Exact location changes with SDAccel/AMI version
  - *FPGA HDK 1.5.0:* `_x/aws_hls4ml.hw_emu.xilinx_aws-vu9p-f1-04261818_dynamic_5_0/aws_hls4ml/aws_hls4ml/solution/syn/report/`**`aws_hls4ml_csynth.rpt`**

```
================================================================
== Performance Estimates
================================================================
+ Timing (ns):
    * Summary:
    +--------+-------+----------+------------+
    |  Clock | Target| Estimated| Uncertainty|
    +--------+-------+----------+------------+
    |ap_clk  |   4.00|     2.920|        1.08|
    +--------+-------+----------+------------+

+ Latency (clock cycles):
    * Summary:
    +-----+-----+-----+-----+---------+
    |  Latency  |  Interval | Pipeline|
    | min | max | min | max |   Type  |
    +-----+-----+-----+-----+---------+
    |   58|   58|   58|   58|   none  |
    +-----+-----+-----+-----+---------+
```

Vivado HLS Synthesis Reports

- Can access standard Vivado HLS reports after `hw_emu/hw`
  - Exact location changes with SDAccel/AMI version
  - *FPGA HDK 1.5.0:* `_x/aws_hls4ml.hw_emu.xilinx_aws-vu9p-f1-04261818_dynamic_5_0/aws_hls4ml/aws_hls4ml/solution/syn/report/`**`aws_hls4ml_csynth.rpt`**

```
================================================================
== Utilization Estimates
================================================================
* Summary:
+-----------------+---------+-------+---------+---------+-----+
|       Name      | BRAM_18K| DSP48E|    FF   |   LUT   | URAM|
+-----------------+---------+-------+---------+---------+-----+
|DSP              |        -|      -|        -|        -|    -|
|Expression       |        -|      -|        -|        -|    -|
|FIFO             |        -|      -|        -|        -|    -|
|Instance         |        2|    123|    21791|    66490|    -|
|Memory           |        -|      -|        -|        -|    -|
|Multiplexer      |        -|      -|        -|      362|    -|
|Register         |        -|      -|      768|        -|    -|
+-----------------+---------+-------+---------+---------+-----+
|Total            |        2|    123|    22559|    66852|    0|
+-----------------+---------+-------+---------+---------+-----+
|Available        |     4320|   6840|  2364480|  1182240|  960|
+-----------------+---------+-------+---------+---------+-----+
|Utilization (%)  |    ~0   |      1|    ~0   |        5|    0|
+-----------------+---------+-------+---------+---------+-----+
```

STOP

Makefile: edit to point to the 1 layer MLP

```
#--v--v--
#these need to be set by the user for their specific installation
HLS4ML_BASE := /home/centos/fpga4hep/hls4ml
HLS4ML_PROJECT := my-hls-test-1layer
HLS4ML_NAME := myproject
HLS4ML_PROJ_TYPE := DENSE
#possible options are: DENSE, CONV1D
#--^--^--
```

Run:

```
make clean
nohup make TARGETS=hw DEVICES=$AWS_PLATFORM all >& out.log &
wait && ./create.sh
```

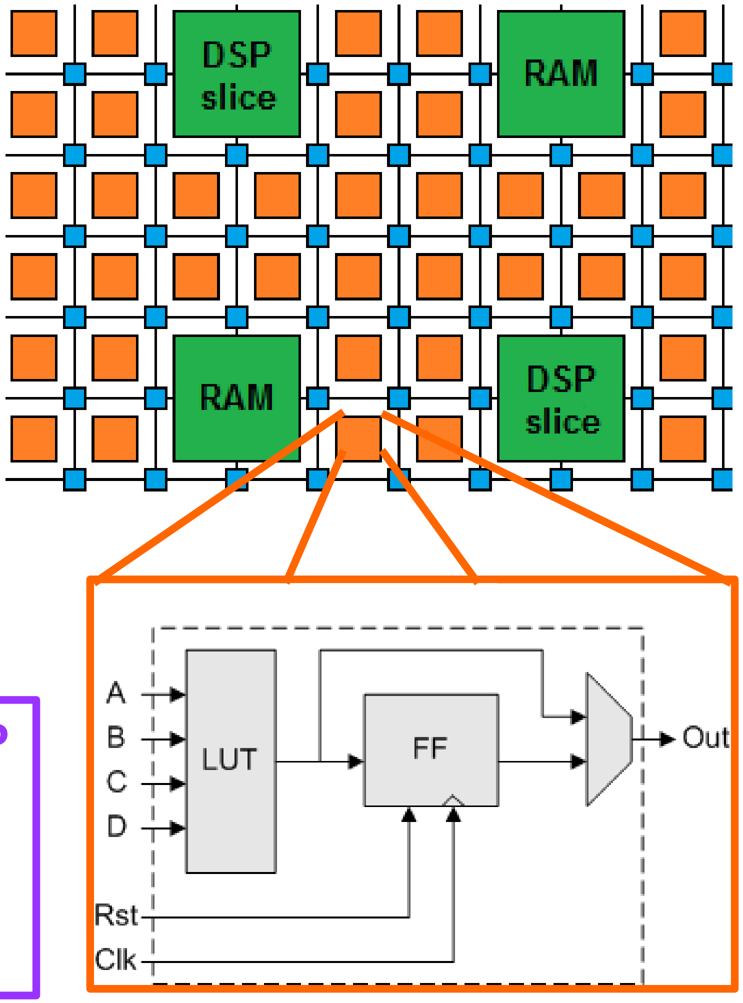
Summary

- Have used SDAccel to accelerate an hls4ml project
  - Used wrapper to handle inputs/outputs in manner necessary for SDAccel ✓
    - Global memory mapped AXI-stream
  - Used host code to pass known inputs to FPGA kernel ✓
- Analyzed HLS reports ✓
- Have compiled kernel and host code, ready to run on F1 ✓

BACKUP

What is an FPGA?

- Building blocks:
  - **Multiplier units (DSPs) [arithmetic]**
  - **Look Up Tables (LUTs) [logic]**
  - **Flip-flops (FFs) [registers]**
  - **Block RAMs (BRAMs) [memory]**
- Algorithms are wired onto the chip
- Run at high frequency – *hundreds of MHz*
  - Can compute outputs in O(ns)
- Programming traditionally done in Verilog/VHDL
  - Low-level hardware languages
- Possible to translate C to Verilog/VHDL using High Level Synthesis (HLS) tools



Virtex 7 XC7VX690T
3600 DSPs
400K LUTs
800K FFs
10 Mb BRAM

Virtex Ultrascale+ VU9P
6800 DSPs
1M LUTs
2M FFs
75 Mb BRAM

High Level Synthesis

- Transforms untimed **C/C++** into **RTL VHDL/Verilog**
  - Vivado HLS in this talk

```
module dut(rst, clk, q);
  input rst;
  input clk;
  output q;
  reg [7:0] c;

  always @ (posedge clk)
  begin
    if (rst == 1b’1) begin
      c <= 8'b00000000;
  end
  else begin
    c <= c + 1;
  end

  assign q = c;
endmodule
```

RTL Verilog

vs.

```
uint8 dut() {
  static uint8 c = 0;
  c+=1;
  return c;
}
```

Untimed C

Kernel Development

- Vivado HLS tool is already used by SDAccel
- HLS-based kernel for SDAccel must meet certain I/O specifications
  - Must be AXI (Advanced eXtensible Interface) memory-mapped (DDR)
    - AXI is a protocol for transferring data between different RTL blocks
  - Must have specific control port
- SDAccel automatically creates block diagram around HLS IP
  - Sets up AXI interconnect, DMA controller, etc.