# HLS

## Streaming_free_running_k2k

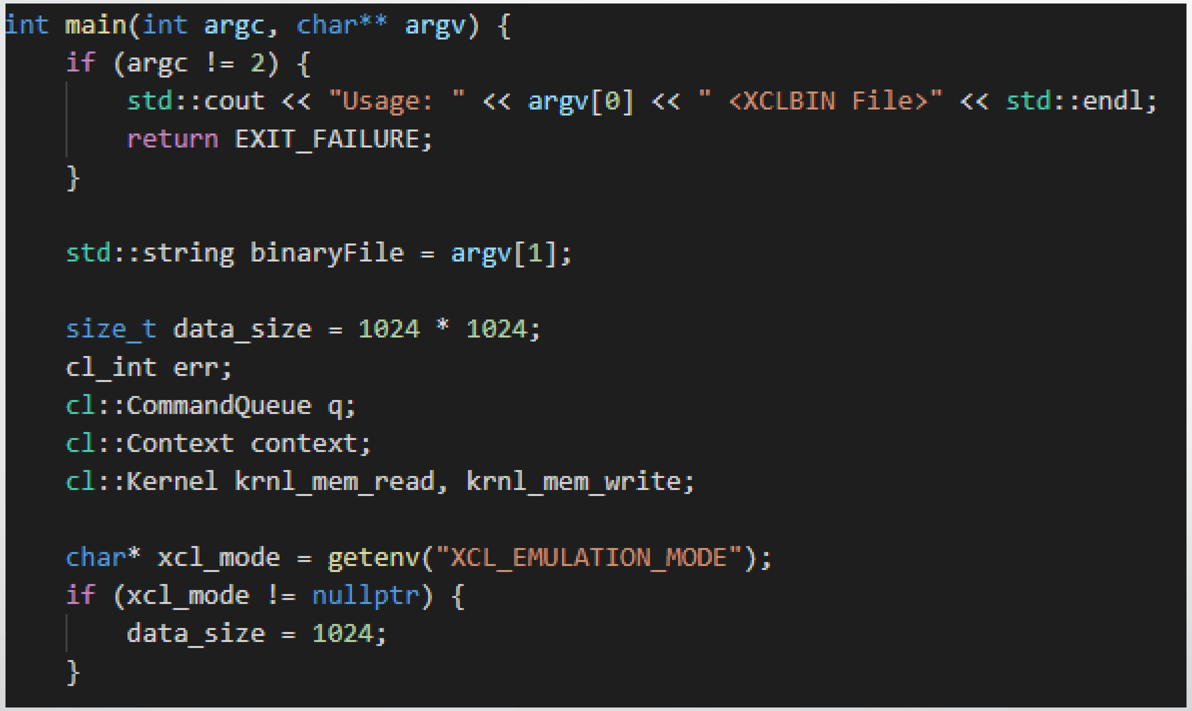
Tim Hsu

## Outline

- Streaming_free_running_k2k
    - Memory Read
    - Increment
    - Memory Write
    - Vitis_analyzer
- Streaming_k2k_mm
    - Read/Write Stream
    - Vitis_analyzer

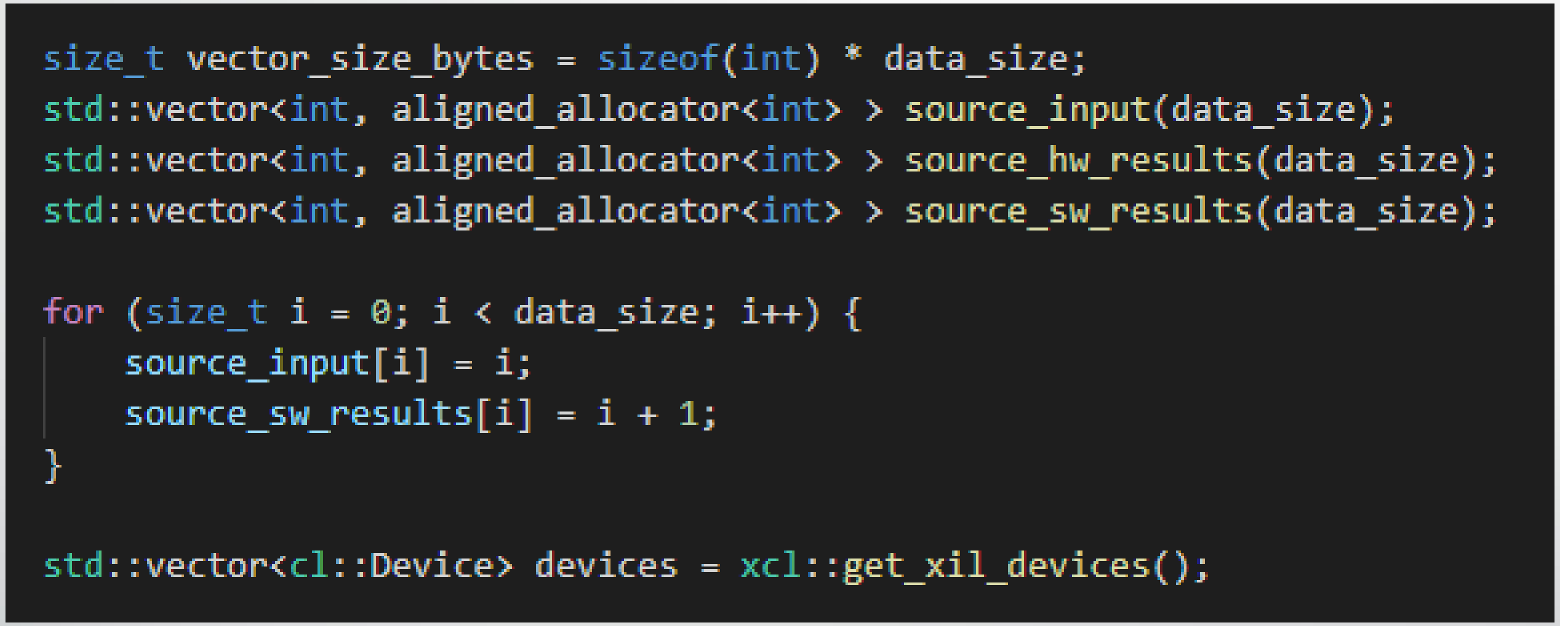
## Streaming_free_running_k2k

# Streaming_free_running_k2k
## Memory Read



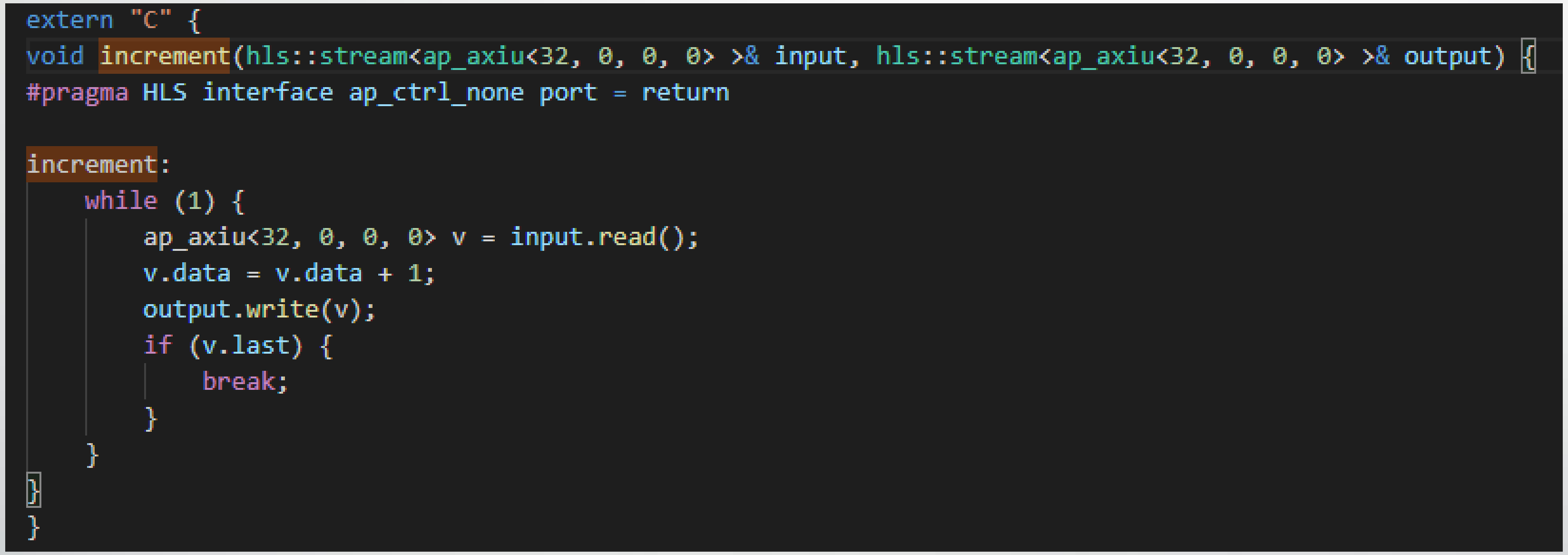
```cpp
int main(int argc, char** argv) {
    if (argc != 2) {
        std::cout << "Usage: " << argv[0] << " <XCLBIN File>" << std::endl;
        return EXIT_FAILURE;
    }

    std::string binaryFile = argv[1];

    size_t data_size = 1024 * 1024;
    cl_int err;
    cl::CommandQueue q;
    cl::Context context;
    cl::Kernel krnl_mem_read, krnl_mem_write;

    char* xcl_mode = getenv("XCL_EMULATION_MODE");
    if (xcl_mode != nullptr) {
        data_size = 1024;
    }
```

# Streaming_free_running_k2k Increment



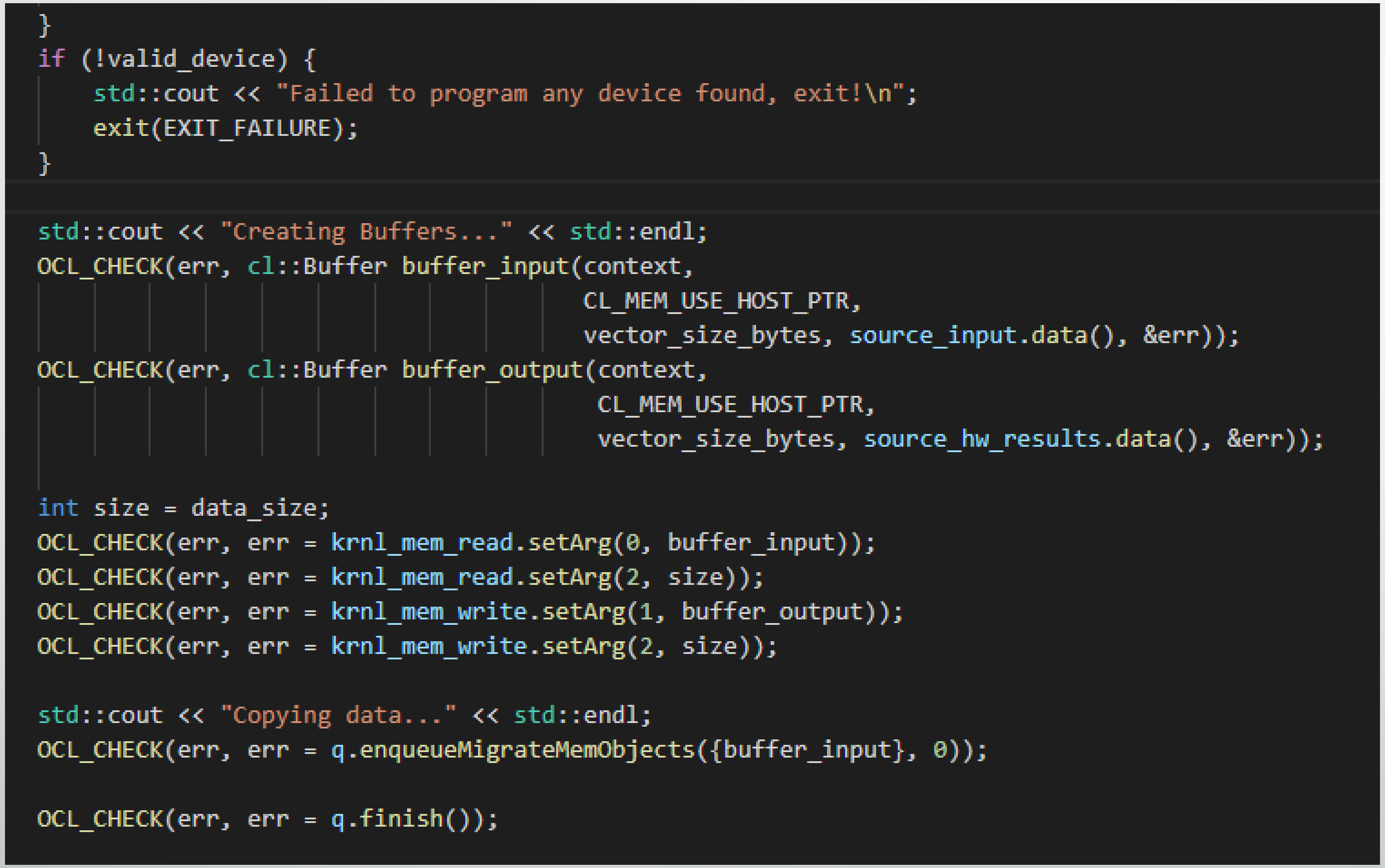
```
size_t vector_size_bytes = sizeof(int) * data_size;
std::vector<int, aligned_allocator<int> > source_input(data_size);
std::vector<int, aligned_allocator<int> > source_hw_results(data_size);
std::vector<int, aligned_allocator<int> > source_sw_results(data_size);

for (size_t i = 0; i < data_size; i++) {
    source_input[i] = i;
    source_sw_results[i] = i + 1;
}

std::vector<cl::Device> devices = xcl::get_xil_devices();
```

## Streaming_free_running_k2k Increment



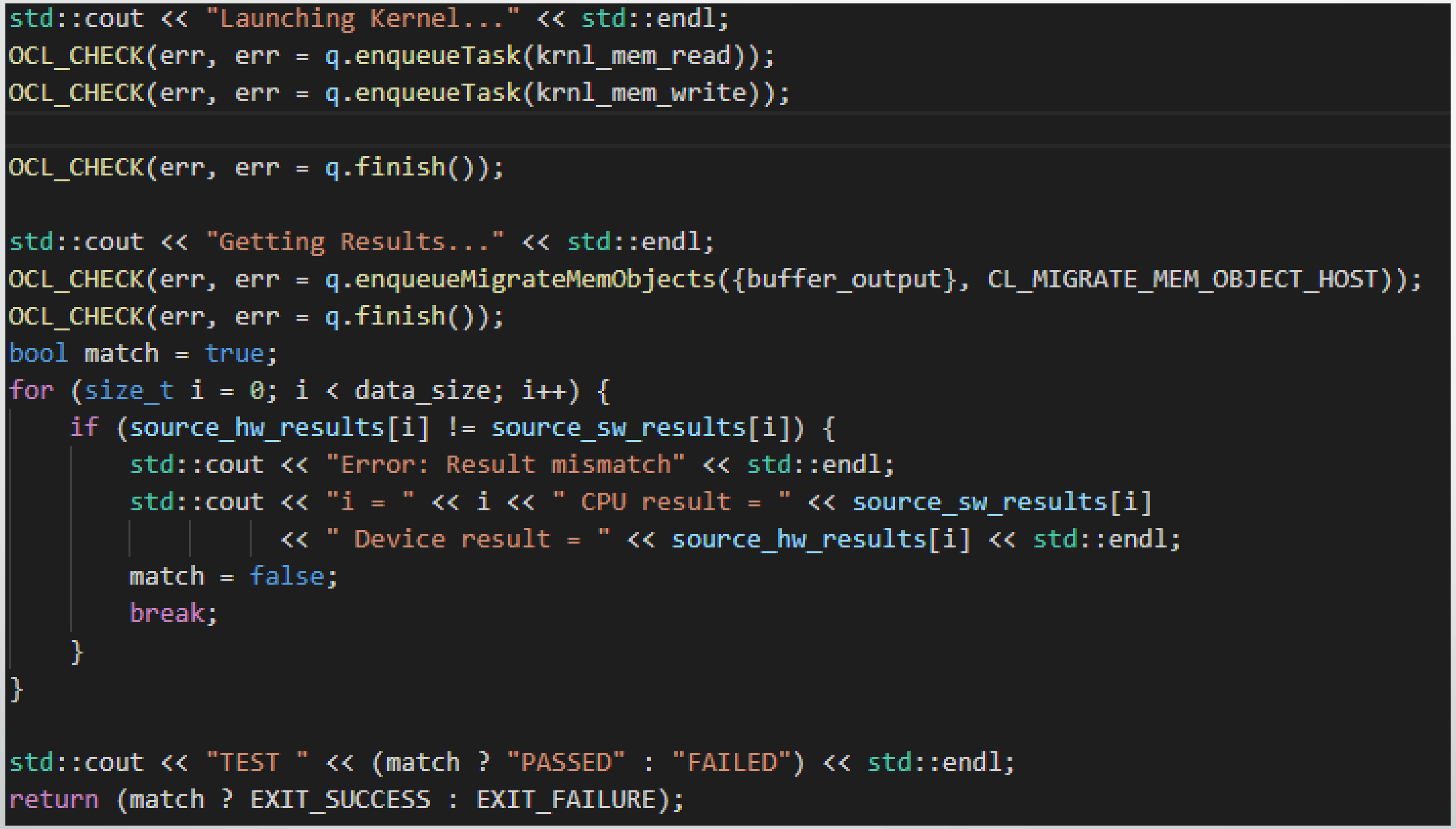
```
extern "C" {
void increment(hls::stream<ap_axiu<32, 0, 0, 0> >& input, hls::stream<ap_axiu<32, 0, 0, 0> >& output) {
#pragma HLS interface ap_ctrl_none port = return

increment:
    while (1) {
        ap_axiu<32, 0, 0, 0> v = input.read();
        v.data = v.data + 1;
        output.write(v);
        if (v.last) {
            break;
        }
    }
}
}
```

# Streaming_free_running_k2k
## Memory Write



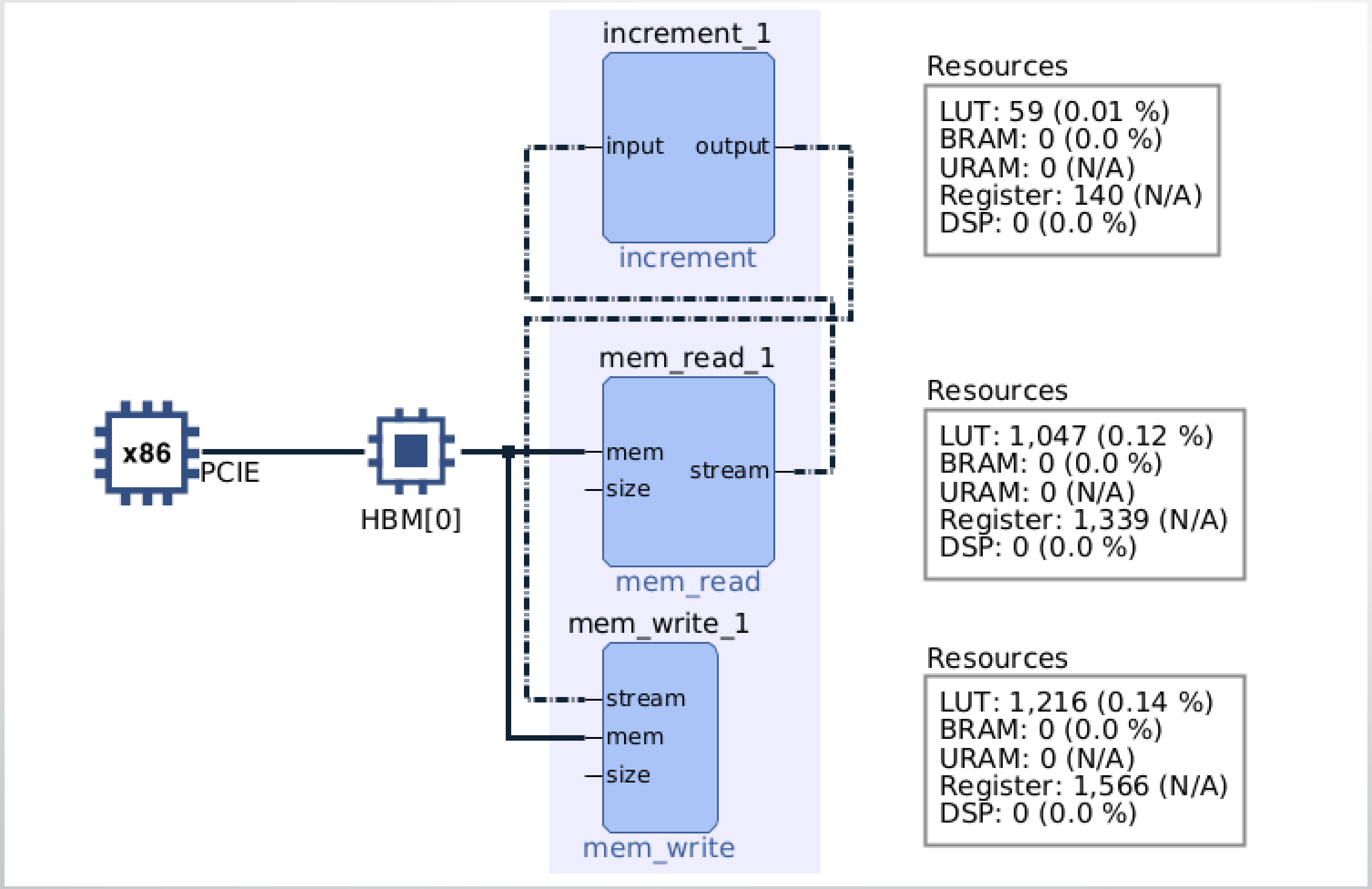
```
}
if (!valid_device) {
    std::cout << "Failed to program any device found, exit!\n";
    exit(EXIT_FAILURE);
}

std::cout << "Creating Buffers..." << std::endl;
OCL_CHECK(err, cl::Buffer buffer_input(context,
                                       CL_MEM_USE_HOST_PTR,
                                       vector_size_bytes, source_input.data(), &err));
OCL_CHECK(err, cl::Buffer buffer_output(context,
                                        CL_MEM_USE_HOST_PTR,
                                        vector_size_bytes, source_hw_results.data(), &err));

int size = data_size;
OCL_CHECK(err, err = krnl_mem_read.setArg(0, buffer_input));
OCL_CHECK(err, err = krnl_mem_read.setArg(2, size));
OCL_CHECK(err, err = krnl_mem_write.setArg(1, buffer_output));
OCL_CHECK(err, err = krnl_mem_write.setArg(2, size));

std::cout << "Copying data..." << std::endl;
OCL_CHECK(err, err = q.enqueueMigrateMemObjects({buffer_input}, 0));

OCL_CHECK(err, err = q.finish());
```

# Streaming_free_running_k2k
## Memory Write



```
std::cout << "Launching Kernel..." << std::endl;
OCL_CHECK(err, err = q.enqueueTask(krnl_mem_read));
OCL_CHECK(err, err = q.enqueueTask(krnl_mem_write));

OCL_CHECK(err, err = q.finish());

std::cout << "Getting Results..." << std::endl;
OCL_CHECK(err, err = q.enqueueMigrateMemObjects({buffer_output}, CL_MIGRATE_MEM_OBJECT_HOST));
OCL_CHECK(err, err = q.finish());
bool match = true;
for (size_t i = 0; i < data_size; i++) {
    if (source_hw_results[i] != source_sw_results[i]) {
        std::cout << "Error: Result mismatch" << std::endl;
        std::cout << "i = " << i << " CPU result = " << source_sw_results[i]
                  << " Device result = " << source_hw_results[i] << std::endl;
        match = false;
        break;
    }
}

std::cout << "TEST " << (match ? "PASSED" : "FAILED") << std::endl;
return (match ? EXIT_SUCCESS : EXIT_FAILURE);
```

# Streaming_free_running_k2k
## Vitis_analyzer



Resources

LUT: 59 (0.01 %)
BRAM: 0 (0.0 %)
URAM: 0 (N/A)
Register: 140 (N/A)
DSP: 0 (0.0 %)

Resources

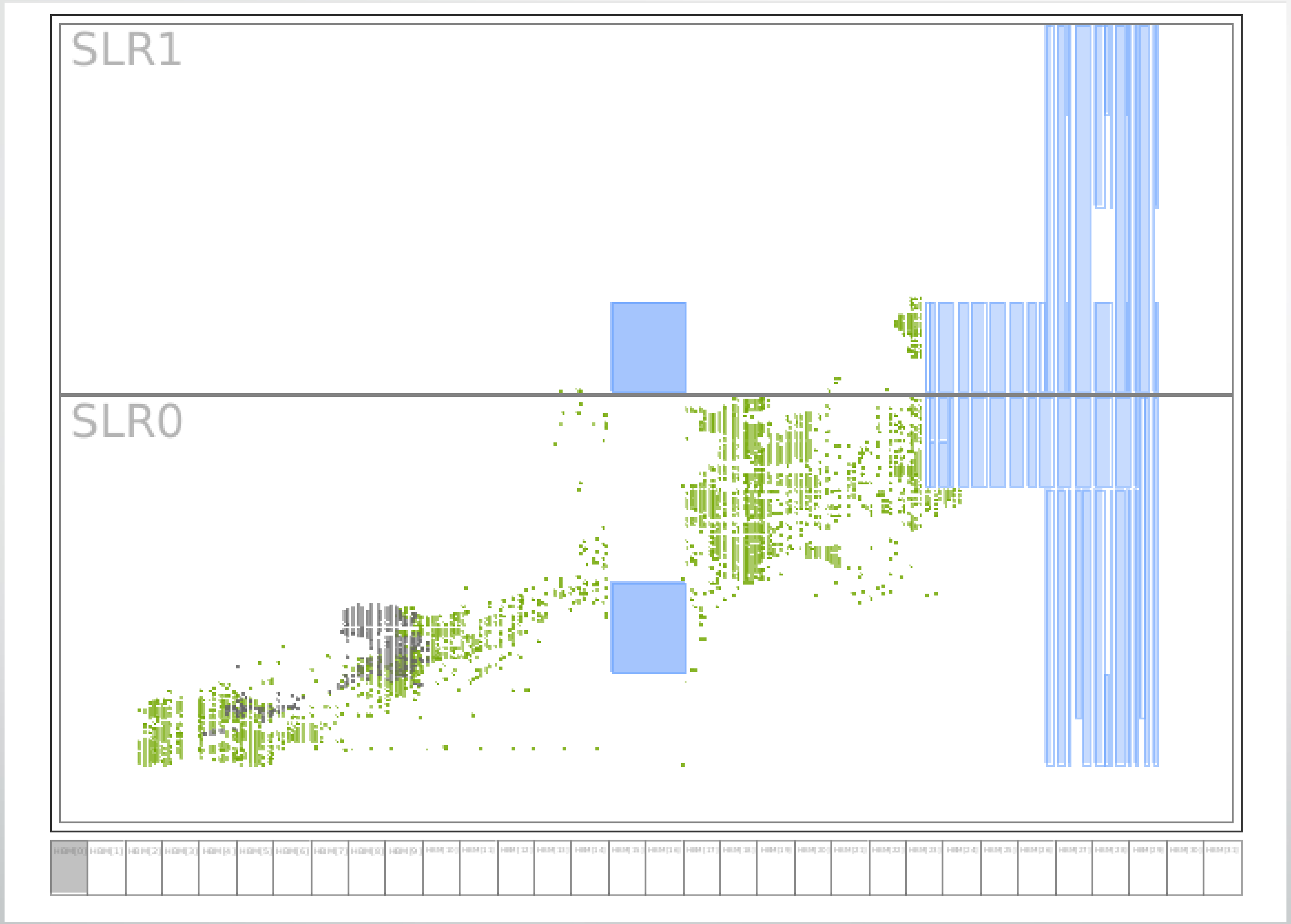
LUT: 1,047 (0.12 %)
BRAM: 0 (0.0 %)
URAM: 0 (N/A)
Register: 1,339 (N/A)
DSP: 0 (0.0 %)

Resources

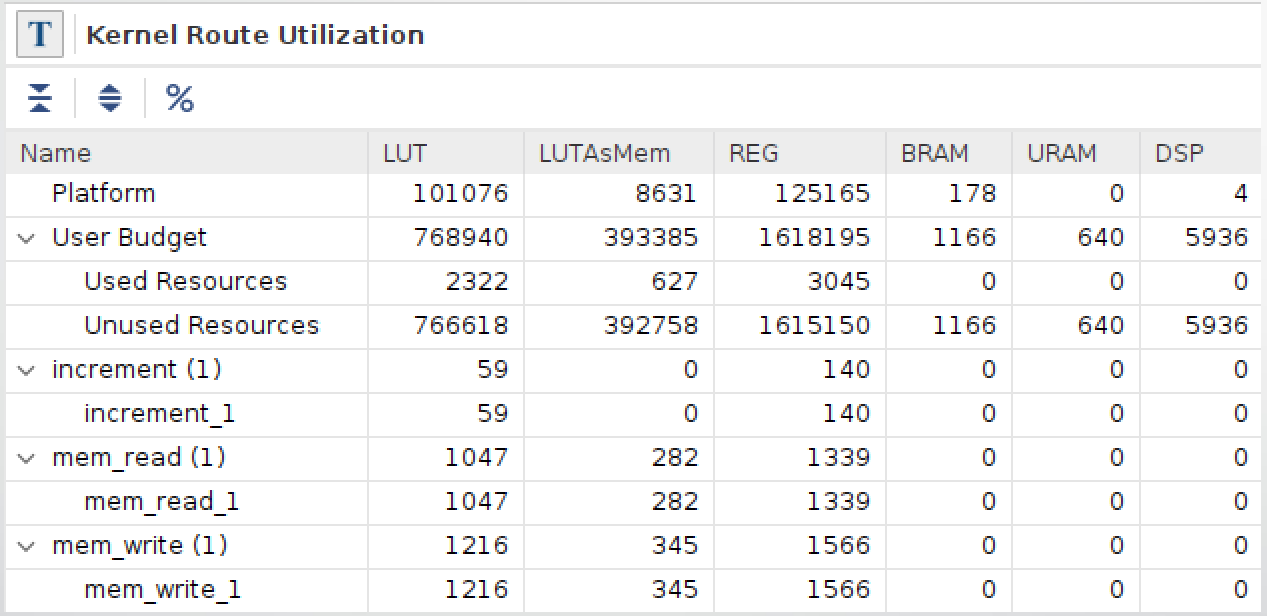
LUT: 1,216 (0.14 %)
BRAM: 0 (0.0 %)
URAM: 0 (N/A)
Register: 1,566 (N/A)
DSP: 0 (0.0 %)

# Streaming_free_running_k2k

## Vitis_analyzer



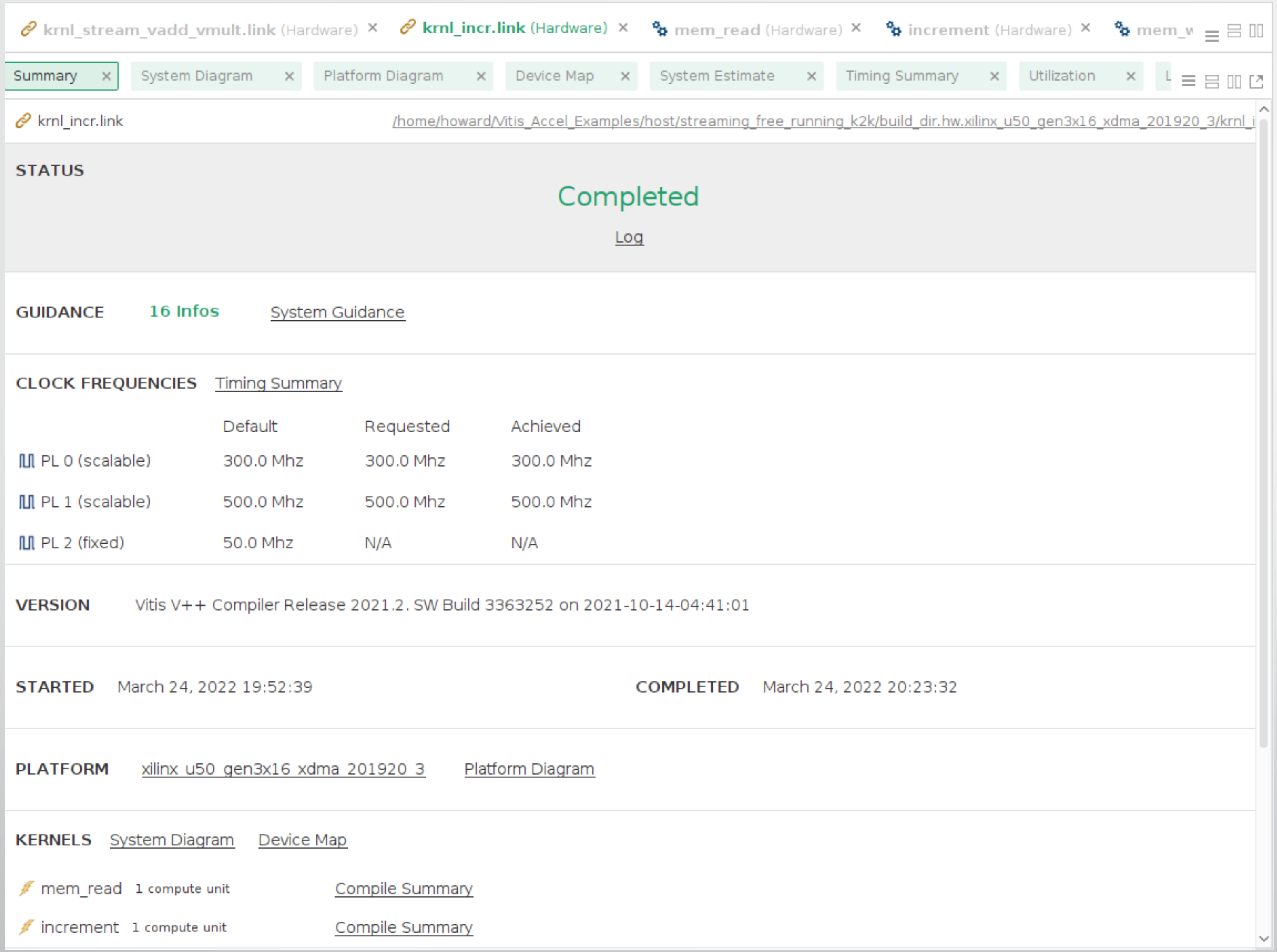
# Streaming_free_running_k2k
## Vitis_analyzer



**Kernel Route Utilization**

| Name | LUT | LUTAsMem | REG | BRAM | URAM | DSP |
|---|---|---|---|---|---|---|
| Platform | 101076 | 8631 | 125165 | 178 | 0 | 4 |
| User Budget | 768940 | 393385 | 1618195 | 1166 | 640 | 5936 |
| Used Resources | 2322 | 627 | 3045 | 0 | 0 | 0 |
| Unused Resources | 766618 | 392758 | 1615150 | 1166 | 640 | 5936 |
| increment (1) | 59 | 0 | 140 | 0 | 0 | 0 |
| increment_1 | 59 | 0 | 140 | 0 | 0 | 0 |
| mem_read (1) | 1047 | 282 | 1339 | 0 | 0 | 0 |
| mem_read_1 | 1047 | 282 | 1339 | 0 | 0 | 0 |
| mem_write (1) | 1216 | 345 | 1566 | 0 | 0 | 0 |
| mem_write_1 | 1216 | 345 | 1566 | 0 | 0 | 0 |

# Streaming_free_running_k2k
## Vitis_analyzer



krnl_stream_vadd_vmult.link (Hardware) | krnl_incr.link (Hardware) | mem_read (Hardware) | increment (Hardware) | mem_w

Summary | System Diagram | Platform Diagram | Device Map | System Estimate | Timing Summary | Utilization

krnl_incr.link /home/howard/Vitis_Accel_Examples/host/streaming_free_running_k2k/build_dir.hw.xilinx_u50_gen3x16_xdma_201920_3/krnl_i

**STATUS**

Completed

Log

**GUIDANCE** 16 Infos System Guidance

**CLOCK FREQUENCIES** Timing Summary

| | Default | Requested | Achieved |
|---|---|---|---|
| PL 0 (scalable) | 300.0 Mhz | 300.0 Mhz | 300.0 Mhz |
| PL 1 (scalable) | 500.0 Mhz | 500.0 Mhz | 500.0 Mhz |
| PL 2 (fixed) | 50.0 Mhz | N/A | N/A |

**VERSION** Vitis V++ Compiler Release 2021.2. SW Build 3363252 on 2021-10-14-04:41:01

**STARTED** March 24, 2022 19:52:39 **COMPLETED** March 24, 2022 20:23:32

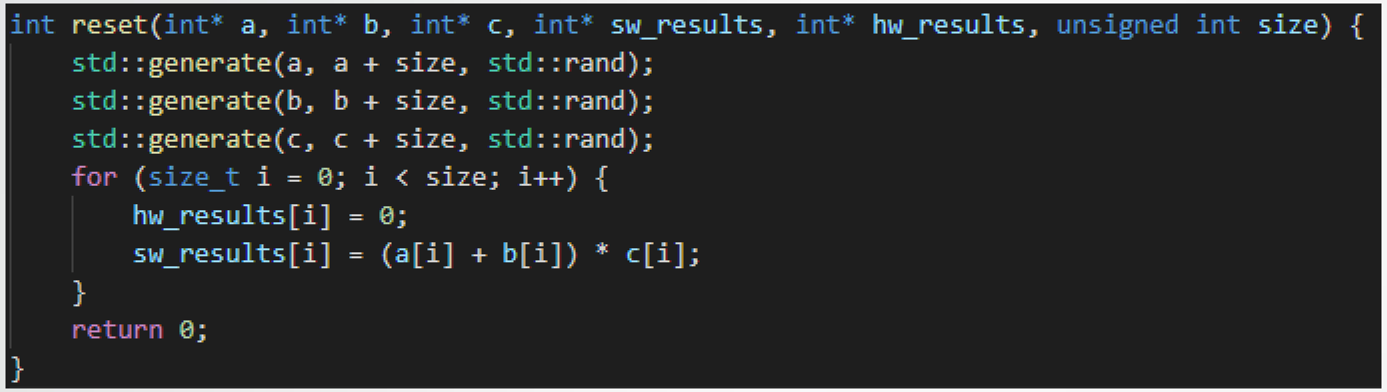
**PLATFORM** xilinx_u50_gen3x16_xdma_201920_3 Platform Diagram

**KERNELS** System Diagram Device Map

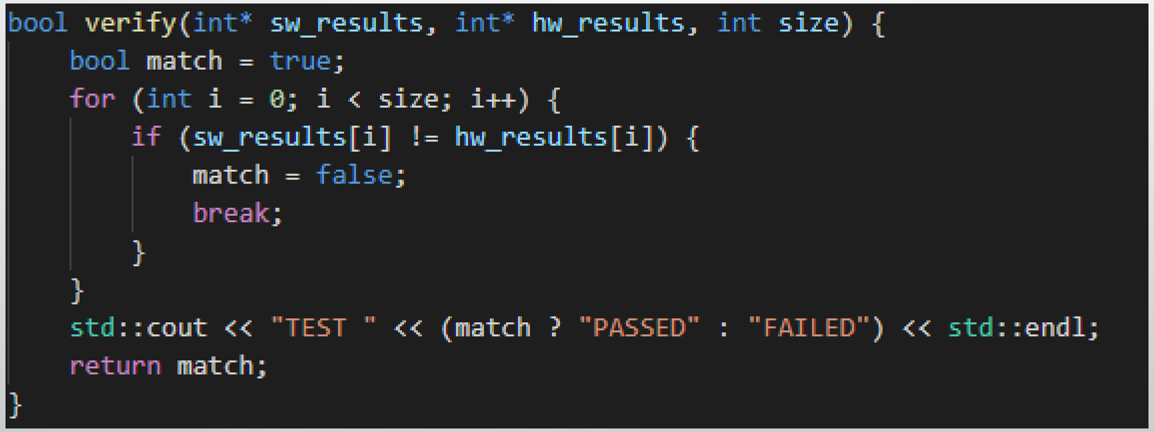
| | | |
|---|---|---|
| mem_read | 1 compute unit | Compile Summary |
| increment | 1 compute unit | Compile Summary |

## Streaming_free_k2k_mm

## Streaming_k2k_mm
### Read/Write Stream



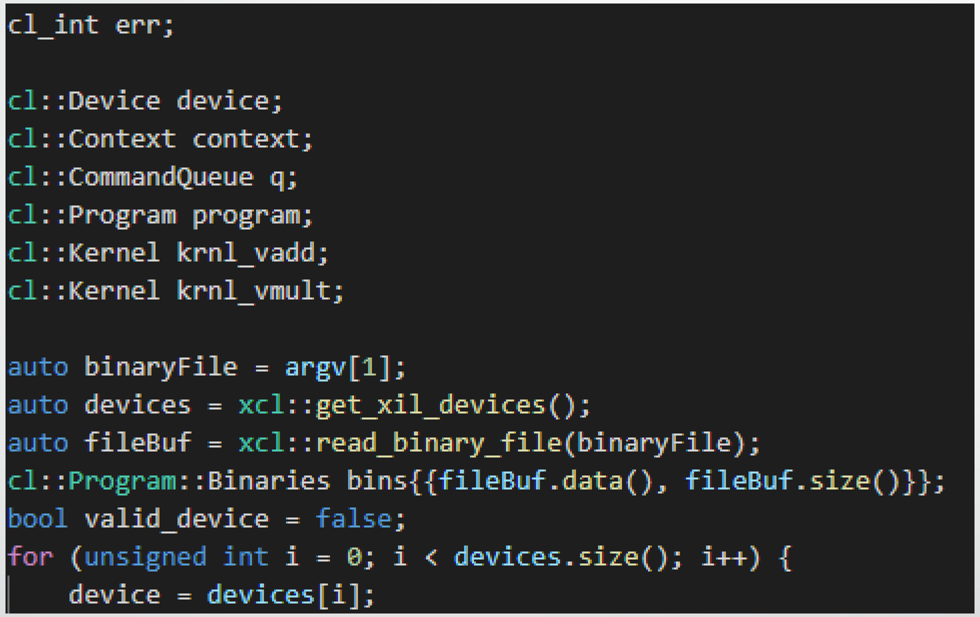
```
int reset(int* a, int* b, int* c, int* sw_results, int* hw_results, unsigned int size) {
    std::generate(a, a + size, std::rand);
    std::generate(b, b + size, std::rand);
    std::generate(c, c + size, std::rand);
    for (size_t i = 0; i < size; i++) {
        hw_results[i] = 0;
        sw_results[i] = (a[i] + b[i]) * c[i];
    }
    return 0;
}
```



```
bool verify(int* sw_results, int* hw_results, int size) {
    bool match = true;
    for (int i = 0; i < size; i++) {
        if (sw_results[i] != hw_results[i]) {
            match = false;
            break;
        }
    }
    std::cout << "TEST " << (match ? "PASSED" : "FAILED") << std::endl;
    return match;
}
```

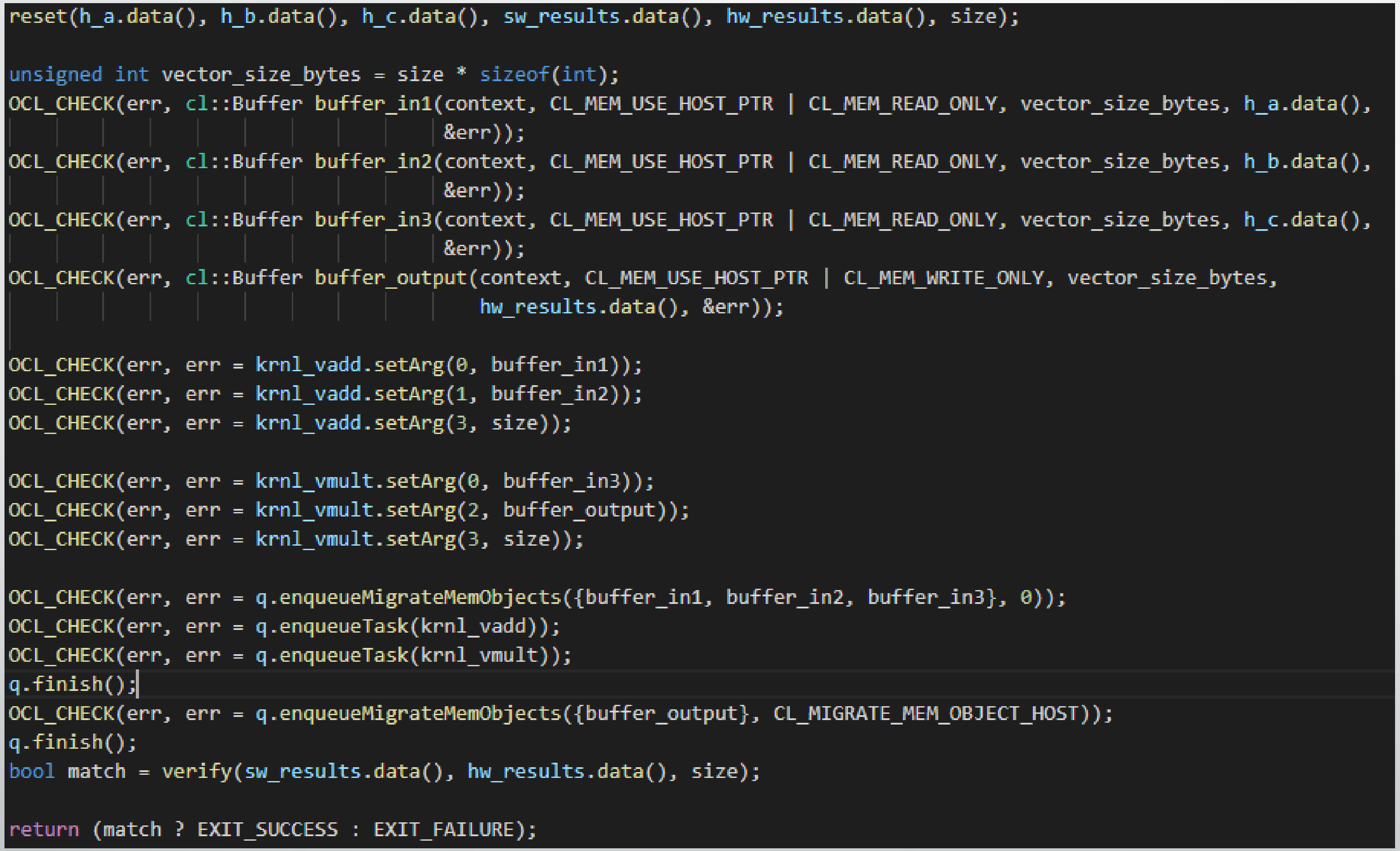
# Streaming_k2k_mm

## Read/Write Stream



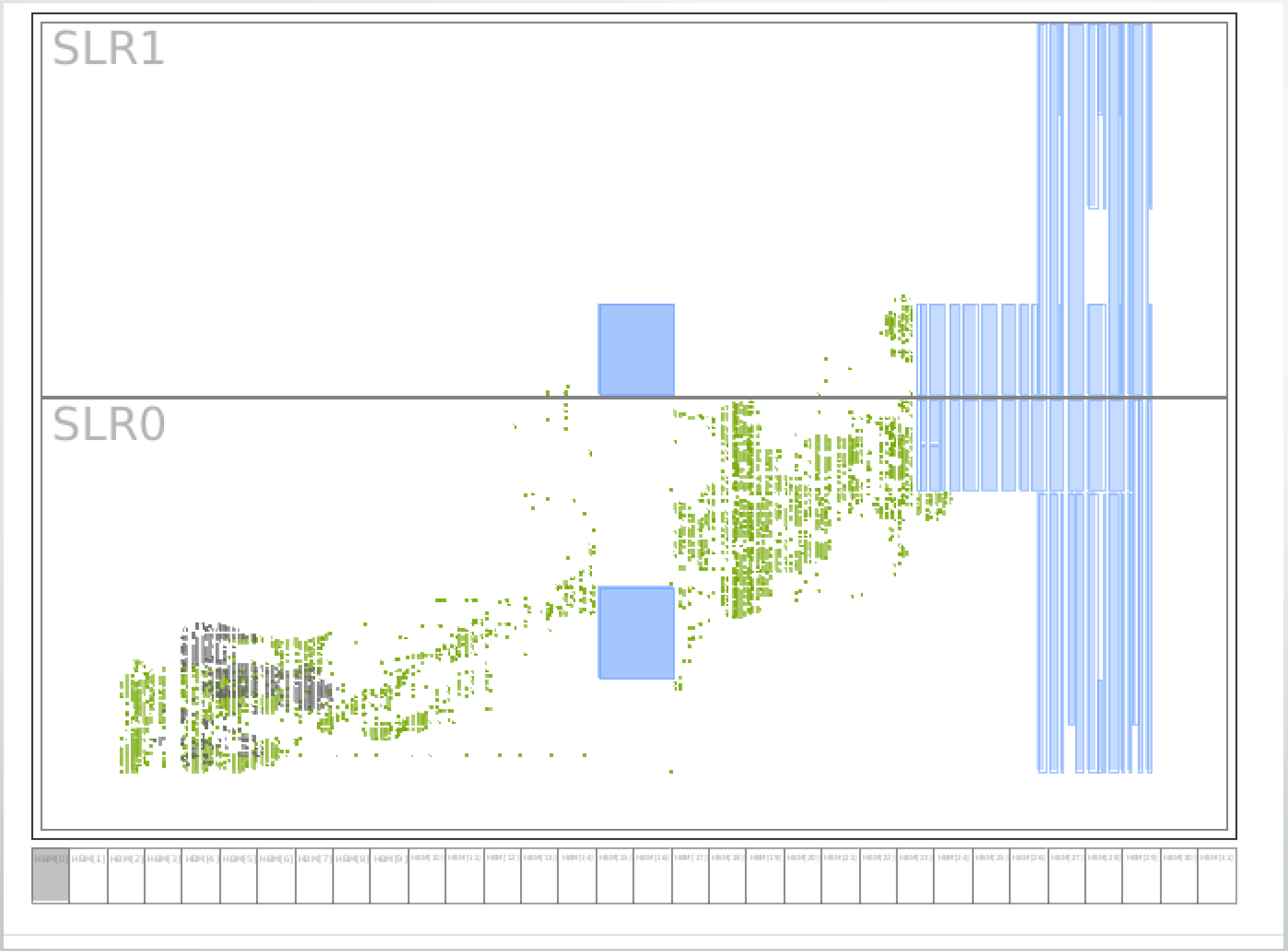
```
cl_int err;

cl::Device device;
cl::Context context;
cl::CommandQueue q;
cl::Program program;
cl::Kernel krnl_vadd;
cl::Kernel krnl_vmult;

auto binaryFile = argv[1];
auto devices = xcl::get_xil_devices();
auto fileBuf = xcl::read_binary_file(binaryFile);
cl::Program::Binaries bins{{fileBuf.data(), fileBuf.size()}};
bool valid_device = false;
for (unsigned int i = 0; i < devices.size(); i++) {
    device = devices[i];
```

# Streaming_k2k_mm
## Read/Write Stream

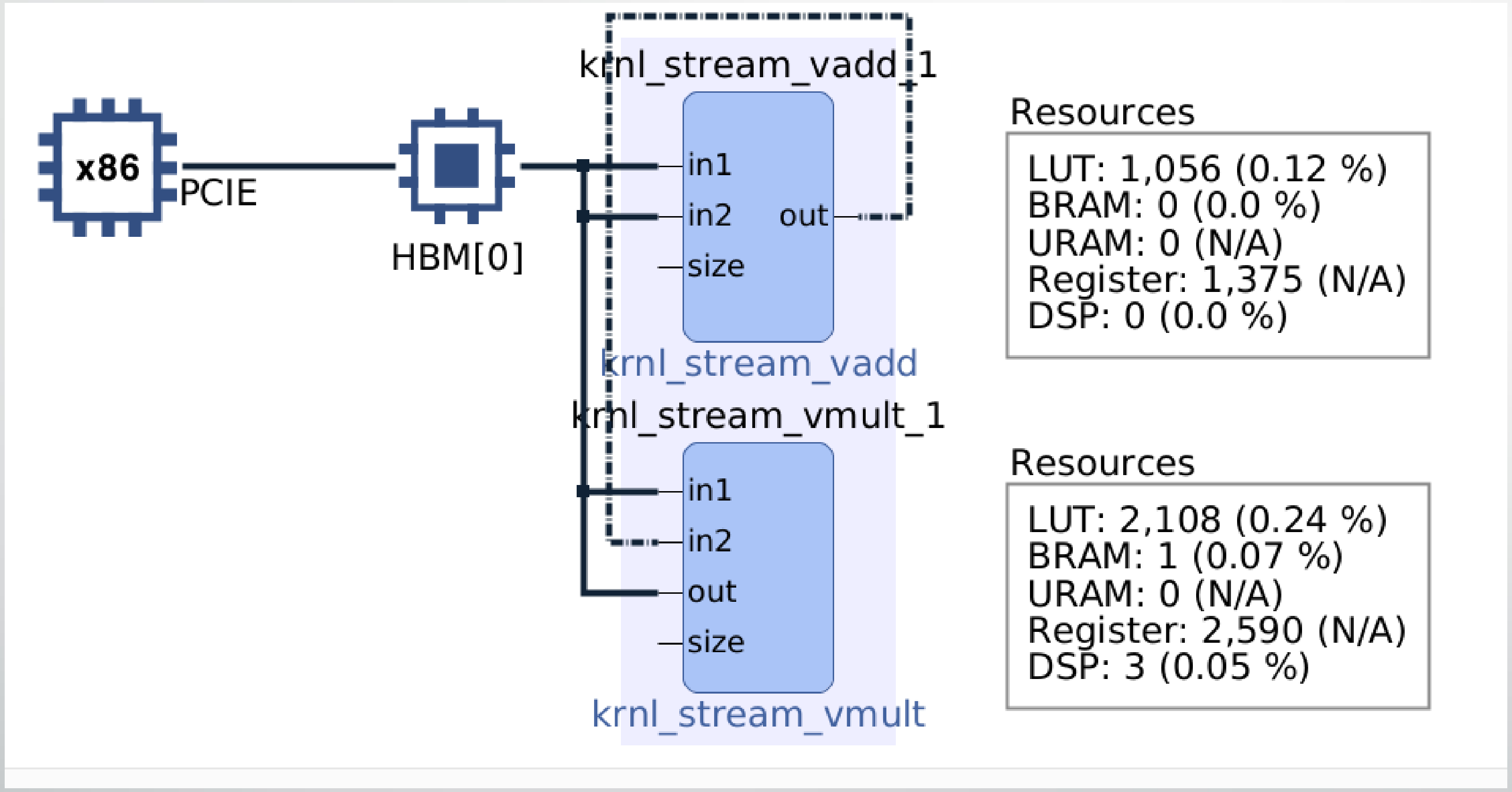


```
reset(h_a.data(), h_b.data(), h_c.data(), sw_results.data(), hw_results.data(), size);

unsigned int vector_size_bytes = size * sizeof(int);
OCL_CHECK(err, cl::Buffer buffer_in1(context, CL_MEM_USE_HOST_PTR | CL_MEM_READ_ONLY, vector_size_bytes, h_a.data(),
                                     &err));
OCL_CHECK(err, cl::Buffer buffer_in2(context, CL_MEM_USE_HOST_PTR | CL_MEM_READ_ONLY, vector_size_bytes, h_b.data(),
                                     &err));
OCL_CHECK(err, cl::Buffer buffer_in3(context, CL_MEM_USE_HOST_PTR | CL_MEM_READ_ONLY, vector_size_bytes, h_c.data(),
                                     &err));
OCL_CHECK(err, cl::Buffer buffer_output(context, CL_MEM_USE_HOST_PTR | CL_MEM_WRITE_ONLY, vector_size_bytes,
                                        hw_results.data(), &err));

OCL_CHECK(err, err = krnl_vadd.setArg(0, buffer_in1));
OCL_CHECK(err, err = krnl_vadd.setArg(1, buffer_in2));
OCL_CHECK(err, err = krnl_vadd.setArg(3, size));

OCL_CHECK(err, err = krnl_vmult.setArg(0, buffer_in3));
OCL_CHECK(err, err = krnl_vmult.setArg(2, buffer_output));
OCL_CHECK(err, err = krnl_vmult.setArg(3, size));

OCL_CHECK(err, err = q.enqueueMigrateMemObjects({buffer_in1, buffer_in2, buffer_in3}, 0));
OCL_CHECK(err, err = q.enqueueTask(krnl_vadd));
OCL_CHECK(err, err = q.enqueueTask(krnl_vmult));
q.finish();
OCL_CHECK(err, err = q.enqueueMigrateMemObjects({buffer_output}, CL_MIGRATE_MEM_OBJECT_HOST));
q.finish();
bool match = verify(sw_results.data(), hw_results.data(), size);

return (match ? EXIT_SUCCESS : EXIT_FAILURE);
```

# Streaming_k2k_mm
## Vitis_analyzer



# Streaming_k2k_mm
## Vitis_analyzer



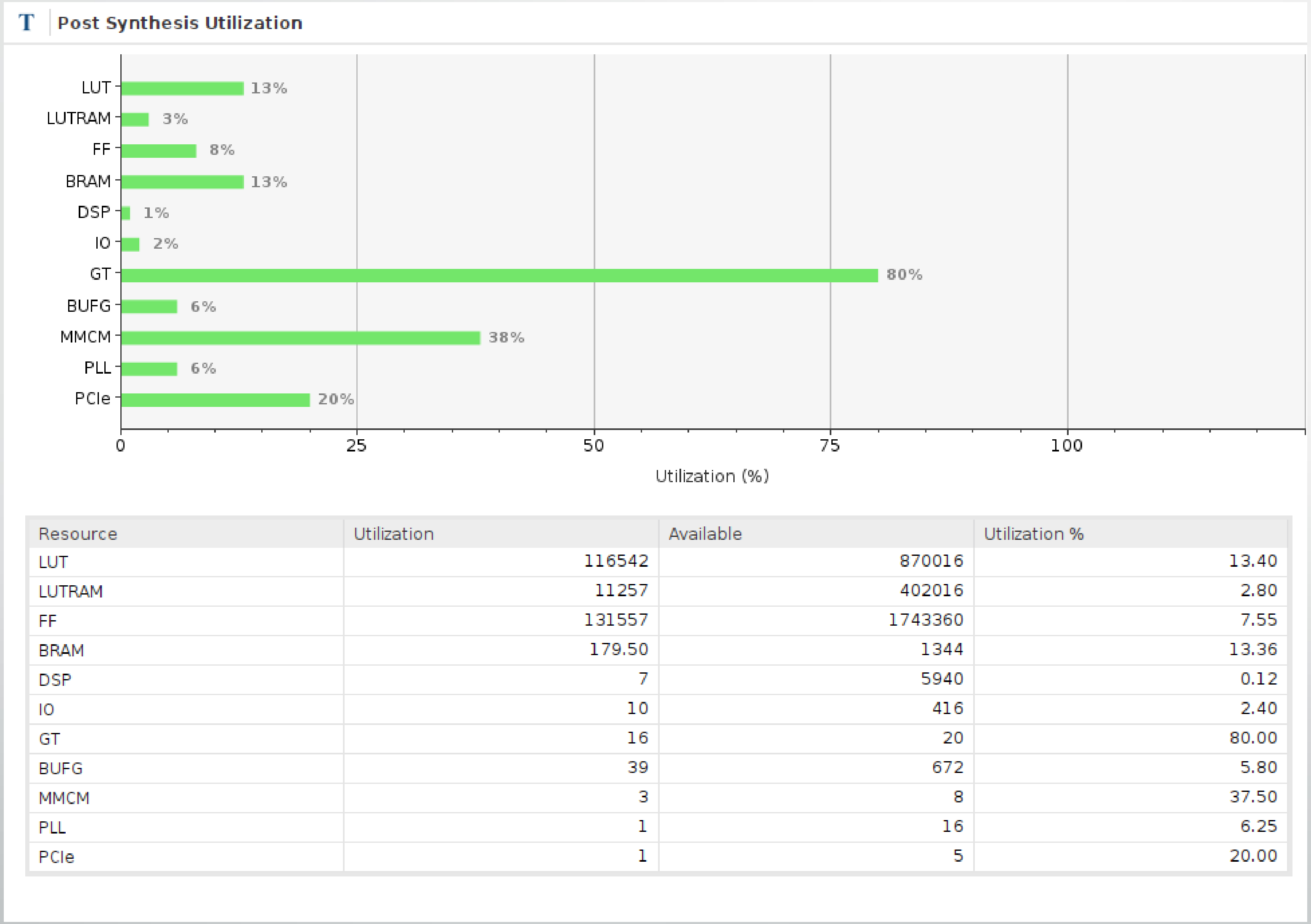
Resources

LUT: 1,056 (0.12 %)
BRAM: 0 (0.0 %)
URAM: 0 (N/A)
Register: 1,375 (N/A)
DSP: 0 (0.0 %)

Resources

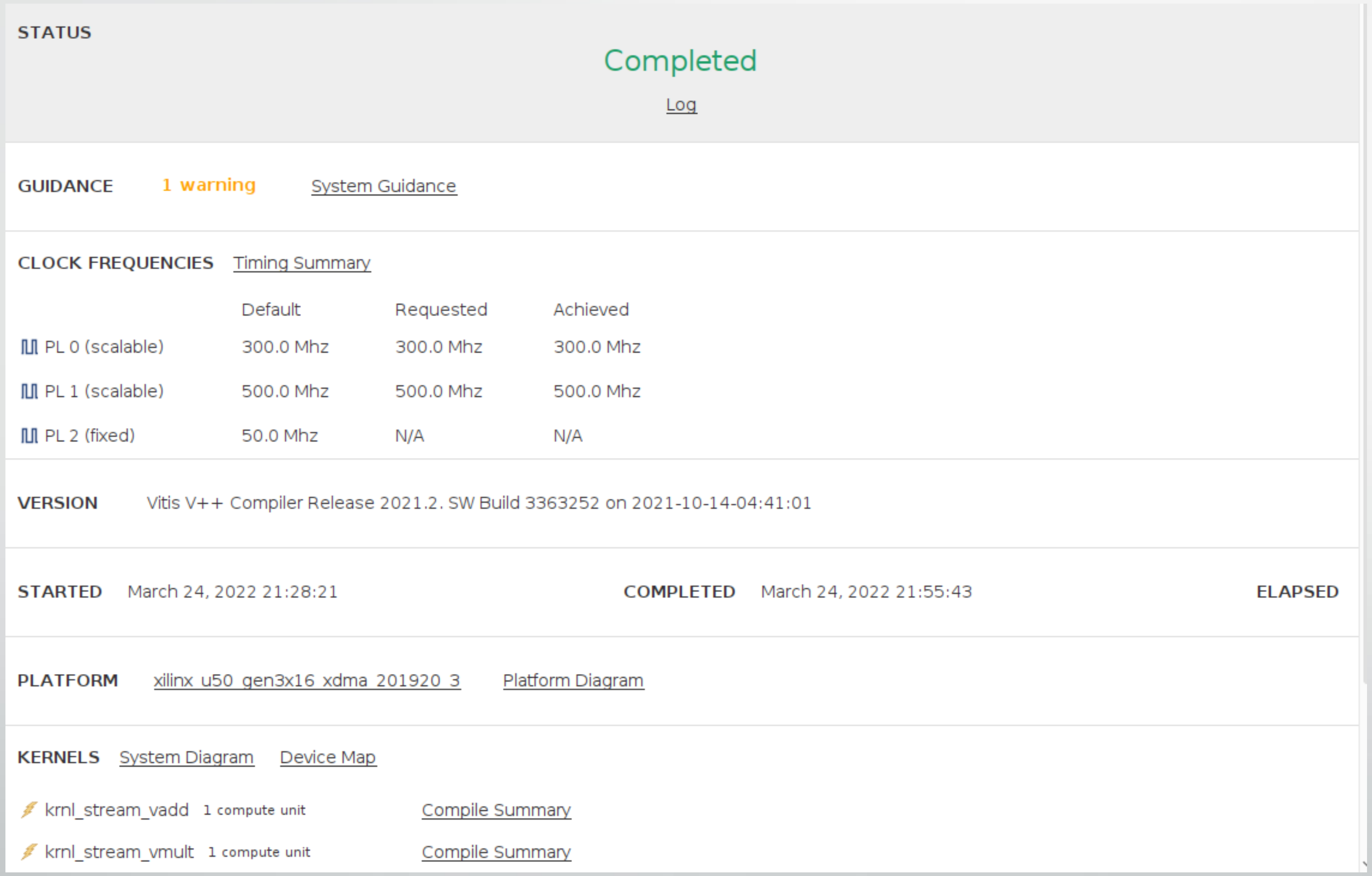
LUT: 2,108 (0.24 %)
BRAM: 1 (0.07 %)
URAM: 0 (N/A)
Register: 2,590 (N/A)
DSP: 3 (0.05 %)

# Streaming_k2k_mm
## Vitis_analyzer


Post Synthesis Utilization

| Resource | Utilization | Available | Utilization % |
|---|---|---|---|
| LUT | 116542 | 870016 | 13.40 |
| LUTRAM | 11257 | 402016 | 2.80 |
| FF | 131557 | 1743360 | 7.55 |
| BRAM | 179.50 | 1344 | 13.36 |
| DSP | 7 | 5940 | 0.12 |
| IO | 10 | 416 | 2.40 |
| GT | 16 | 20 | 80.00 |
| BUFG | 39 | 672 | 5.80 |
| MMCM | 3 | 8 | 37.50 |
| PLL | 1 | 16 | 6.25 |
| PCIe | 1 | 5 | 20.00 |

# Streaming_k2k_mm
## Vitis_analyzer



**STATUS**

Completed

Log

**GUIDANCE** 1 warning System Guidance

**CLOCK FREQUENCIES** Timing Summary

| | Default | Requested | Achieved |
|---|---|---|---|
| PL 0 (scalable) | 300.0 Mhz | 300.0 Mhz | 300.0 Mhz |
| PL 1 (scalable) | 500.0 Mhz | 500.0 Mhz | 500.0 Mhz |
| PL 2 (fixed) | 50.0 Mhz | N/A | N/A |

**VERSION** Vitis V++ Compiler Release 2021.2. SW Build 3363252 on 2021-10-14-04:41:01

**STARTED** March 24, 2022 21:28:21 **COMPLETED** March 24, 2022 21:55:43 **ELAPSED**

**PLATFORM** xilinx_u50_gen3x16_xdma_201920_3 Platform Diagram

**KERNELS** System Diagram Device Map

| | | |
|---|---|---|
| krnl_stream_vadd | 1 compute unit | Compile Summary |
| krnl_stream_vmult | 1 compute unit | Compile Summary |