Vector Addition

DPC++ Program

In [8]:

```
%%writefile ~/arc/vector_addition.cpp
#include <CL/sycl.hpp>
#include <iostream>
#include <sycl/ext/intel/fpga_extensions.hpp>
#include <chrono>
using namespace sycl;

#define VECTOR_SIZE 1024

int main(int argc, char *argv[]) {

    // define input and output vectors
    std::vector<int> in1;
    std::vector<int> in2;
    std::vector<int> out;
    std::vector<int> val;

    // resize vectors
    in1.resize(VECTOR_SIZE);
    in2.resize(VECTOR_SIZE);
    out.resize(VECTOR_SIZE);
    val.resize(VECTOR_SIZE);

    // load input vectors
    for (size_t i = 0; i < in1.size(); i++) {
        in1.at(i) = i;
        in2.at(i) = i;
    }

    // create the queue
    queue queue(property::queue::enable_profiling{});
    std::cout << "Offload Device: " << queue.get_device().get_info<info::device::name>() << "\n";

    // create a range object for the buffers
    range<1> num_items{ in1.size() };

    // create buffers
    buffer in1_buffer(in1);
    buffer in2_buffer(in2);
    buffer out_buffer(out);

    auto device_start = std::chrono::high_resolution_clock::now();
    auto event = queue.submit([&](handler& handler) {

        // create accessors for the input/output buffers
        auto in1_accessor = in1_buffer.get_access<access::mode::read>(handler);
        auto in2_accessor = in2_buffer.get_access<access::mode::read>(handler);
        auto out_accessor = out_buffer.get_access<access::mode::write>(handler);

        // perform operation using parallel_for
        // 1st param: num work items
        // 2nd param: kernel to specify what to do per work item
        handler.parallel_for(num_items, [=](auto i) {
            out_accessor[i] = in1_accessor[i] + in2_accessor[i];
        });

    });
    auto device_stop = std::chrono::high_resolution_clock::now();
    auto device_duration = std::chrono::duration_cast<std::chrono::microseconds>(device_stop - device_start);

    // allow read access for output buffer
    out_buffer.get_access<access::mode::read>();

    // get reported times from kernel event profile
    auto kernel_end = event.get_profiling_info<info::event_profiling::command_end>();
    auto kernel_start = event.get_profiling_info<info::event_profiling::command_start>();
    auto kernel_duration = (kernel_end-kernel_start)/1.0e3;

    // host computation for validation and timing comparision
    auto host_start = std::chrono::high_resolution_clock::now();
    for (size_t i = 0; i < val.size(); i++) {
        val.at(i) = in1.at(i) + in2.at(i);
    }
    auto host_stop = std::chrono::high_resolution_clock::now();
    auto host_duration = std::chrono::duration_cast<std::chrono::microseconds>(host_stop - host_start);

    // validate
    for (size_t i = 0; i < val.size(); i++){
        if (out.at(i) != val.at(i)) {
            std::cout << "Incorrect values from device.\n"
                << out.at(i) << " != " << val.at(i) << "\n";
            return -1;
        }
    }

    int indices[]{ 0, 1, 2, (static_cast<int>(in1.size()) - 3), (static_cast<int>(in1.size()) - 2), (static_cast<int>(in1.size()) - 1) };
    constexpr size_t indices_size = sizeof(indices) / sizeof(int);

    // Print results.
    std::cout << "\n";
    for (int i = 0; i < indices_size; i++) {
        int j = indices[i];
        if (i == indices_size - 3) std::cout << "...\n";
        std::cout << "[" << j << "]: " << in1[j] << " + " << in2[j] << " = "
            << out[j] << "\n";
    }

    std::cout << "\nKernel execution time: " << kernel_duration << " microseconds\n"
        << "Device duration: " << device_duration.count() << " microseconds\n"
        << "Host comparison: " << host_duration.count() << " microseconds\n";

    return 0;
}
```

```
Overwriting /home/u167808/arc/vector_addition.cpp
```

Shell script to compile and run program

In [9]:
```
%%writefile ~/arc/vector_addition.sh
#!/bin/bash
source /opt/intel/inteloneapi/setvars.sh > /dev/null 2>&1

echo ====================
echo vector_addition
dpcpp ~/arc/vector_addition.cpp -o ~/arc/vector_addition -w -O3
~/arc/vector_addition
echo ====================
```

```
Overwriting /home/u167808/arc/vector_addition.sh
```

Script to queue jobs on Intel DevCloud

In [10]:
```
%%writefile ~/arc/submit_job.sh
#==========================================
# Copyright © 2020 Intel Corporation
#
# SPDX-License-Identifier: MIT
#==========================================
# Script to submit job in Intel(R) DevCloud
# Version: 0.72
#==========================================

if [ -z "$1" ]; then
    echo "Missing script argument, Usage: ./q run.sh"
elif [ ! -f "$1" ]; then
    echo "File $1 does not exist"
else
    echo "Job has been submitted to Intel(R) DevCloud and will execute soon."
    echo ""
    script=$1
    property=$2
     if [ "$property" == "GPU GEN9" ]; then
             value="gen9"
        elif [ "$property" == "GPU Iris XE Max" ]; then
            value="iris_xe_max"
        elif [ "$property" == "CPU Xeon 8153" ]; then
            value="renderkit"
        elif [ "$property" == "CPU Xeon 8256" ]; then
            value="stratix10"
        elif [ "$property" == "CPU Xeon 6128" ]; then
            value="skl"
        else
            value="gen9"
    fi
    if [ "$property" == "{device.value}" ]; then
        echo "Selected Device is: GPU"
    else
        echo "Selected Device is: "$property
    fi
    echo ""
    # Remove old output files
    rm *.sh.* > /dev/null 2>&1
    # Submit job using qsub
    qsub_id=`qsub -l nodes=1:$value:ppn=2 -d . $script`
    job_id="$(cut -d'.' -f1 <<<"$qsub_id")"
    # Print qstat output
    qstat
    # Wait for output file to be generated and display
    echo ""
    echo -ne "Waiting for Output "
    until [ -f $script.o$job_id ]; do
        sleep 1
        echo -ne "█"
        ((timeout++))
        # Timeout if no output file generated within 60 seconds
        if [ $timeout == 60 ]; then
            echo ""
            echo ""
            echo "TimeOut 60 seconds: Job is still queued for execution, check for output file later ($script.o$job_id)"
            echo ""
            break
        fi
    done
    # Print output and error file content if exist
    if [ -n "$(find -name '*.sh.o'$job_id)" ]; then
        echo " Done⬇"
        cat $script.o$job_id
        cat $script.e$job_id
        echo "Job Completed in $timeout seconds."
        rm *.sh.*$job_id > /dev/null 2>&1
    fi
fi
```

```
Overwriting /home/u167808/arc/submit_job.sh
```

Execute program

In [11]:
```
! chmod 755 ~/arc/submit_job.sh; chmod 755 ~/arc/vector_addition.sh; ~/arc/submit_job.sh ~/arc/vector_addition.sh "GPU Gen9";
```

```
Job has been submitted to Intel(R) DevCloud and will execute soon.

Selected Device is: GPU Gen9

Job ID                    Name             User            Time Use S Queue
------------------------- ---------------- --------------- -------- - -----
2034760.v-qsvr-1           ...ub-singleuser u167808         00:00:20 R jupyterhub
2034787.v-qsvr-1           ...r_addition.sh u167808                0 Q batch

Waiting for Output ███████████████████████████████████ Done⬇

########################################################################
#      Date:           Wed 09 Nov 2022 03:36:14 AM PST
#    Job ID:           2034787.v-qsvr-1.aidevcloud
#      User:           u167808
# Resources:           neednodes=1:gen9:ppn=2,nodes=1:gen9:ppn=2,walltime=06:00:00
########################################################################

====================
vector_addition
Offload Device: Intel(R) UHD Graphics P630 [0x3e96]

[0]: 0 + 0 = 0
[1]: 1 + 1 = 2
[2]: 2 + 2 = 4
...
[1021]: 1021 + 1021 = 2042
[1022]: 1022 + 1022 = 2044
[1023]: 1023 + 1023 = 2046

Kernel execution time: 14.608 microseconds
Device duration: 252699 microseconds
Host comparison: 1 microseconds
====================

########################################################################
# End of output for job 2034787.v-qsvr-1.aidevcloud
# Date: Wed 09 Nov 2022 03:36:24 AM PST
########################################################################

Job Completed in 35 seconds
```