Ian Picariello 11/26/17

EECE.6540 – Lab 3 – Evaluation of FPGA Board Test

**Description of lab:**

The goal of this lab is to make the host and device binaries and execute the device binary on the FPGA board. The board emulator is not to be used and the physical hardware is to be used to run the series of tests called in the execution. The compilation for the device takes about an hour, while the host compilation and code execution each run in less than a minute. After the binary is executed, the results are displayed to the screen and two of the tests are to be post-processed and understood. The tests being investigated are the kernel clock frequency test and the kernel latency test.

**Summary of the outcome:**

The lab was executed successfully with the results attached to the code linked at the bottom of the report. The following two tests were examined: kernel clock frequency test and the kernel latency test.

Kernel Clock Frequency Test is implemented to calculate the speed of numerous kernel executions on the board compared to the speed reported by the hardware. The comparison takes the number of executions divided by the total time of all the kernel executions and check if it is within two percent of the boards compiled frequency. The kernel execution time was 290. 245606MHz to the quartus frequency of 290.690002MHz, which results within the two percent (0.153%) frequency of the board’s operation rate. The test is executed in the following steps:

1. Setting up variables – global work size, local work size, and timer
2. Creating the kernel to execute
3. Starting the timer – gets current time from the linux computer
4. Executing the kernel (global size is 134,217,728; with the number of work items in a work group = 32,768)
5. Finish the command and release the kernel
6. Stop the timer – gets the current time from the linux computer
7. Calculates the difference between the start and stop times in seconds
8. Calculate the bandwidth – global work items / time difference; note the equation is multiplied by two for future tests
9. Determine the measured frequency – equal to the bandwidth; note the equation is divided by a floating constant two to create a floating variable and remove the two multiplied when calculating the bandwidth
10. Open the quartus report file – if the file is not present or an error occurs, a message is display and the test aborts, freeing the resources in the process
11. If the file opened correctly, the test then runs through the file until it finds the string “Actual” and breaks
12. If the quartus report file is not at the end, the test then scans out the next two words before returning the quartus frequency
13. The test then prints the measured frequency (from step 9) and the frequency reports from the quartus report
14. The test determines if the two frequencies are within two percent of each other – (quartus frequency – measured frequency) / quartus frequency \* 100
15. A message is displayed with the text dependent on if the percentage is above or below two
16. The test ends by freeing the resources

Kernel Latency Test is used to determine the duration it takes to execute the kernel. The test displays three separate results all based upon the time elapsed between the start of executing until a set number of executions complete. The test was setup to execute 10,000 tasks, which took the board 234.8487 milliseconds to complete. Each kernel ran for 23.4849 microseconds with the board having a bandwidth of 42.5806 kernels per millisecond. The test is executed in the following steps:

1. Create the kernel to execute
2. Start the timer
3. Queue up a task to execute the kernel and loop for 10,000 instances
4. Stop the timer when all executions have completed
5. Determine the length of time it took to operate all 10,000 kernels – get\_time\_s function; duration was equal to ~0.235 seconds or ~235 milliseconds
6. Calculate the bandwidth in number of kernels per milliseconds – # of kernels / [time \* 1000]
7. Print the value for the three highlighted results – duration of executing 10,000 kernels = ~235 millisecond; duration of executing one kernel = ~23.5 microseconds; bandwidth = 42.6 kernels per millisecond
8. Duration of execution one kernel was calculated by taking the total duration (~235 milliseconds) and dividing by the number of kernels (10,000) to return ~23.5 microseconds
9. Release the resources and end the test

**Main hurdles and difficulties:**

The two hurdles experienced when performing this lab were connecting to the server after the holiday shutdown and running the main executable. The first issue was attributed to the holiday break and the computer not being setup post break and the second issue was because the linux kernel was incorrectly booted. Other than these issues, the lab was main straightforward.

**Things learned:**

Understanding how to create the binary for the board with the following command “aoc -v --board de5net\_a7 --report device/boardtest.cl -o bin/boardtest.aocx”, which took about an hour to run. This function only provided the FPGA binary, where the host side binary was created with the “make” function and took less than a minute to run.

Learning how to run the code took a few attempts, most of them after finding out the computer was not setup properly post-holiday break. The execution for the host side binary was done by simply run the executable “./boardtest\_host”. The execution took about a minute to run.

**Suggestions:**

None. This lab provided insight on how to create both host and device binaries, while executing them as well.

**GitHub link location:**

https://github.com/ianpic612/EECE6540\_Fall2017/tree/master/Labs/Lab3