Exercise Manual

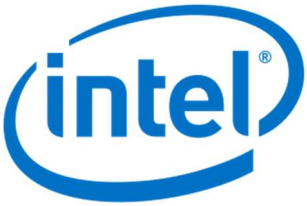
for

Introduction to OpenCL™ for Intel® FPGAs

Lab Exercise 4

Software Requirements

CentOS 7 Linux* OS
Eclipse IDE
Intel® FPGA SDK for OpenCL™ version 19.1
Intel® Quartus® Prime Pro software version 19.1 with Arria® 10 family
Intel® Code Builder for OpenCL™ (Needed to run fast emulator)



Exercise 4

Examining Kernel Compile Results

In this exercise, we will examine some of the outputs of the Intel® FPGA OpenCL™ offline compiler that can help us debug and optimize the kernels.

Step 1. Convert and Compile the Kernel

____ 1. In a terminal, go to `/home/student/fpga_trn/OpenCL/OCL_19_1`

a. Run **`opencl_init.sh`** if it hasn't been run

____ 2. Compile **SimpleKernel_For.cl,** this time NOT in the emulation mode, but we'll stop the compiler after the RTL files are generated. Do this by typing the following command:

```
aoc -board=a10gx -rtl SimpleKernel_For.cl
```

We're going to recompile the for loop version of the kernel from Lab 2. The -rtl option stops the compiler after the RTL generation but before full FPGA compilation which is a much quicker compile than the full FPGA compile. The -rtl compile still allows us to see the detailed static kernel reports for optimization purposes.

This time you'll see warnings if you didn't use the restrict keyword on global arguments due to possible performance sacrifices. You may wish to add the keyword restrict to remove the warning. e.g. ***__global const float * restrict in***

____ 3. In the terminal, type "cd SimpleKernel_For" to navigate to

```
/home/student/fpga_trn/OpenCL/OCL_18_0/SimpleKernel_For
```

This folder is named after the kernel file and contains all the compilation results including the generated hardware source files.

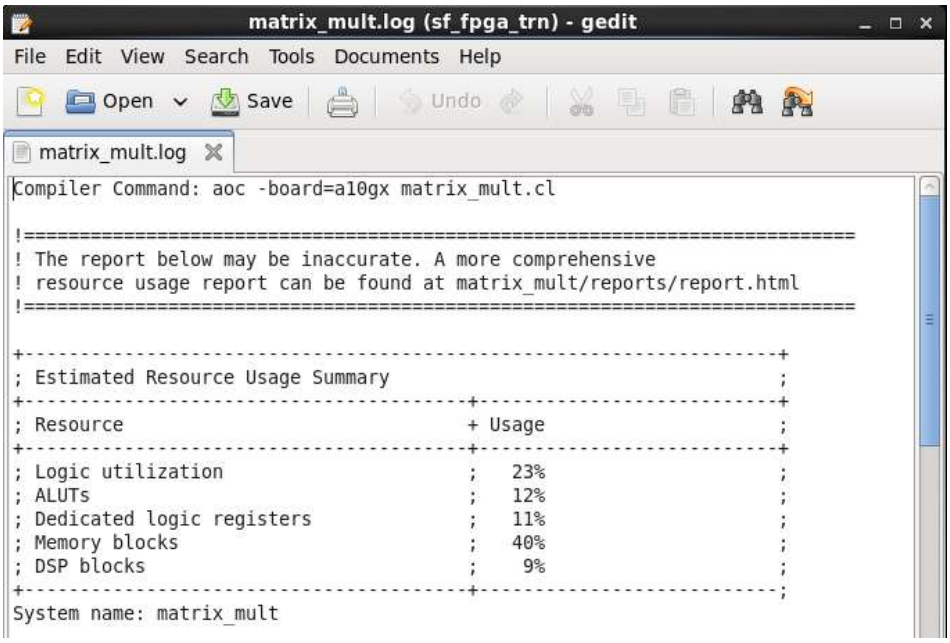
____ 4. Examine the log file

a. Inside the SimpleKernel_For folder, look for SimpleKernel_For.log

This is the compile log which contains Estimated Resource Utilization Report and Compile Messages

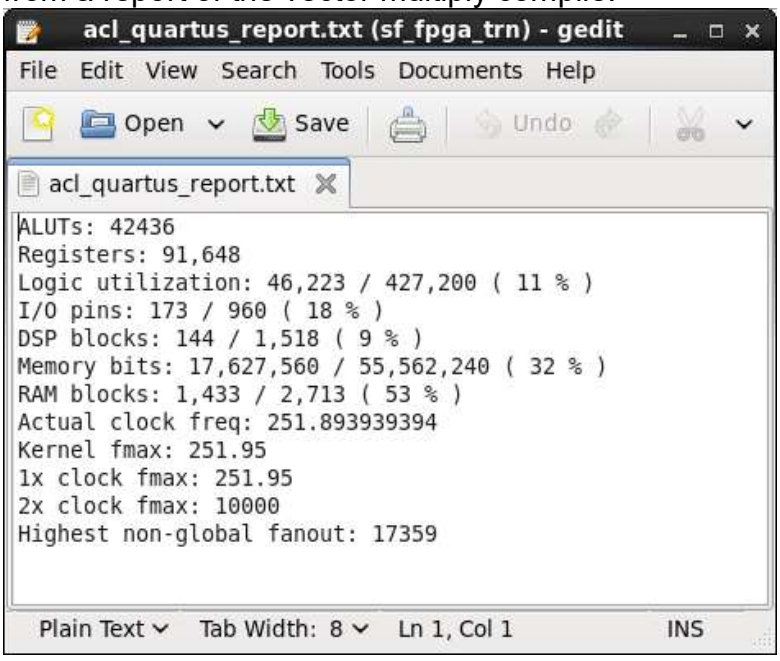
b. Type "gedit SimpleKernel_For.log" to examine the log file.

____ 5. The log file includes an estimated resource usage report. This report is an early estimate prior to the Intel® Quartus® Prime software compilation.



The screen capture is for a vector multiply example. You may wonder why such a kernel consumes this many resources. The reasons are that much of the logic is occupied by the board support package to support the various interfaces, loops have been unrolled, and the number of concurrent work items (which are smaller matrices) increased to get good throughput..

____ 6. If this was a full compilation you would see acl_quartus_report.txt which contains information on the actual resource utilization as well as the actual clock frequency used in the Kernel subsystem. The screen capture below is from a report of the vector multiply compile.



As you can see full compilation resource usage matches very closely with the early estimate for DSP block and RAM blocks. For logic utilization, aoc was conservative with its estimate.

____ 7. In the terminal, navigate to

```
/home/student/fpga_trn/OpenCL/OCL_19_1/SimpleKernel_For/
reports
```

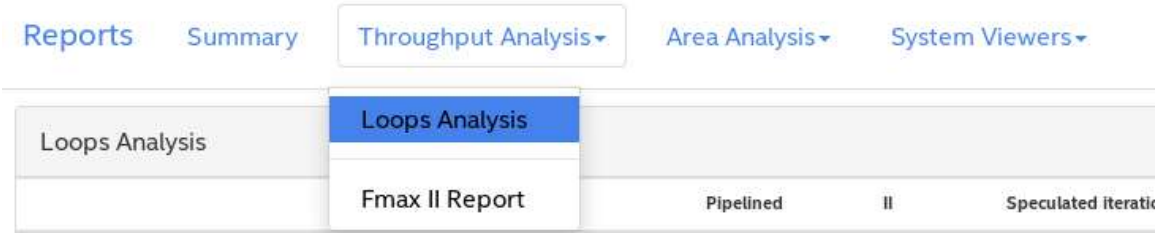
____ 8. Type “firefox report.html” to open it in a web browser

This is the unified kernel compilation static report that’s capable of showing many different types of information.

____ 9. Scroll down in the summary page. Is your kernel compiled as a single work-item or ND Range kernel? _____

This summary page shows information about compile warnings, estimated resource usage, and how kernels are compiled.

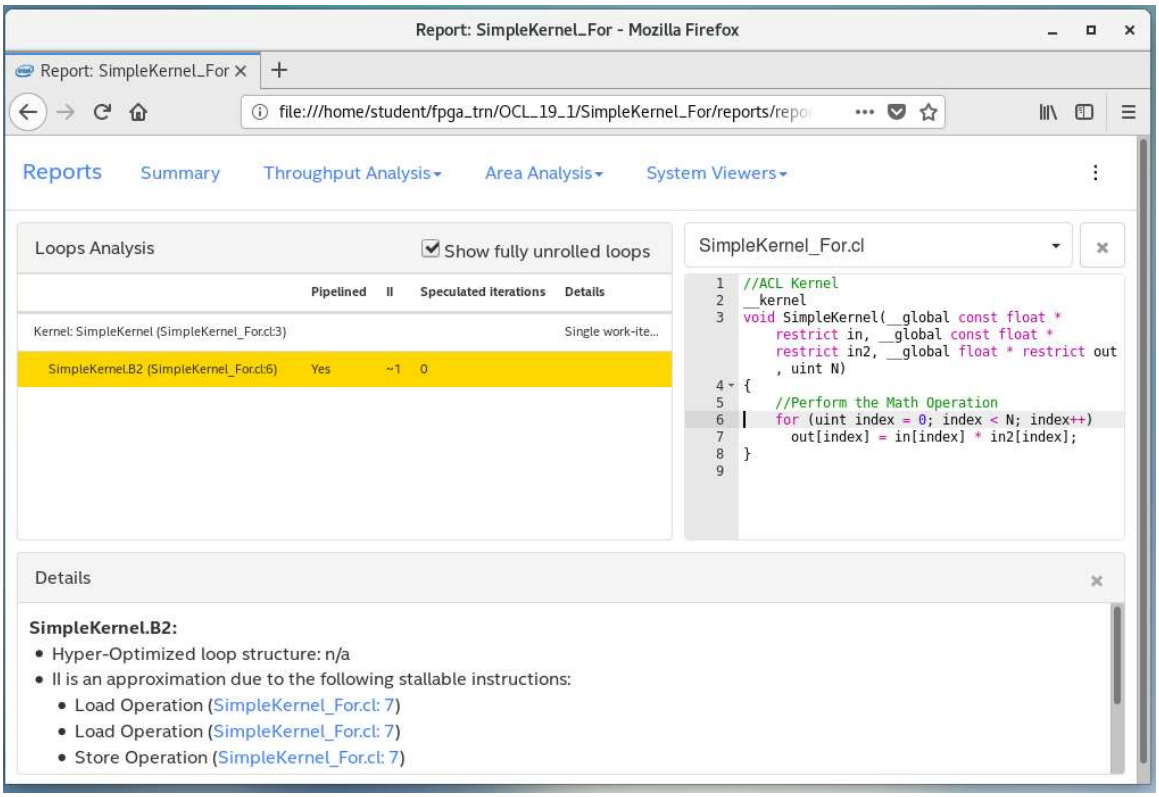
____ 10. At the top of the web page, you can choose among several different types of viewers.



____ 11. Examine the Loop implementation information by choosing **Loop analysis** from the Throughput Analysis drop down

____ 12. How every loop is implemented is shown in the report.

We expect our loops to be pipelined which is the case here. You’ll also see an II value, which is the initiation interval or cycles between iteration launches. II=1 is the best the compiler can do. If the II is > 1 the analysis will also show details about that in the report.



____ 13. Click on the loop as shown above.

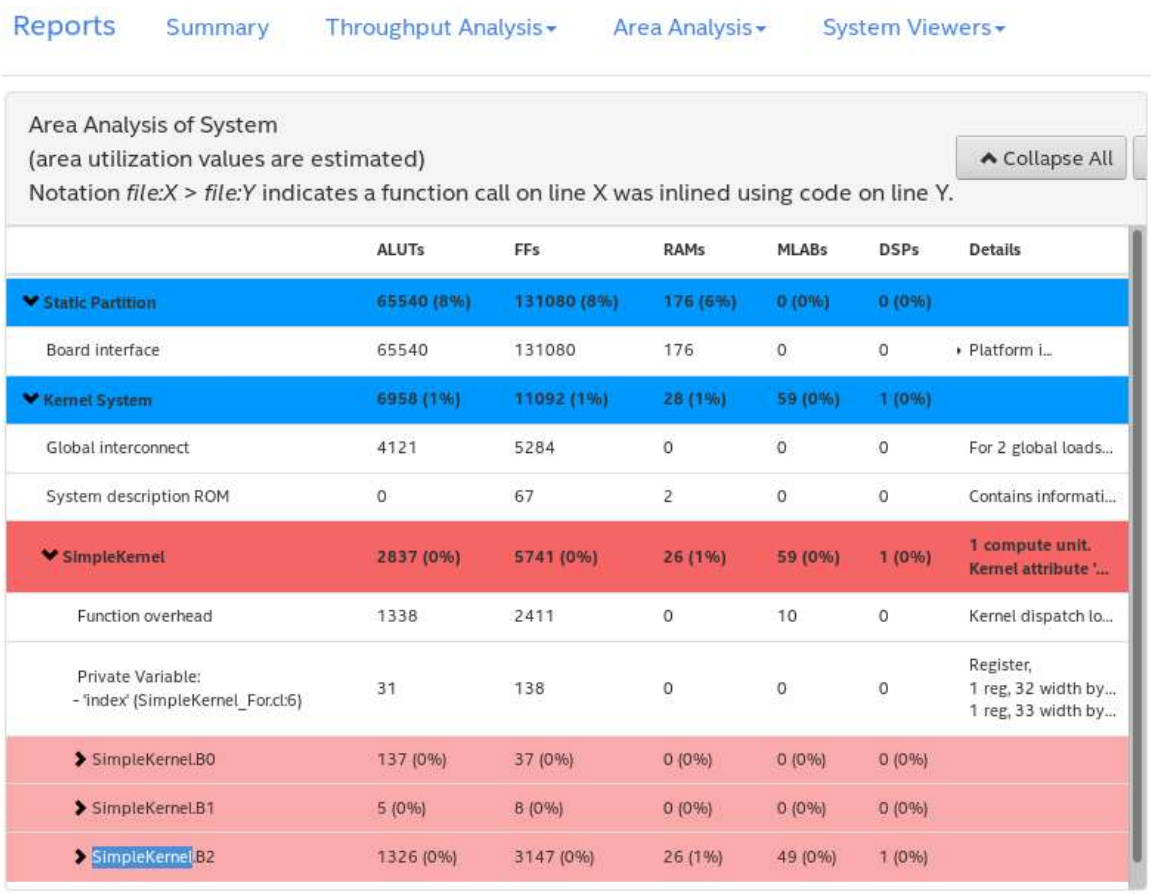
Clicking on the loop will show the actual location of the loop in the source file.

____ 14. Next, let's examine the detailed area report by choosing **Area analysis of System** from the Area Analysis drop-down menu

There are two ways to view the detailed area report by source code or by system blocks. Source code will correspond well to the original source cl file. Blocks will correspond well to the actual circuit created.

____ 15. Expand "Static Partition" and "Kernel System" and then expand "SimpleKernel"

Now we will see some detailed information and breakdown of resource usage.



	ALUTs	FFs	RAMs	MLABs	DSPs	Details
Static Partition	65540 (8%)	131080 (8%)	176 (6%)	0 (0%)	0 (0%)	
Board interface	65540	131080	176	0	0	Platform i...
Kernel System	6958 (1%)	11092 (1%)	28 (1%)	59 (0%)	1 (0%)	
Global interconnect	4121	5284	0	0	0	For 2 global loads...
System description ROM	0	67	2	0	0	Contains informati...
SimpleKernel	2837 (0%)	5741 (0%)	26 (1%)	59 (0%)	1 (0%)	1 compute unit. Kernel attribute '...
Function overhead	1338	2411	0	10	0	Kernel dispatch lo...
Private Variable: - 'index' (SimpleKernel_For.cl:6)	31	138	0	0	0	Register, 1 reg, 32 width by... 1 reg, 33 width by...
SimpleKernel.B0	137 (0%)	37 (0%)	0 (0%)	0 (0%)	0 (0%)	
SimpleKernel.B1	5 (0%)	8 (0%)	0 (0%)	0 (0%)	0 (0%)	
SimpleKernel.B2	1326 (0%)	3147 (0%)	26 (1%)	49 (0%)	1 (0%)	

As you can see for our simplekernel, the majority of resources were taken up by the Board interface, which is the logic used by the board support package interfaces and components.

____ 16. Click on several of the lines inside SimpleKernel. Expand the pink blocks and then sub categories.

If there's a corresponding line in the source file it will be highlighted. You will also see a detailed breakdown of resource utilization by each operator at each of the line numbers.

____ 17. Switch to the "System Viewer" from the System Viewer dropdown

The System view shows the latencies of the individual blocks in the kernel pipeline as well as the load and store to the memory.

____ 18. Hover over various blocks in the pipeline including the loop and loop end blocks.

Pay attention to the latency information as well as other information under "Details"

____ 19. Click on the LD and ST units

Notice the type of Load and Store units created. In the AOC implementation, each variable gets its own load store units with its own cache. The access to global memory however is arbitrated. Here you can also see the number of access to each memory. If your global or local memory is banked, you will see the number of accesses to each bank.

____ 20. Once you've finished examining this report close the web browser.

____ 21. Shutdown the virtual machine by going to **System -> Shut Down…** on the desktop and clicking **Shut Down** on the popup window.

Exercise Summary

- Examined the various reports created by the AOC tool which will help you debug and optimize your kernel.

Congratulations!

You have completed Lab 4
