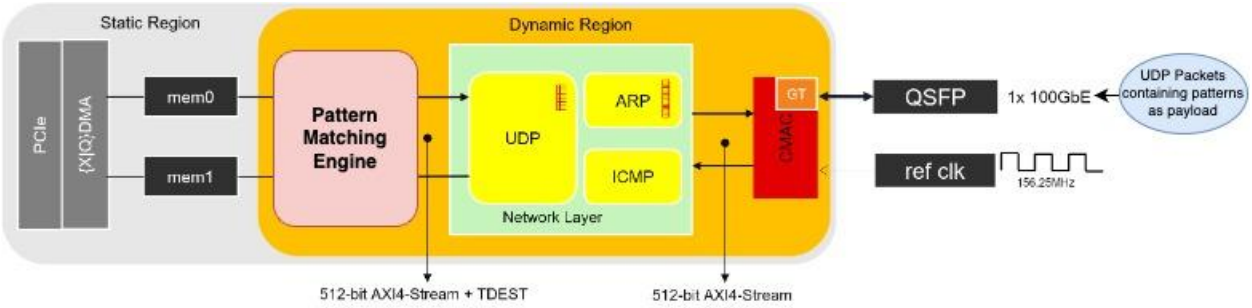
DAT480 Pattern Matching Lab Project

The goal of the lab project is to build a network processing engine for pattern matching using the Alveo U55C FPGA board to achieve a high processing throughput of up to line-rate.



Network Input: UDP packets with payload containing the patterns to be detected at up to 100Gb/s (64 bytes per FPGA clock cycle of 300MHz).

Input interface to your Pattern Matching Engine: Continuous stream of 64 byte data chunks per FPGA cycle extracted from the UDP packet. Note that a pattern may span over multiple 64 byte data chunks.

Output to be produced by your Pattern Matching Engine: Pattern-IDs matched, based on the line numbers in the rule set file provided. A list of matched IDs should be communicated to the host in some way, which for example could be done by writing the matched IDs to memory or streaming them back over the network.

Report Archive Submission: You will need to submit an archive (zip file) on Canvas with the report PDF describing your design decisions and the results. The report should contain the synthesis results, the resource utilization on the FPGA, and the achieved processing throughput, as well as a block diagram explaining your design. Keep it short and simple, at most two pages.
The archive should also contain the source code for your pattern matching engine, the associated configuration files, and the instructions for running and verifying your submission on the Alveo U55C FPGA board. The submitted files will be passed through plagiarism check, and you should have shown your design(both the code and it running on the FPGA) to a TA before handing it in.
The hard deadline for this submission is before the course ends in January, but you should aim to submit before the 19th of December, TAs will not be available for help during Christmas.

Presentation: There will also be a presentation where you will present your work. The emphasis of this presentation should be on your design decisions and how you choose to approach the problem.

Grading: The following points as well as the quality of your work will be used to determine the grades:

- Processing throughput: Minimum one byte per cycle working on hardware. To add to a higher grade your design should process multiple bytes per cycle.
- The number of patterns supported. From either the MINI_pattern_match_snort3_content.txt (minimum) or pattern_match_snort3_content.txt files.
- Automatic preprocessing / design generation from provided patterns.

Instructions:

Before running anything on the FPGA, look at the canvas page “FPGA Booking”, and make sure that you have booked the current timeslot for the FPGA you are going to use!
Now to the project.

Start by downloading the project template:

```
git clone https://git.chalmers.se/magnusos/dat480_project_base.git --recursive
```

The template is built upon the open source project XUP Vitis Network Example(VNx). For understanding how the FPGA <-> Network interface works, and how you can use it, please start by looking at the basic design provided by VNx. The basic design sets up a UDP/IP stack as well as two stream engines(krnl_mm2s and krnl_s2mm) to push or pull data to/from the FPGA DRAM over the network. The HLS code for these stream engines(found under Basic_kernels), as well as the config file that connects them to the network stack(config_files/connectivity_basic_if0.ini), and the top level and Basic_kernels Makefiles are well worth studying, for you to understand how to set up your project implementation.
You should also run this example on the FPGA by going through the Notebooks/vnx-basic.ipynb jupyter notebook. For the bitfile(.xclbin) you can either build it yourself by running:

```
make all DESIGN=basic
```

This build will take a couple of hours. Or you can copy it from /home/shared/vnx_basic_if0.xclbin

Now, depending on if you want to do the project in RTL or HLS, have a look in the Project_kernels_RTL or Project_kernels_HLS folder, there you will find a very basic VHDL template, or an empty C++ file, as well as the Makefiles to build these into Vitis Kernels(.xo files).
You will also have to modify the config_files/connectivity_project_if0.ini config file to correctly instantiate and hook up whatever kernel(s) your project implementation requires. You might also have to edit the top level Makefile if your design uses other kernels than the one currently defined in Project_kernels_HLS.

Remember that synthesis and implementation(place and route) take a long time for a large FPGA system like this, so always verify your design through simulation(both kinds if using HLS) before building a bitfile and trying on real hardware. The time for this project is also limited, so please make sure that you get a basic implementation(processing one byte a cycle for the limited pattern set) working before trying out more advanced or experimental designs.