Assignment 3: Motion Detection

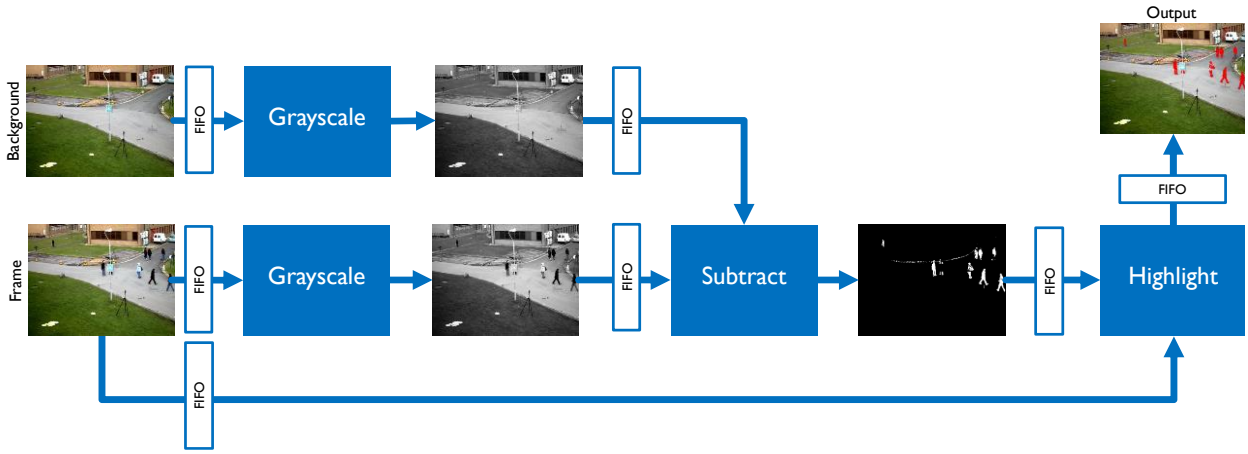
In this assignment we're building a motion detection algorithm to identify objects moving in a scene. Generally this is used with static cameras. The basic concept is to subtract one image from a background image, or an average of pervious images. We generally expect some noise between the background and subsequent images. To mitigate the difference, two methods are generally employed. The first is gaussian blur, which we are not including in this algorithm. The second is using a threshold in measuring the difference between pixels in the two images.

Goals:

- Implement a streaming architecture using FIFOs to pass data between modules
- Understand how to pass data across multiple data paths
- Read and write binary images in simulation and verify bit-true accuracy

Implementation:

- Run the provided C program of the motion_detect application to generate the output image for the motion detection.
    - **gcc -o motion_detect motion_detect.c**
    - There is also a video implementation using OpenCV. Run the provided **compile** script to build the program. This is not required for the lab assignment, but demonstrates how to detect motion in a video stream.
- Implement the **grayscale**, **background subtraction**, and **highlight** modules in SystemVerilog. The modules should read and write to FIFOs with only a small number of elements (e.g. 32).



- The modules and FIFOs should be instantiated in a motion_detect_top wrapper module. The implementation should be relatively similar to the streaming Vectorsum example from class.
- The modules should be implemented using the 2-process method (always_ff and always_comb), and should contain a finite-state machine (FSM) using the FIFO read/write operations and discussed in class.
- The application should process the entire image is approximately O(N) time in the size of the image.

Verification:

- Create a testbench that reads the base (background) and pedestrian BMP images and push the data into the FIFOs. Note that the BMP headers are 54 bytes which should be stored and written to the output image file.
- Read in the BMP output image from the software and compare against the modelsim simulation output, and report any errors.
- Create a simulation.do file that compiles the SystemVerilog files, sets up the wave form, and runs the simulation. You should be able to run "vsim -do motion_detect_sim.do" from the commandline.
- Run the simulation in modelsim and verify the design is bit-true accurate.

Synthesis:

- Compile the design in Synplify Premier to get high-level resource results and timing information.
- Target the Intel Cyclone IV-E FPGA
- Get the resource utilization and timing information (max frequency)

Reporting:

- Submit a PDF file with simulation and synthesis results with the following:
  - Simulation results:
    - Clock cycle count
    - Errors reported (if any)
  - Synthesis results (some might not apply):
    - Maximum frequency
    - Registers / LUTs / Logic Elements
    - Memory utilization
    - Multipliers (DSPs)
    - Worst path (timing analysis)
    - Schematic architecture (RTL)

Turn in your designs with the report. Your file should be zipped, and should include the SystemVerilog files, synthesis, simulation, and input/output files. **PLEASE MAKE SURE TO REMOVE ALL THE INTERMEDIATE WORK DIRECTORIES.**