EECS150 Fall 2013 Checkpoint 4: Testing Gaussian Filter Block

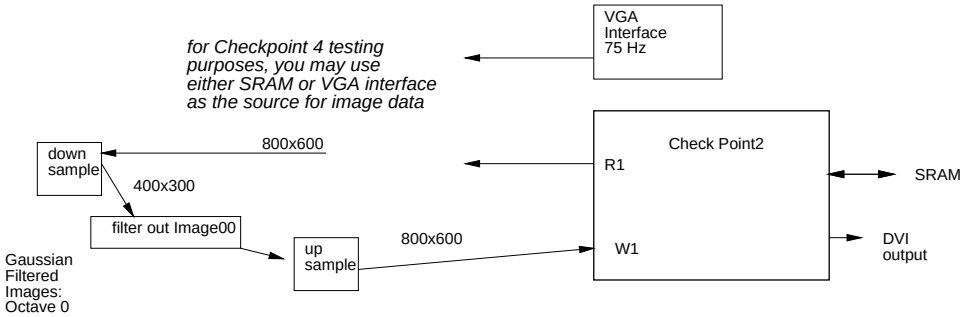
Prof. Ronald Fearing, GSIs: Austin Buchan, Stephen Twigg
Department of Electrical Engineering and Computer Sciences
College of Engineering, University of California, Berkeley

Due week of Nov. 18.
Draft v1.1 Nov. 16.

References

1. Bonato, V.; Marques, E.; Constantinides, G.A., "A Parallel Hardware Architecture for Scale and Rotation Invariant Feature Detection," *Circuits and Systems for Video Technology, IEEE Transactions on,* vol.18, no.12, pp.1703,1712, Dec. 2008
2. CS150 Lectures #14, #15

For this checkpoint, you will be implementing and testing one Gaussian filtering pipeline block. (This is one of the main blocks you drew the detailed block diagram for in Checkpoint 3.) The overall block diagram for this checkpoint is shown below. For testing purposes, you will read an 800x600 image from **either** one SRAM buffer or the provided VGA interface module. Then down-sample to 400x300, filter it through the Gaussian filter pipeline, up-sample to 800x600, and then store the result in the other SRAM buffer. You will use your SramArbiter from Checkpoint2, where port W1 values will come from the up-sampled Gaussian filter output (instead of *Overlay*) and image data is read from R1. We will provide an ImageBufferWriter which stores a test image in SRAM for you. (If your Checkpoint2 did not work correctly, we will provide a reference design you can use.)



(Note that the final project checkoff will closely follow the Checkpoint3 design, so we encourage you to go beyond this filter test and get a head start on implementing the rest of the hierarchy of filters and the difference of Gaussians.)

Design Specification for Checkpoint4

1. A 5x5 separable window FIR filter for the Gaussian blurring. (This means a 5 tap FIR filter for row filtering followed by a 5 tap filter for column filtering. 8 bit filter parameters available on Piazza.) The filter output should be renormalized to 8 bits.
2. Add a switch (or timer) to alternately display the original or filtered image.
3. The filter block should work with a 400x300 image, and can then run at a slower clock rate than the pixel clock rate. (The DCM module may be useful for generating a slower clock.)
4. You will need to down-sample the 800x600 image after it is read from in SRAM, and up-sample back to 800x600 for storing in SRAM and displaying on DVI.

Checkpoint4 Goals

All code should be pushed to GitHub. By the end of this checkpoint, you should be able to show:

1. (5 pts) The FPGA programmed with a design that selectively displays either the original or Gaussian filtered image over the DVI interface. *Report resource utilization and critical path from the Xilinx tools.*
2. (3 pts) Testbench for Gaussian filter block, showing processing of 5 rows, including end-of-row and end-of-frame.
3. (0.5 pt) Detailed block diagram for final Gaussian filter block. Level of detail should be registers, muxes, arithmetic elements, etc. Specify widths of all data paths. (This is updated version of block diagram from Checkpoint3.)
4. (0.5 pt) Identify in a separate list all the input control signals and output status signals from the datapath. (Updated version from Checkpoint3.)
5. (0.5 pt) Describe detailed operation sequence for the image processing pipeline in RT Language. (Updated version from Checkpoint3.)
6. (0.5 pt) State diagrams for the controller for the Gaussian filter block, including any needed interfaces with frame buffer. (Updated version from Checkpoint3.)

Choose a checkoff slot on the Doodle poll for Nov. 19-22.
Please bring a printout of your design documents to the checkoff to hand in.