*Shashank Kotturi (vskottur), Samiur Rahman (s72rahma), Dong-Hyun Kim d73kim, Nosheen Adil (n2adil)*

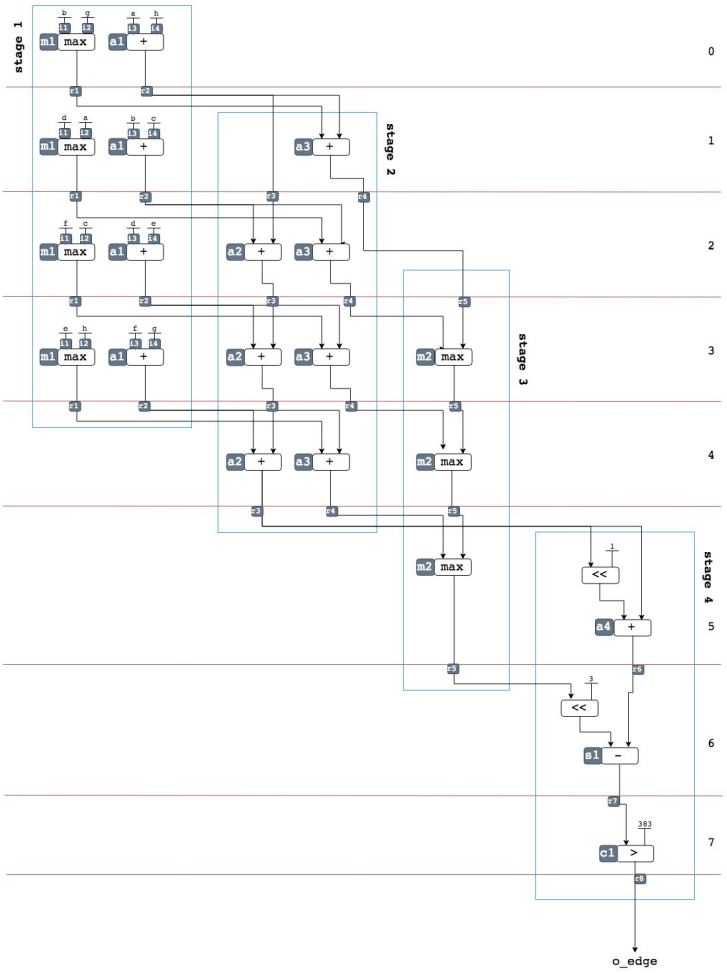
# ECE 327 Project Design Report

## Design Goals and Strategy

The design goal for this project was to achieve high performance. This goal was chosen upon discovering that the DFD can be designed to achieve maximum clock speed. The clock speed was maximized by scheduling operations to occur such that, at most one operation occurs in a clock cycle (ie. add, max, sub, cmp). Taking advantage of the concurrent nature of hardware, this was achieved with a 4-stage overlapping pipeline.

The majority of the optimality is due to the initial DFD design. Consequently, major additional revisions were not required in order to boost the optimality. A notable change was made however, involving the width of several components to account for arithmetic overflow. In the initial design, it was wrongfully assumed that all operations will produce 8-bit wide results, resulting in overflow for operations like addition and subtraction. This error was resolved in the next design iteration.

The following image is a DFD of the final design. This DFD demonstrates the maximized clock speed that the design aimed to achieve with the 4-stage overlapping pipeline.



## Performance Results vs. Projections

The performance estimates at the outset of our design was:

- Area of Major Components: 131 cells (Estimated Total Area: 131 + 160 = 291 cells)
- Maximum Clock Speed: 336 MHz
- Optimality: 1154

The performance results are:

- Area of Major Components: 117 cells (Total Area: 217 cells)
- Maximum Clock Speed: 280 MHz
- Optimality: 1290

Additionally, according to timing reports, the critical path was:

```
INFO: CRITICAL PATHS (slowest 5 paths, estimated by logic synthesis)
INFO: .....Delays.....
INFO:  Total  Datapath  Routing        Source           Dest
INFO: ------ -------- -------  -------------------- --------------------
INFO:   3.57     3.42     0.15    v(2)                r1_ty(0)/d
```

The final system has a latency of 8 and throughput of ¼.

The initial DFD design stage suggested that the design should attempt to limit the number of registers, which explains the reuse of one of the last two registers in the DFD to act as o_edge.

However, the following points were discovered during implementation:

- Multiplexers are expensive in terms of area.
- Registers with multiplexers on the input also decreased our clock speed.
- The design had more LUTs than registers; there were “free” registers to use without having to sacrifice on area.
- Adding a one-bit register is worth less complicated control circuitry

Since o_edge is already provided as an output signal, it was decided that it should be used as a register for the final compare operation. This reduced the number of 2:1 multiplexers in the design, therefore reducing the overall area, as well as increasing the clock speed.

Additionally, over the course of the design implementation, maintaining an ideal latency of 8 clock cycles proved to be a challenge. The measurement of latency begins when the pixel belonging in the second row and second column of the convolution table arrives. The convolution table is a set of registers and the DFD reads from the registered values for the first 4 clock cycles of the computation. So, one must wait one clock cycle before the DFD can begin reading from the convolution table. This resulted in a latency of 9 clock cycles.

In order to solve this problem, the initial plan involved pushing the subtract and compare operations at the end of the DFD together into a single clock cycle. However, this change resulted in a reduction in the optimality, from a score of 1270 (before penalty of latency > 8) to 1100. This optimization attempt was unsuccessful, considering it substantially reduced the maximum clock speed.

The solution to this problem was discovered soon after, when the pattern of reading from the convolution table registers in clock cycle 0 of the DFD was realized. Taking into account the priority of the eight directions, each one is computed from highest to lowest priority (W, NW, N, NE, E, SE, S, SW). In clock cycle 0, W and NW are computed, meaning inputs a, h, b and g from the convolution table are read into the DFD. In this same clock cycle, column 2 of the convolution table is registering its values; these are inputs c, d, and e. Therefore, none of the inputs being written into the convolution table in clock cycle 0 of the computation are read into the DFD. This indicates that the operation can occur concurrently. Once this change was made in the design, an ideal latency of 8 clock cycles was achieved. The optimality score increased from 1270 (before penalty of latency > 8) to 1290 as a result.

### Verification Plan and Test Cases

The first step of the verification plan was to test the design on a smaller 8x8 image input test case. This test case helped to verify the functional correctness of our design which included:

- Reading and writing of memory.
- Reading and writing of the convolution table.
- DFD computations and their outputs (o_valid, o_edge, o_dir)
- Operating mode (idle, busy and reset)
- Row and column counters

Once verified, the design was tested using larger 256x256 images. Testing commenced with the diff_ted command - this helped identify that the registers of the convolution table which were being read as inputs into the DFD had been switched. As a result, the differences between the .sim file and the .spec file consisted of edges where S and SW always differed.

Testing using the physical FPGA took place after achieving a perfect simulation. During several test runs, another interesting bug was discovered. The first image tested would run perfectly with a 0% error. However, if another image was sent after the initial, an error of 1% would occur. In other words, each image resulted in a 0% error when a test run was executed after a reset, and sending consecutive images would result in an error. This helped identify a problem with how the memory counters were being set before a new image was sent to the circuit.

When the system is reset, the counters are reset to zero. After an image has finished processing both o_row and o_col remain at 255 as per project requirements. When the first pixel of the second image arrived, the circuit would read the first pixel into row 255 and column 255 - this happened while o_row and o_counter were being set back to zero. The counters need to be at zero when the first pixel arrives but it's impossible to know when that will take place. The issue was resolved once it had been discovered that only o_col (along with a wr_en signal) determine where a pixel is written to in memory. The other counter, o_row, simply keeps track of the image processing progress. Since the project requirements only ask for o_row to remain at 255 until a new image arrives, it was determined that o_col could be used however the group sees fit. Therefore, once the system finishes processing an image, o_col is immediately set to zero while o_row is set back to zero when the first pixel of the next image arrives.