Overview

● Aim
● Design
  ○ Neural Network Structure
  ○ Hardware
  ○ Software
● Results
● Lesson Learned
Aim

Our project focuses on using convolutional neural network for image processing. Specifically we would like to recreate an image in an artistic style. The output image is created through convolutional neural network that recognizes the content of the image and applies the style of a separate image.

Our aim is to accelerate this Very Deep Convolutional Neural Network by implementing layers of the network in hardware and allow a software program to interface between these layers.
VGG network

Our project implemented the convolution and average pool functions as well as additional units to handle the back-propagation
Top level

- 64 bit ISA
- Memory control instruction
  - High 32 bits: 1 bit (mem or ALU), 3 bits (buffer to read/write), 1 bit (reset), 26 bits DDR3 address
  - Low 32 bits: 8 bits (stride), 8 bits (rows), 16 bits (block)
- ALU instruction
  - High 32 bits: 1 bit (mem or ALU), 3 bits (buffer to read), 4 bits (output sub ID), 2 bits (input/output sub block), 2 bits (read row + whether it’s a row or column - also use input/output sub block), 1 bit reverse mask), 19 empty bits
  - Low 32 bits: 16 bits (input block ID), 16 bits (output block ID)
- 128 bit data - may used shared memory to make this transfer
Mem control Unit

- Read buffer 1
  - 256x256 buffer
  - Composed of 64 4x4RAMs
    - This is to access blocks of data in one clock cycle
    - Bit for padded read
- Main buffer used
- Reads image from DDR3 based on stride, row, block inputs
Mem control unit contd.

- Read buffer 2
  - 256x128 buffer
  - Composed of 4x4 RAMs
  - Bit for padded read
- Secondary buffer - only used for Gram matrix calculations and
- Reads from DDR3
Mem control unit contd.

- Mask buffer
  - 4x4 buffer
- Reads mask from memory
Write back accumulator

- Write buffer
  - 256x256
  - Composed of 16 dual-port rams
    - To write/accumulate in one cycle
- Writes results back to DDR3 once accumulation is done
64 RAMs

- Otherwise takes 1-2 hours to compile
  - Single-cycle 4x4 block access too complicated for altera to optimize/infer RAM
- Single-cycle access made possible with striding

<p>| | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ALU

- Takes in 1 64-bit input as the instruction set along with 6x6 matrix of 32 bits.
- If first bit is high instruction set goes to ALU
- 3 bits encode which buffer to read/write from.
- 4 bits for output sub id from block
- 2 bits for input/output sub-block
- 2 bits enable read and encodes whether data is in rows or columns
- 1 bit rev mask
- 16 bits encode the block ID in memory for input
- 16 bits encode the block ID in memory for output
Multipliers

- 27-fixed point multiplications for ALU units
  - 1 bit signed
  - 14 bits integer
  - 13 bits fraction
- 112 multipliers on board
- 144 multipliers needed for 3x3 convolutions
- Solution: 112 hard multipliers, 22 soft multipliers
Additional Peripherals

To fully implement this project, a vga framebuffer was also implemented in the device. Pixel data is sent from CPU and stored on the framebuffer and displayed through the VGA capabilities on the FPGA.