Mid-semester Assignment

Neelam Sharma (18307R030)
EE 789: Algorithmic Design of Digital Systems

April 28, 2020

1 Introduction

An accelerator is designed to convolve 32×32(pixel) sized images with 4×4(pixel) sized kernel.

1.1 Problem Statement

An image is given with pixels $\{x_{i,j}: 0 \leq i < 32, 0 \leq j < 32\}$. Further, assume that you are given a kernel which is a 4x4 image $\{k_{i,j}: 0 \leq i < 4, 0 \leq j < 4\}$. Assume that the pixels and kernel values are coded as 16-bit unsigned integers. Ignore overflows. Let p take on values 0,1, . . . ,28, and q take on values 0,1, . . . ,28. The accelerator is supposed to compute the following numbers:

$$u_{p,q} = \sum_{i=0}^{3} \sum_{j=0}^{3} x_{(p+i),(q+j)} \times k_{i,j} \tag{1}$$

2 Design Decisions

2.1 Memory layout

1. Used a 61×32 memory in order to use it as a 2-D matrix for easy understanding of memory organization. Accelerator also uses 3 4×4 for it's processing.

$$\text{Total memory} = 3904B + (32 \times 3B) = 3.90625kB \tag{2}$$

2. Data width: 16 bit. Address width: 11-bit (6-bits for row selection and 5-bits for column selection).

3. Location of command: kernel is stored at a fixed location with following layout of memory addresses

$$\begin{bmatrix} mem_{filt\_addr,0} & mem_{filt\_addr,1} & mem_{filt\_addr,2} & mem_{filt\_addr,3} \\ mem_{filt\_addr+1,0} & mem_{filt\_addr+1,1} & mem_{filt\_addr+1,2} & mem_{filt\_addr+1,3} \\ mem_{filt\_addr+2,0} & mem_{filt\_addr+2,1} & mem_{filt\_addr+2,2} & mem_{filt\_addr+2,3} \\ mem_{filt\_addr+3,0} & mem_{filt\_addr+3,1} & mem_{filt\_addr+3,2} & mem_{filt\_addr+3,3} \end{bmatrix}$$

where filt_addr = start row address of a filter in shared memory = 32 (in my implementation).

4. Layout of incoming image in memory:

$$\begin{bmatrix} mem_{0,0} & mem_{0,1} & \cdots & mem_{0,31} \\ mem_{1,0} & mem_{1,1} & \cdots & mem_{1,31} \\ \vdots & \vdots & \ddots & \vdots \\ mem_{31,0} & mem_{31,1} & \cdots & mem_{31,31} \end{bmatrix}$$

   where $\text{mem}_{\text{i,j}}$ = Data stored at row-address i and column address j.

5. Layout of outgoing $u_{i,j}$:

$$\begin{bmatrix} mem_{0,0} & mem_{0,1} & \cdots & mem_{0,28} \\ mem_{1,0} & mem_{1,1} & \cdots & mem_{1,28} \\ \vdots & \vdots & \ddots & \vdots \\ mem_{28,0} & mem_{28,1} & \cdots & mem_{28,28} \end{bmatrix}$$

6. Coding of values on pipes: request[31:0] = <16,6,5,4,1> , response[15:0] =<16>.
   where, request[31:16] = 16-bit unsigned data, request[15:10] = row address, request[9:5] = column address, request[4:1] = unused bits, request[0:0] = rwbar (0 for writing into the memory and 1 for reading from it).
   response[15:0] = 16-bit data read from memory corresponding to the request passed.

7. Command counter: It is a counter value (16-bit) whose value provides the information of which image/command no. will be processed next by accelerator.
   Location: $\text{mem}_{32,31}$

8. Status of accelerator: 16-bit value that takes value 1 when accelerator has finished executing for current image otherwise it takes a 0 value.
   Location: $\text{mem}_{32,30}$.

9. Parallelization Scheme implemented: Used two engines that process two consecutive rows concurrently (i.e., one on even-valued row no.s and other one on odd-valued rows).

3 Block Diagram

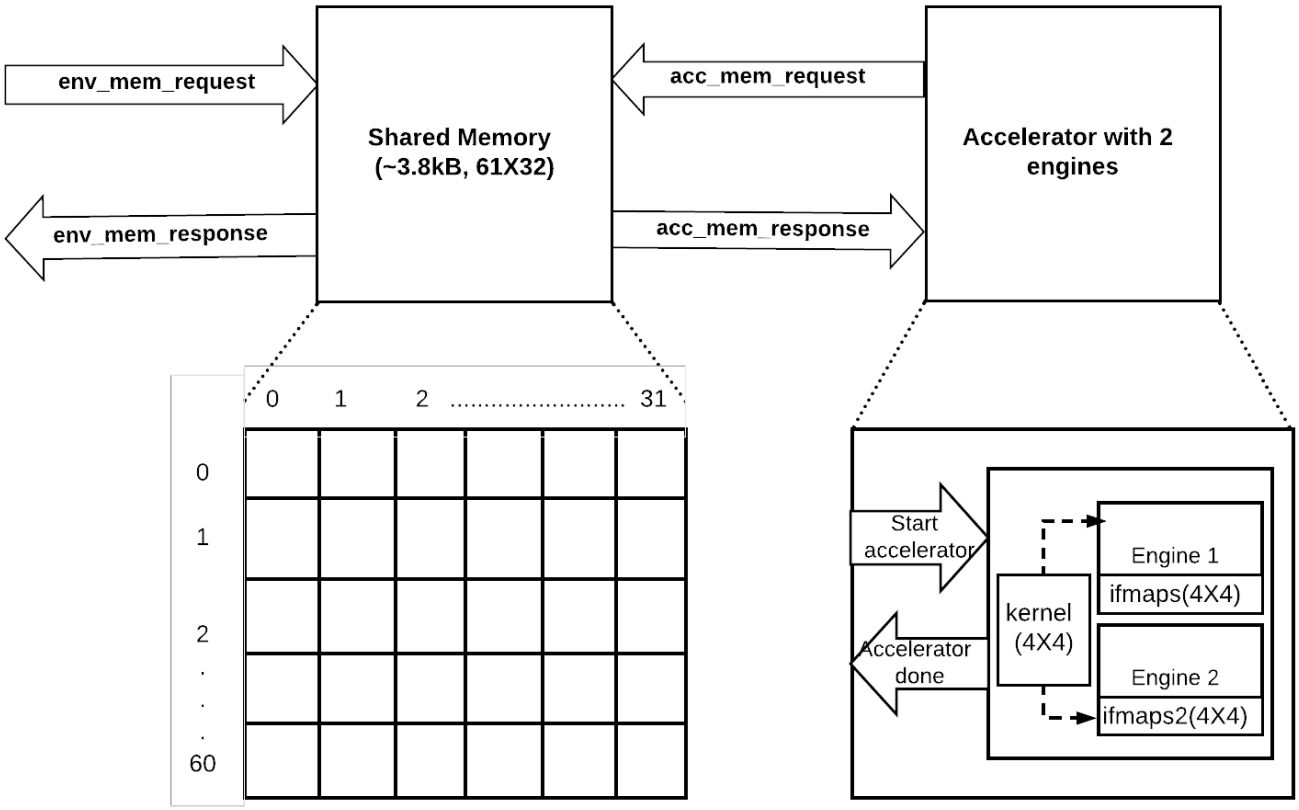

Figure 1: Overall architecture

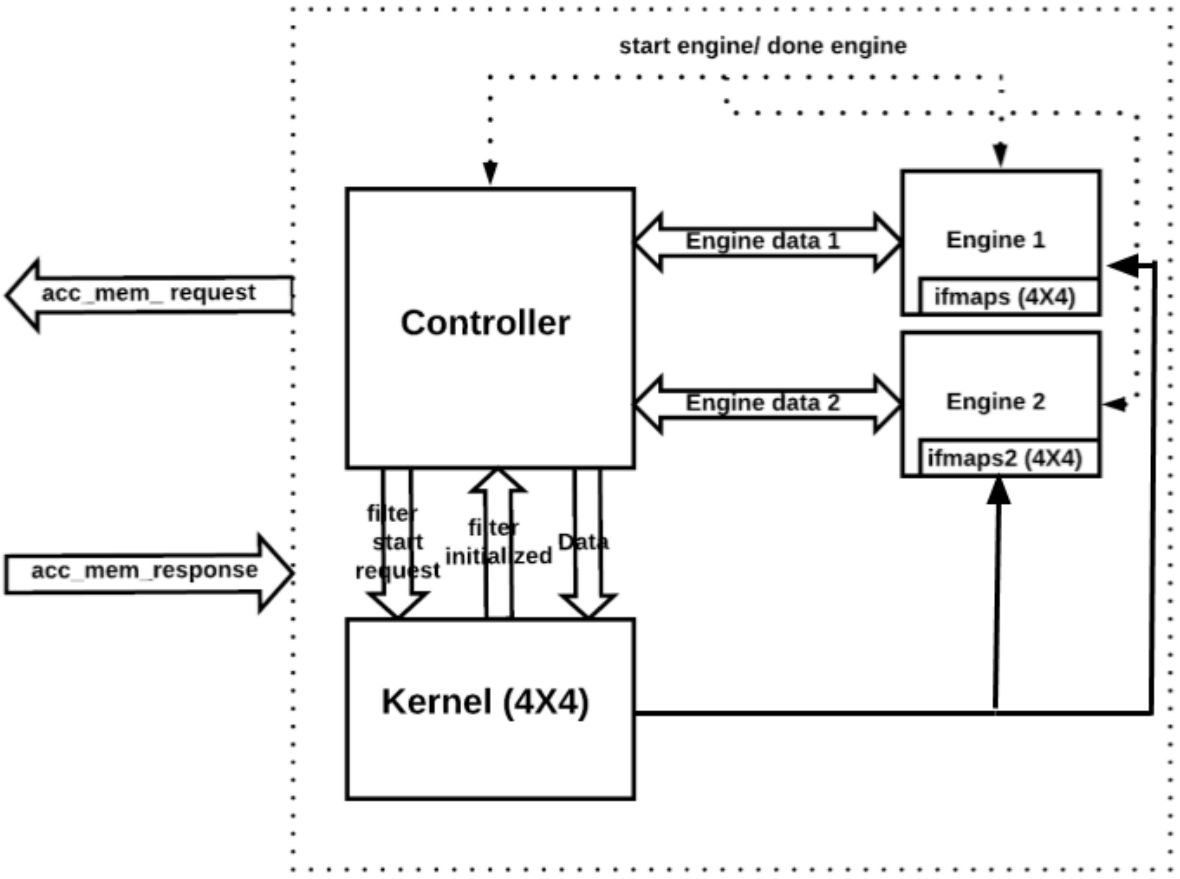

Figure 2: Accelerator architecture

4 Program Flow

When the accelerator is started following steps execute:

1. Initialize 4×4 kernel (It is executed only once for a given command).
2. For each new row of $u_{i,j}$ ifmaps (4×4) and ifmaps2 (4×4) are initialized from memory (It runs once for a new row) and each engine. proceedes by fetching new column for a given row from memory.
3. Each $u_{i,j}$ is calculated by it's row respective engine, i.e., Engine 1 for even-valued rows and Engine 2 for odd-valued rows.
4. $u_{i,j}$'s computed in step 3 are stored back into the memory.
5. Accelerator writes into the accelerator_done pipe indicating that it has finished computing and has written $u_{i,j}$'s into the memory.

5 Results

For a given kernel and image pixels values as given below:

$$kernel = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad image = \begin{bmatrix} 0 & 1 & 2 & 3 & \cdots & 31 \\ 0 & 1 & 2 & 3 & \cdots & 31 \\ 0 & 1 & 2 & 3 & \cdots & 31 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & 2 & 3 & \cdots & 31 \end{bmatrix}$$

Expected $u_{i,j}$ is:

$$u = \begin{bmatrix} 6 & 10 & 14 & 18 & \cdots & 118 \\ 6 & 10 & 14 & 18 & \cdots & 118 \\ 6 & 10 & 14 & 18 & \cdots & 118 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 6 & 10 & 14 & 18 & \cdots & 118 \end{bmatrix}$$

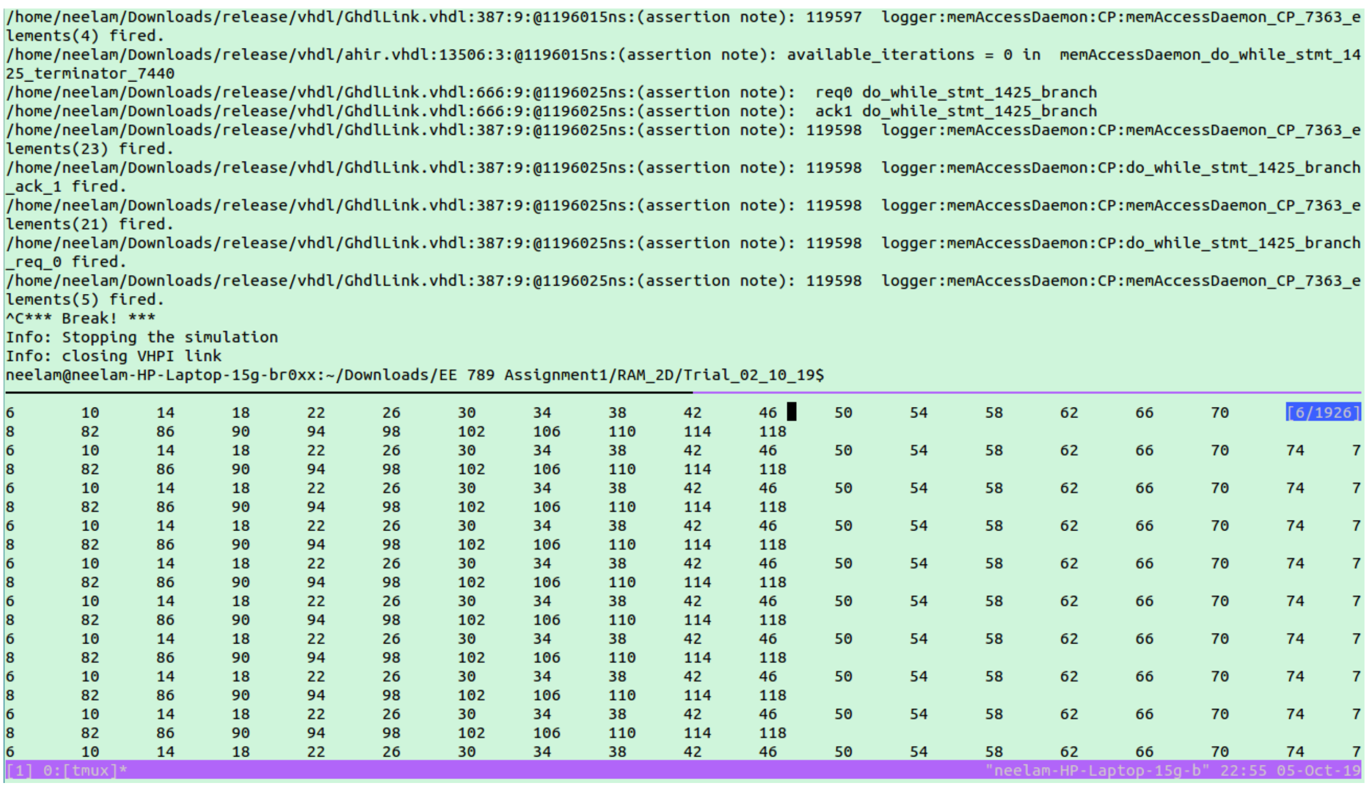

Figure 3: Terminal output