EE789: Accelerator

Dimple Kochar- Roll Number 16D070010

October 6, 2019

1 Overview of the assignment

We are designing an accelerator which will perform dot-products of a data (image) matrix, with a kernel. Suppose there is an image with pixels
$x_{i,j} : 0 \leq i < 32, 0 \leq j < 32$. Further, our kernel is a 4x4 image
$k_{i,j} : 0 \leq i < 4, 0 \leq j < 4$. The pixels and kernel values are coded as 16-bit unsigned integers, and overflows are ignored. 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:

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

2 Design/Algorithm

The accelerator starts functioning on obtaining a start signal from a pipe. Once that is sent, it makes sends a status = 0 through a pipe (and also writes 0 to the status register). It then fetches the kernel and image and computes the dot product and stores it where the image was. It then sends a status = 1 signal (and writes to status register in memory), so that outside world can take out the computed $u_{i,j}$. Command counter is then checked, and if it is non-zero, the accelerator decrements it and starts working on another image. The assignment is basically divided in 3 major tasks:
1) To fetch memory efficiently
2) To compute dot product
3) To store it back in memory

As memory takes time to be accessed, we need to fetch an optimum amount so as to not take up to much storage as we've a constraint on memory and not let it become a bottleneck either. What I have done is that I fetch a 4*4 block initially by 4 memory accesses. Then, since the latter 3 columns will be used for the next computation, I store, I fetch only one more column for the next computation and so on.
My design decisions:
1) I have a 1536 length array of 16 bit unsigned int as memory (3KB) mem[1536]. 0-1023 are used as incoming image. 1024-1039 are used as kernel. 1040 is used as command counter. 1041 is used as status register. Outgoing image is written in place of incoming image. The layout of $x_{i,j}$ and $u_{i,j}$ is mem[32*i+j]. I am keeping an address width of 12 bits (11 bits required but initially taken).
2) I have implemented parallelism through pipelining of innermost nested blocks in the code.

3 Design implementation

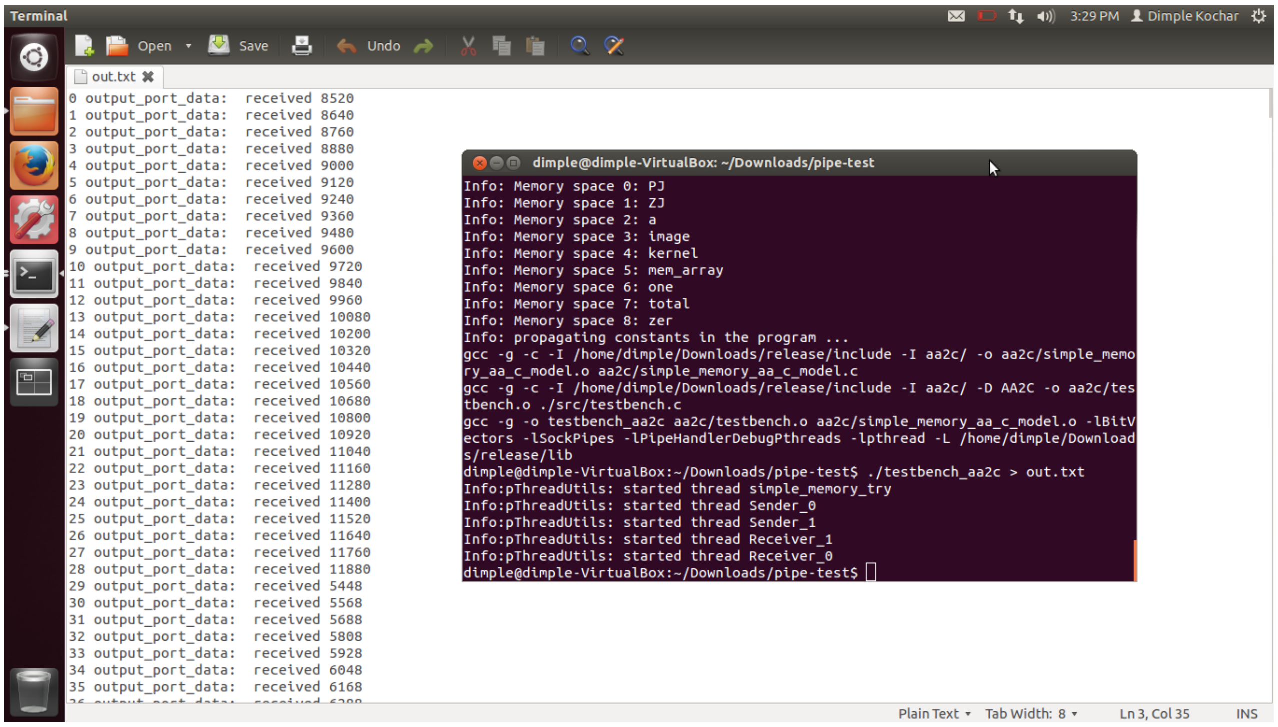
1) We initialize memory with numbers (our image and kernel).
2) Accelerator waits till the start signal arrives and once it arrives, sets status $= 0$.
3) It first fetches kernel and stores it in an array of 16 unsigned ints.
4) Then it starts with the first row of image. It initially fetches 4*4 block and computes $u_{1,1}$
5) Using the previously fetched columns, it then just fetches one 4*1 column and computes $u_{1,2}$. this continues till $u_{1,28}$ is computed.
6) This repeats for all 29 rows. And the image is stored right where $p_{i,j}$ was.
7) It then sets status $= 1$ and checks if the command counter is not zero. If yes, it again starts the process. Else, it stops.

Testbench was made with obtaining status, output image with the address of output image as outputs from the pipe, while feeding in a start signal. Command counter can be changed in code or a pipe can be used (can be changed depending on user).

4 Results

On running aa2c, ./testbench_aa2c, we get the right results for 1 or more images.

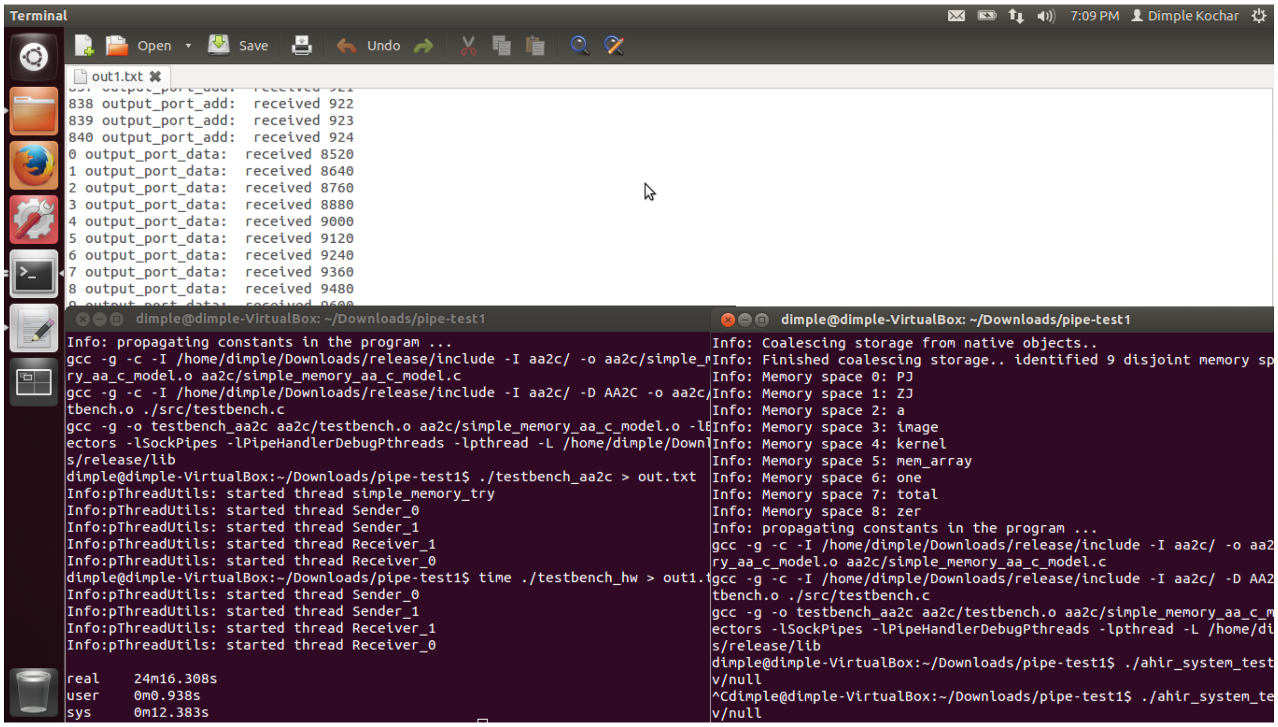
VHDLsim too gives us the correct results. Pipelined version takes lesser time compared to non-pipelined as expected.Non-pipelined takes around 30 minutes of simulation time whereas pipelined around 23 minutes.



```
0 output_port_data:  received 8520
1 output_port_data:  received 8640
2 output_port_data:  received 8760
3 output_port_data:  received 8880
4 output_port_data:  received 9000
5 output_port_data:  received 9120
6 output_port_data:  received 9240
7 output_port_data:  received 9360
8 output_port_data:  received 9480
9 output_port_data:  received 9600
10 output_port_data:  received 9720
11 output_port_data:  received 9840
12 output_port_data:  received 9960
13 output_port_data:  received 10080
14 output_port_data:  received 10200
15 output_port_data:  received 10320
16 output_port_data:  received 10440
17 output_port_data:  received 10560
18 output_port_data:  received 10680
19 output_port_data:  received 10800
20 output_port_data:  received 10920
21 output_port_data:  received 11040
22 output_port_data:  received 11160
23 output_port_data:  received 11280
24 output_port_data:  received 11400
25 output_port_data:  received 11520
26 output_port_data:  received 11640
27 output_port_data:  received 11760
28 output_port_data:  received 11880
29 output_port_data:  received 5448
30 output_port_data:  received 5568
31 output_port_data:  received 5688
32 output_port_data:  received 5808
33 output_port_data:  received 5928
34 output_port_data:  received 6048
35 output_port_data:  received 6168
```

```
dimple@dimple-VirtualBox: ~/Downloads/pipe-test
Info: Memory space 0: PJ
Info: Memory space 1: ZJ
Info: Memory space 2: a
Info: Memory space 3: image
Info: Memory space 4: kernel
Info: Memory space 5: mem_array
Info: Memory space 6: one
Info: Memory space 7: total
Info: Memory space 8: zer
Info: propagating constants in the program ...
gcc -g -c -I /home/dimple/Downloads/release/include -I aa2c/ -o aa2c/simple_memo
ry_aa_c_model.o aa2c/simple_memory_aa_c_model.c
gcc -g -c -I /home/dimple/Downloads/release/include -I aa2c/ -D AA2C -o aa2c/tes
tbench.o ./src/testbench.c
gcc -g -o testbench_aa2c aa2c/testbench.o aa2c/simple_memory_aa_c_model.o -lBitV
ectors -lSockPipes -lPipeHandlerDebugPthreads -lpthread -L /home/dimple/Download
s/release/lib
dimple@dimple-VirtualBox:~/Downloads/pipe-test$ ./testbench_aa2c > out.txt
Info:pThreadUtils: started thread simple_memory_try
Info:pThreadUtils: started thread Sender_0
Info:pThreadUtils: started thread Sender_1
Info:pThreadUtils: started thread Receiver_1
Info:pThreadUtils: started thread Receiver_0
dimple@dimple-VirtualBox:~/Downloads/pipe-test$
```

aa2c simulation.



```
838 output_port_add:  received 922
839 output_port_add:  received 923
840 output_port_add:  received 924
0 output_port_data:  received 8520
1 output_port_data:  received 8640
2 output_port_data:  received 8760
3 output_port_data:  received 8880
4 output_port_data:  received 9000
5 output_port_data:  received 9120
6 output_port_data:  received 9240
7 output_port_data:  received 9360
8 output_port_data:  received 9480
```

```
Info: propagating constants in the program ...
gcc -g -c -I /home/dimple/Downloads/release/include -I aa2c/ -o aa2c/simple_m
ry_aa_c_model.o aa2c/simple_memory_aa_c_model.c
gcc -g -c -I /home/dimple/Downloads/release/include -I aa2c/ -D AA2C -o aa2c/
tbench.o ./src/testbench.c
gcc -g -o testbench_aa2c aa2c/testbench.o aa2c/simple_memory_aa_c_model.o -lE
ectors -lSockPipes -lPipeHandlerDebugPthreads -lpthread -L /home/dimple/Downl
s/release/lib
dimple@dimple-VirtualBox:~/Downloads/pipe-test1$ ./testbench_aa2c > out.txt
Info:pThreadUtils: started thread simple_memory_try
Info:pThreadUtils: started thread Sender_0
Info:pThreadUtils: started thread Sender_1
Info:pThreadUtils: started thread Receiver_1
Info:pThreadUtils: started thread Receiver_0
dimple@dimple-VirtualBox:~/Downloads/pipe-test1$ time ./testbench_hw > out1.t
Info:pThreadUtils: started thread Sender_0
Info:pThreadUtils: started thread Sender_1
Info:pThreadUtils: started thread Receiver_1
Info:pThreadUtils: started thread Receiver_0

real    24m16.308s
user    0m0.938s
sys     0m12.383s
```

```
Info: Coalescing storage from native objects..
Info: Finished coalescing storage.. identified 9 disjoint memory sp
Info: Memory space 0: PJ
Info: Memory space 1: ZJ
Info: Memory space 2: a
Info: Memory space 3: image
Info: Memory space 4: kernel
Info: Memory space 5: mem_array
Info: Memory space 6: one
Info: Memory space 7: total
Info: Memory space 8: zer
Info: propagating constants in the program ...
gcc -g -c -I /home/dimple/Downloads/release/include -I aa2c/ -o aa2
ry_aa_c_model.o aa2c/simple_memory_aa_c_model.c
gcc -g -c -I /home/dimple/Downloads/release/include -I aa2c/ -D AA2
tbench.o ./src/testbench.c
gcc -g -o testbench_aa2c aa2c/testbench.o aa2c/simple_memory_aa_c_m
ectors -lSockPipes -lPipeHandlerDebugPthreads -lpthread -L /home/di
s/release/lib
dimple@dimple-VirtualBox:~/Downloads/pipe-test1$ ./ahir_system_test
v/null
^Cdimple@dimple-VirtualBox:~/Downloads/pipe-test1$ ./ahir_system_te
v/null
```

vhdlsim