Algorithmic Design of Digital Systems Matrix-Vector Product (MVP) Unit

Mitul Tyagi, Roll no. : 193079017

10th December 2020

1 Introduction

A matrix multiplier unit (MVP) is designed, implemented and verified. The MVP has a single input pipe and a single output pipe. A 32x32 matrix A whose entries are 32-bit unsigned integers is stored inside the MVP. Vectors of length 32 are supplied to the MVP using the input pipe. The MVP takes the input vector x and computes the matrix-vector product vector y. The generated output vector is sent out on the output pipe.

$$y = Ax$$

2 Target

A reference implementation of the MVP together with a test bench was provided. The goal was to design and implement an MVP which works at least 4-times faster than the reference implementation. That is, the maximum rate at which input vectors can be fed to the mvp_unit must be at least 4 times the rate at which the reference implementation performs the multiplication.

3 Interface of the MVP Unit

The interfaces to the MVP are shown below. At the beginning (after reset), the **32x32 Matrix A** is fed into the MVP unit in row major order. That is, the matrix entries are fed in the following order:

$$A[0][0]\ A[0][1]\ ....\ A[0][31]$$
$$A[1][0]\ A[1][1]\ ....\ A[1][31]$$
$$.......................................$$
$$A[31][0]\ A[31][1]\ ....\ A[31][31]$$

After the matrix entries have been fed in, the input vectors are streamed into the in data pipe. The input vector X is supplied in the following order:

$$x[0]\ x[1]\ x[2]\ ...\ x[31]$$

The input vector is used to computed the matrix vector product, and resulting vector y is streamed out of the out data pipe in the following order:

$$y[0]\ y[1]\ y[2]\ ...\ y[31]$$

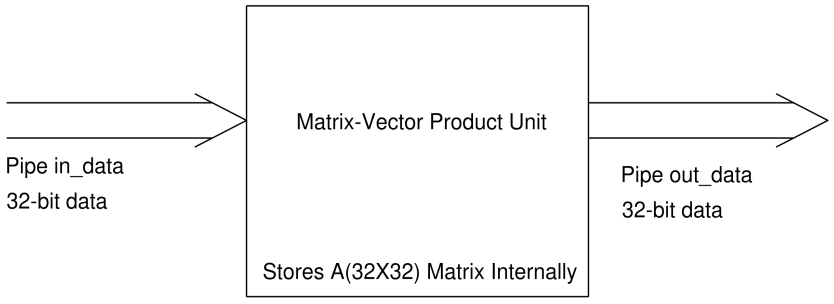


Figure 1: Block Diagram

4 Design Principles

The internal structure of the mvp_unit is shown in Figure 2 which indicates various modules and storage devices present and how they interact with each other.

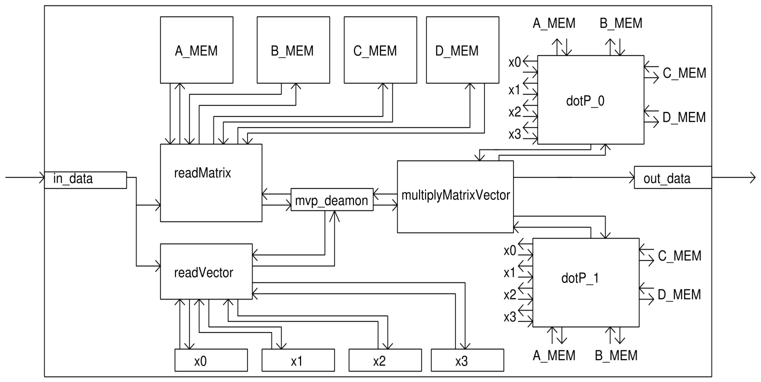


Figure 2: Internal Structure

Following are the various modules present in the **mvp_unit**:

4.1 mvp_daemon

This is the top-level module which interacts with the outside world and calls all the other modules for the multiplication of the vector X with the matrix A.

4.2 readMatrix Module

This module reads the matrix A from the input port and routes the packet to the four storage units A_MEM, B_MEM,C_MEM and D_MEM. The division of matrix A to these four storage devices is done column-wise in the following order:

$$A\_MEM: \; Col\_0, \; Col\_4, \; Col\_8, \; Col\_12, \; Col\_16, \; Col\_20, \; Col\_24, \; Col\_28$$

$$B\_MEM: \; Col\_1, \; Col\_5, \; Col\_9, \; Col\_13, \; Col\_17, \; Col\_21, \; Col\_25, \; Col\_29$$

$$C\_MEM: \; Col\_2, \; Col\_6, \; Col\_10, \; Col\_14, \; Col\_18, \; Col\_22, \; Col\_26, \; Col\_30$$

$$D\_MEM: \; Col\_3, \; Col\_7, \; Col\_11, \; Col\_15, \; Col\_19, \; Col\_23, \; Col\_27, \; Col\_31$$

4.3 readVector Module

This module reads the vector X from the input port and routes the packet to the four storage units x0, x1,x2, and x3. The division of vector X to these four storage devices is done row-wise in the following order:

$$x0: \; Row\_0, \; Row\_4, \; Row\_8, \; Row\_12, \; Row\_16, \; Row\_20, \; Row\_24, \; Row\_28$$

$$x1: \; Row\_1, \; Row\_5, \; Row\_9, \; Row\_13, \; Row\_17, \; Row\_21, \; Row\_25, \; Row\_29$$

$$x2: \; Row\_2, \; Row\_6, \; Row\_10, \; Row\_14, \; Row\_18, \; Row\_22, \; Row\_26, \; Row\_30$$

$$x3: \; Row\_3, \; Row\_7, \; Row\_11, \; Row\_15, \; Row\_19, \; Row\_23, \; Row\_27, \; Row\_31$$

4.4 multiplyMatrixVector Module

This module calls the two computational modules by providing the value of the Row(R) that has to be computed. The result which it receives from those modules is then sent to the pipe.

4.5 dotP_0 Module and dotP_1 Module

These two modules are the core computational modules that perform the actual multiplication. Each module interacts with all the 8 storage units to perform multiplication and returns the result for each Row. The **dotP_0 Module** performs the multiplication of all the *Even Rows* and the **dotP_1 Module** performs the multiplication of all the *Odd Rows*. Each module unrolls the loop by 4 and hence 4 parallel units are working together for each row. Both the modules are pipelined too.

4.6 C Testbench

The C Testbench validates for the correctness of the multiplication. The design tools convert this Aa Code to the VHDL Code and the simulation was performed using **GHDL**. This testbench helps in validation and functional verification of the system.

4.7 Design Parameters

Following design parameters were used while performing simulation:

- Depth of Pipes at input and output ports is 3.
- Loop Unrolling factor for the two computational modules is 4.

5 Compiling the Project

It involves following three main steps:

- Compile the Aa Files to generate the C files and VHDL files for testing and simulation. Run the script **compile.sh** present in the **hw folder** .
- Compile the testbench file using the script **build_aa2c_tb.sh**.
- Build **GHDL Model** and **GHDL Testbench**.

A bash script **build.sh** can alone be used to run all the steps mentioned above

6 Running the Project

To verify using the C Testbench run the command **./bin/testbench_aa2c** with appropriate inputs. To verify using the VHDL Simulation run the command **./bin/testbench_vhdl** followed by **./ahir_system_test_bench** with appropriate inputs. The simulation result can be dumped to a file and can be viewed using **GTKWave Viewer**.

7 Performance Analysis

The VHDL Simulation was done using the GHDL Software and following are the comparison to the reference implementation are as follows:

Design	Load Matrix A(ns)	Total-Time(ns)	Cycles per Vector
Reference Model	83285	2527445	7634
New Design	103765	485845	1190

Performance Speed Up=7634/1190=6.4

8 Results

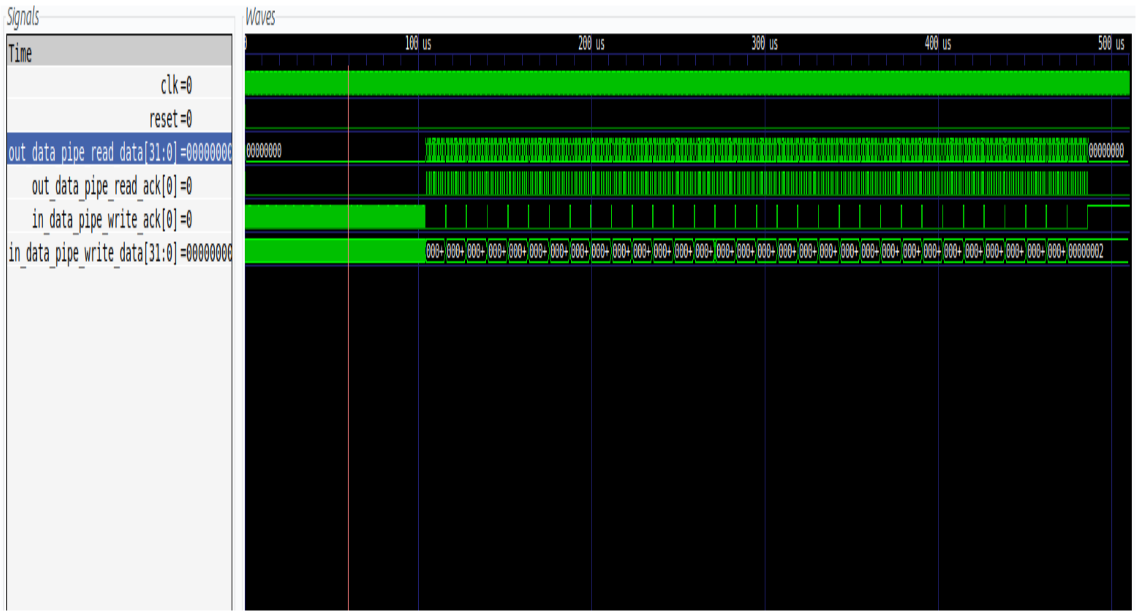
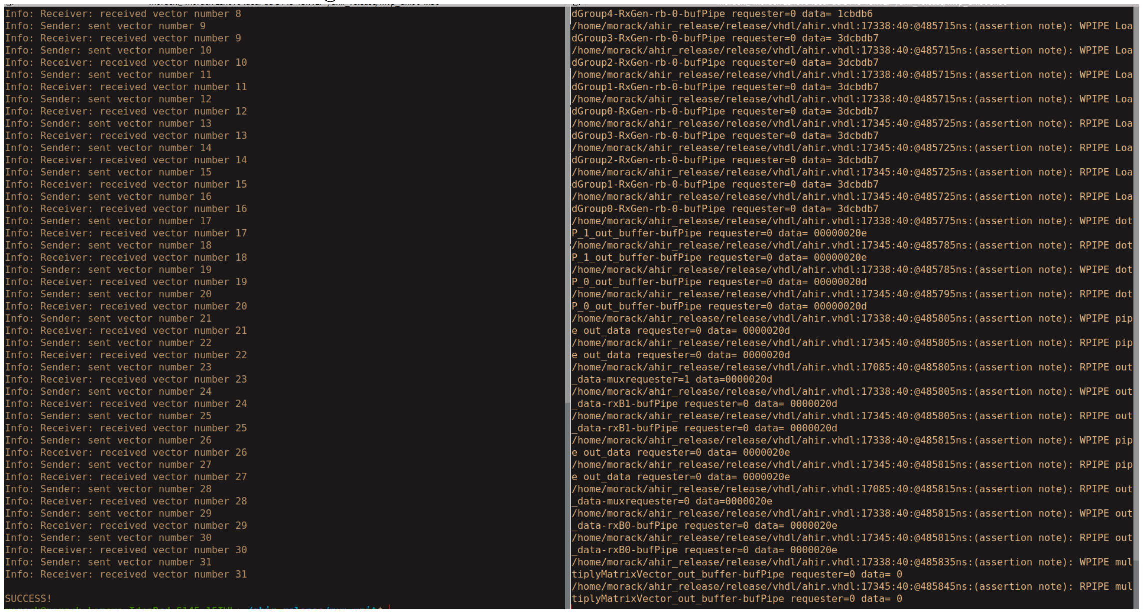

Signals
Waves
Time
clk=0
reset=0
out data pipe read data[31:0]=00000000
out data pipe read ack[0]=0
in data pipe write ack[0]=0
in data pipe write data[31:0]=00000000
100 us
200 us
300 us
400 us
500 us
00000000
00000002

Figure 3: GHDL Simulation

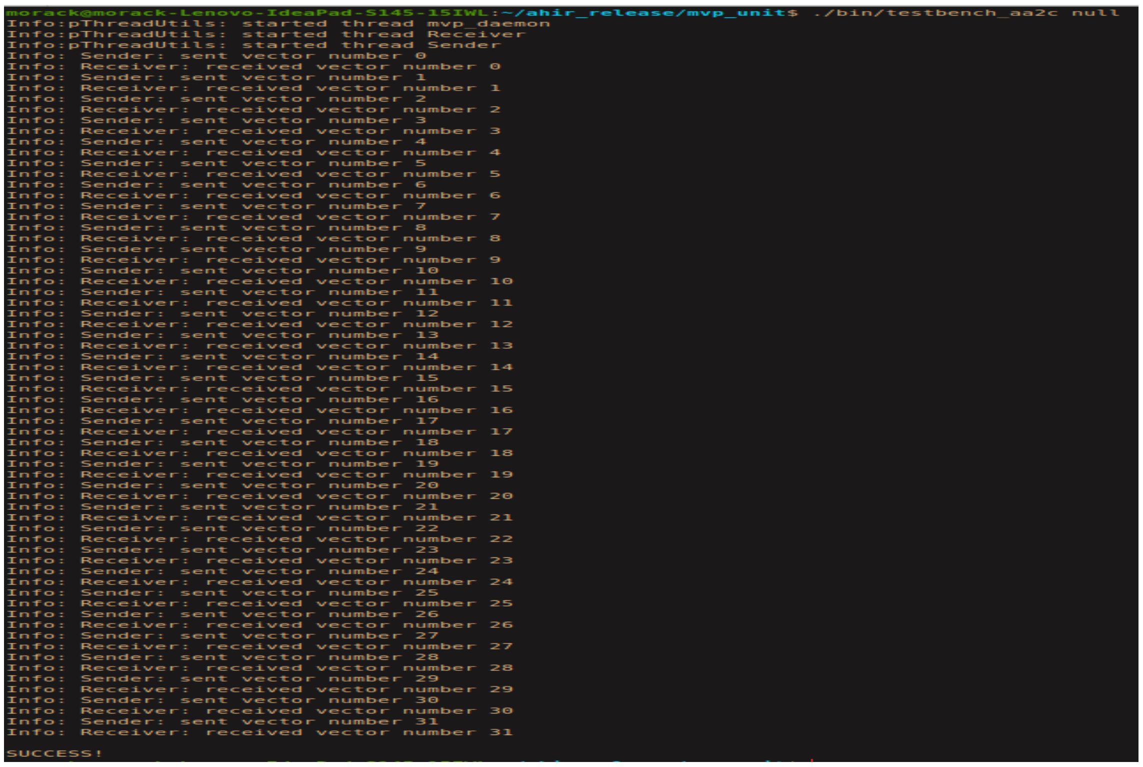


```
Info: Receiver: received vector number 8
Info: Sender: sent vector number 9
Info: Receiver: received vector number 9
Info: Sender: sent vector number 10
Info: Receiver: received vector number 10
Info: Sender: sent vector number 11
Info: Receiver: received vector number 11
Info: Sender: sent vector number 12
Info: Receiver: received vector number 12
Info: Sender: sent vector number 13
Info: Receiver: received vector number 13
Info: Sender: sent vector number 14
Info: Receiver: received vector number 14
Info: Sender: sent vector number 15
Info: Receiver: received vector number 15
Info: Sender: sent vector number 16
Info: Receiver: received vector number 16
Info: Sender: sent vector number 17
Info: Receiver: received vector number 17
Info: Sender: sent vector number 18
Info: Receiver: received vector number 18
Info: Sender: sent vector number 19
Info: Receiver: received vector number 19
Info: Sender: sent vector number 20
Info: Receiver: received vector number 20
Info: Sender: sent vector number 21
Info: Receiver: received vector number 21
Info: Sender: sent vector number 22
Info: Receiver: received vector number 22
Info: Sender: sent vector number 23
Info: Receiver: received vector number 23
Info: Sender: sent vector number 24
Info: Receiver: received vector number 24
Info: Sender: sent vector number 25
Info: Receiver: received vector number 25
Info: Sender: sent vector number 26
Info: Receiver: received vector number 26
Info: Sender: sent vector number 27
Info: Receiver: received vector number 27
Info: Sender: sent vector number 28
Info: Receiver: received vector number 28
Info: Sender: sent vector number 29
Info: Receiver: received vector number 29
Info: Sender: sent vector number 30
Info: Receiver: received vector number 30
Info: Sender: sent vector number 31
Info: Receiver: received vector number 31

SUCCESS!
```

```
dGroup4-RxGen-rb-0-bufPipe requester=0 data= 1cbdb6
/home/morack/ahir_release/release/vhdl/ahir.vhdl:17338:40:@485715ns:(assertion note): WPIPE Loa
dGroup3-RxGen-rb-0-bufPipe requester=0 data= 3dcbdb7
/home/morack/ahir_release/release/vhdl/ahir.vhdl:17338:40:@485715ns:(assertion note): WPIPE Loa
dGroup2-RxGen-rb-0-bufPipe requester=0 data= 3dcbdb7
/home/morack/ahir_release/release/vhdl/ahir.vhdl:17338:40:@485715ns:(assertion note): WPIPE Loa
dGroup1-RxGen-rb-0-bufPipe requester=0 data= 3dcbdb7
/home/morack/ahir_release/release/vhdl/ahir.vhdl:17338:40:@485715ns:(assertion note): WPIPE Loa
dGroup0-RxGen-rb-0-bufPipe requester=0 data= 3dcbdb7
/home/morack/ahir_release/release/vhdl/ahir.vhdl:17345:40:@485725ns:(assertion note): RPIPE Loa
dGroup3-RxGen-rb-0-bufPipe requester=0 data= 3dcbdb7
/home/morack/ahir_release/release/vhdl/ahir.vhdl:17345:40:@485725ns:(assertion note): RPIPE Loa
dGroup2-RxGen-rb-0-bufPipe requester=0 data= 3dcbdb7
/home/morack/ahir_release/release/vhdl/ahir.vhdl:17345:40:@485725ns:(assertion note): RPIPE Loa
dGroup1-RxGen-rb-0-bufPipe requester=0 data= 3dcbdb7
/home/morack/ahir_release/release/vhdl/ahir.vhdl:17345:40:@485725ns:(assertion note): RPIPE Loa
dGroup0-RxGen-rb-0-bufPipe requester=0 data= 3dcbdb7
/home/morack/ahir_release/release/vhdl/ahir.vhdl:17338:40:@485775ns:(assertion note): WPIPE dot
P_1_out_buffer-bufPipe requester=0 data= 00000020e
/home/morack/ahir_release/release/vhdl/ahir.vhdl:17345:40:@485785ns:(assertion note): RPIPE dot
P_1_out_buffer-bufPipe requester=0 data= 00000020e
/home/morack/ahir_release/release/vhdl/ahir.vhdl:17338:40:@485785ns:(assertion note): WPIPE dot
P_0_out_buffer-bufPipe requester=0 data= 00000020d
/home/morack/ahir_release/release/vhdl/ahir.vhdl:17345:40:@485795ns:(assertion note): RPIPE dot
P_0_out_buffer-bufPipe requester=0 data= 00000020d
/home/morack/ahir_release/release/vhdl/ahir.vhdl:17338:40:@485805ns:(assertion note): WPIPE pip
e out_data requester=0 data= 0000020d
/home/morack/ahir_release/release/vhdl/ahir.vhdl:17345:40:@485805ns:(assertion note): RPIPE pip
e out_data requester=0 data= 0000020d
/home/morack/ahir_release/release/vhdl/ahir.vhdl:17085:40:@485805ns:(assertion note): RPIPE out
_data-muxrequester=1 data=0000020d
/home/morack/ahir_release/release/vhdl/ahir.vhdl:17338:40:@485805ns:(assertion note): WPIPE out
_data-rxB1-bufPipe requester=0 data= 0000020d
/home/morack/ahir_release/release/vhdl/ahir.vhdl:17345:40:@485805ns:(assertion note): RPIPE out
_data-rxB1-bufPipe requester=0 data= 0000020d
/home/morack/ahir_release/release/vhdl/ahir.vhdl:17338:40:@485815ns:(assertion note): WPIPE pip
e out_data requester=0 data= 0000020e
/home/morack/ahir_release/release/vhdl/ahir.vhdl:17345:40:@485815ns:(assertion note): RPIPE pip
e out_data requester=0 data= 0000020e
/home/morack/ahir_release/release/vhdl/ahir.vhdl:17085:40:@485815ns:(assertion note): RPIPE out
_data-muxrequester=0 data=0000020e
/home/morack/ahir_release/release/vhdl/ahir.vhdl:17338:40:@485815ns:(assertion note): WPIPE out
_data-rxB0-bufPipe requester=0 data= 0000020e
/home/morack/ahir_release/release/vhdl/ahir.vhdl:17345:40:@485815ns:(assertion note): RPIPE out
_data-rxB0-bufPipe requester=0 data= 0000020e
/home/morack/ahir_release/release/vhdl/ahir.vhdl:17338:40:@485835ns:(assertion note): WPIPE mul
tiplyMatrixVector_out_buffer-bufPipe requester=0 data= 0
/home/morack/ahir_release/release/vhdl/ahir.vhdl:17345:40:@485845ns:(assertion note): RPIPE mul
tiplyMatrixVector_out_buffer-bufPipe requester=0 data= 0
```

Figure 4: Simulation Result



```
morack@morack-Lenovo-IdeaPad-S145-15IWL:~/ahir_release/mvp_unit$ ./bin/testbench_aa2c null
Info:pThreadUtils: started thread mvp_daemon
Info:pThreadUtils: started thread Receiver
Info:pThreadUtils: started thread Sender
Info: Sender: sent vector number 0
Info: Receiver: received vector number 0
Info: Sender: sent vector number 1
Info: Receiver: received vector number 1
Info: Sender: sent vector number 2
Info: Receiver: received vector number 2
Info: Sender: sent vector number 3
Info: Receiver: received vector number 3
Info: Sender: sent vector number 4
Info: Receiver: received vector number 4
Info: Sender: sent vector number 5
Info: Receiver: received vector number 5
Info: Sender: sent vector number 6
Info: Receiver: received vector number 6
Info: Sender: sent vector number 7
Info: Receiver: received vector number 7
Info: Sender: sent vector number 8
Info: Receiver: received vector number 8
Info: Sender: sent vector number 9
Info: Receiver: received vector number 9
Info: Sender: sent vector number 10
Info: Receiver: received vector number 10
Info: Sender: sent vector number 11
Info: Receiver: received vector number 11
Info: Sender: sent vector number 12
Info: Receiver: received vector number 12
Info: Sender: sent vector number 13
Info: Receiver: received vector number 13
Info: Sender: sent vector number 14
Info: Receiver: received vector number 14
Info: Sender: sent vector number 15
Info: Receiver: received vector number 15
Info: Sender: sent vector number 16
Info: Receiver: received vector number 16
Info: Sender: sent vector number 17
Info: Receiver: received vector number 17
Info: Sender: sent vector number 18
Info: Receiver: received vector number 18
Info: Sender: sent vector number 19
Info: Receiver: received vector number 19
Info: Sender: sent vector number 20
Info: Receiver: received vector number 20
Info: Sender: sent vector number 21
Info: Receiver: received vector number 21
Info: Sender: sent vector number 22
Info: Receiver: received vector number 22
Info: Sender: sent vector number 23
Info: Receiver: received vector number 23
Info: Sender: sent vector number 24
Info: Receiver: received vector number 24
Info: Sender: sent vector number 25
Info: Receiver: received vector number 25
Info: Sender: sent vector number 26
Info: Receiver: received vector number 26
Info: Sender: sent vector number 27
Info: Receiver: received vector number 27
Info: Sender: sent vector number 28
Info: Receiver: received vector number 28
Info: Sender: sent vector number 29
Info: Receiver: received vector number 29
Info: Sender: sent vector number 30
Info: Receiver: received vector number 30
Info: Sender: sent vector number 31
Info: Receiver: received vector number 31

SUCCESS!
```

Figure 5: C Testbench Verification

9 Conclusion

The design was complied successfully and a performance boost of 6.4 was achieved with the design. The simulation was done using the GHDL simulator and the waveform obtained was studied using GTKWave Viewer to get the performance metrics. The utilisation of the system is low though. To improve the utilisation adders and multipliers can be shared but this will reduce the performance. Another method is to use lesser parallel units for computation but this also effects the performance. There is a trade-off between the performance and utilisation.