ESE 507 | PROJECT 2

MATRIX-VECTOR MULTIPLICATION

Aswin Natesh Venkatesh & SBU ID: 111582677
Gosakan Srinivasan SBU ID: 111579886

PART 1

QUESTION 4A:

Mathematically, multiplying a 3x3 Matrix and a Vector of size 3 involves 9 multiplication and 6 arithmetic operations which is 15 arithmetic operations in total. This can be generalized for a K x K matrix, which requires K^2 multiplication operations and 2K addition operations. Therefore, total arithmetic operations involved would be [$K^2 + 2K$].

QUESTION 4B:

The control module uses a Finite State Machine (FSM) consisting of 4 discrete states. Each state is assigned a specific set of instructions to be carried out. The FSM keeps track of this and transit states upon successful completion of operations by individual states. The 4 states are listed below in the table.

State No	*Tasks Performed*	*Transition Signal*
State 0	Idle State I Wait for Input Valid Signal & Data.	S_Valid == 1
State 1	Data Input I Handshake and receive Input Data Values [M & X]	Write_complete == 1
State 2	Matrix Vector Multiplication I Feed Data Input to MAC Module, obtain computed values and store them to memory.	Mac_complete == 1
State 3	Display State I Handshake and send results out.	Data_displayed == 1

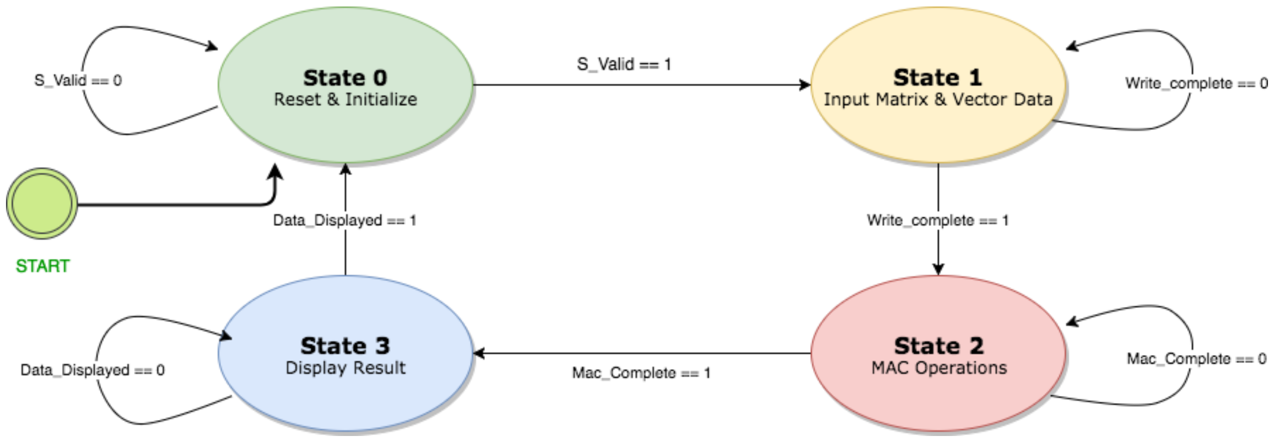

Figure 1: FSM State Transition Diagram

The control modules involve various counters to assert flag signals and write signals. This module also generates the addresses for read/write operations from memory and effectively controls and directs data in/out from in the data path module. The individual state operations are detailed below.

Detailed STATE operations at every positive clock edge: -

STATE 0: Wait for S_Valid signal, Initialize variables and Initiate State 1.

STATE 1: Generate Address values for Input Matrix M, Vector X and corresponding Write Enable signals. Write_complete flag is asserted once data is received completely.

STATE 2: Feed Row wise and Column Wise matrix and vector values into MAC Module, and stores results in Memory Y. Mac_complete flag is asserted once MAC operations are complete. (Address and Write signals are asserted accordingly)

STATE 3: Wait for M_ready signal and display results. Data_displayed flag is asserted once results are displayed and initiate State 0. This completes of one iteration for the given MVM Inputs.

QUESTION 4C:

A C-Program Based Random Value Test Bench was designed to verify the Matrix Vector Multiplication Unit (MVM). This verification strategy consists of two parts.

Part I. This part consists of C-Program which generates random the test data and expected results. The C-Program randomly generates input values for a K x K Matrix and a Vector sized K for a user specified number of iterations. The default number of iterations is set as 5. The randomly generated input values are then stored in a file named "**C_Input**". The input values generated falls within the ranges mentioned below.

Matrix M	: (+127) ~ (-127) - (8 Bit Signed)	
Vector X	: (+127) ~ (-127) - (8 Bit Signed)	
Vector B	: (+127) ~ (-127) - (8 Bit Signed)	(For Part 2 – 4)

The C-Program also performs Matrix Vector Multiplication with overflow detection and writes the results to a file named **"C_Output".**

Part II. This Part is the System Verilog testbench, where the MVM under test is fed with the input values generated in the previous part. The testbench includes a randomized testing methodology for verifying the handshaking protocol. The handshaking variables **"S_Valid"** and **"M_Ready"** signals are asserted based on randomization which makes the testbench more robust. The randomization pattern can be changed by changing the random seeds (42 to any number) in the code snippet below.

```
<vsim –sv seed 42 –c check_timing>
run <desired time>ns                                       // say run 2000ns
```

The result from the MVM unit is obtained and written to a file named **"Sv_Output"**. This file is then compared with the **"C_Output"** generated in previous part for mismatches, if any. The system works correctly when no differences are found between these two files.

The testbench thus effectively tests the system for a large set of random inputs, also tests overflow and proper functionality of handshaking protocol.

QUESTION 4D: PART 01 – SYNTHESIS REPORTS

Target Frequency (GHz)	*1.000*	*0.909*	*0.833*	*0.625*	*0.500*
Clock Period (ns)	1.00	1.10	1.20	1.60	2.00
Combinational Area (μm^2)	1076.50	1059.21	1035.27	1012.66	889.50
Non Combinational Area (μm^2)	994.57	992.45	990.85	990.85	990.32
Total Cell Area (μm^2)	2071.08	2051.66	2026.12	2003.51	1879.82
Total Dynamic Power (mW)	1.43	1.28	1.17	0.88	0.64
Cell Leakage Power (uW)	42.10	41.28	40.71	39.39	35.42
Total Power (mW)	1.47	1.32	1.21	0.92	0.67
Energy/Cycle Operation (pJ)	1.47	1.45	1.45	1.47	1.34
Timing Report (Slack)	-0.21	-0.09	0.00	0.00	0.00
	VIOLATED	VIOLATED	MET	MET	MET
Critical Part - Start Point	data_out_reg [1]	data_out_reg [2]	data_out_reg [3]	data_out_reg [1]	data_out_reg [3]
Critical Part - End Point	overflow_reg	overflow_reg	overflow_reg	overflow_reg	overflow_reg
Output File Name	---	---	Output01.txt	----	----

QUESTION 4E:

Number of Clock Cycles for the system takes to load one set of inputs, compute one matrix-vector multiplication of size k=3, and output the result. Assuming s_valid & m_ready are always asserted	⇨ **37 Clock Cycles** at 833 MHz
Delay of System ⇨ Clock Period X Number of Clock Cycles ⇨ 1.20ns X 37	⇨ 44.40 ns

QUESTION 4F:

Average Delay Product ⇨ Delay X Total Cell Area ⇨ 44.40ns X 2026.12μm^2 X 10^{-3}	⇨ 89.96 $m^2 ns$

QUESTION 4G:

Energy Consumed per Matrix Vector Multiplication ⇨ Clock Period X Total Power X #Clock Cycles ⇨ 1.20ns X 1.21mW X 37	⇨ 53.71 pJ

QUESTION 4G:

Arithmetic Operation Count	⇨ 18
Energy Consumed per Arithmetic Count ⇨ Energy Consumed / Operation Count ⇨ 53.71pJ / 18	⇨ 2.98 pJ

PART 2

QUESTION A:

Mathematically, multiplying a 3x3 Matrix and a Vector of size 3 involves 9 multiplication and 6 arithmetic operations which is 15 arithmetic operations in total. Adding an additional vector to the result involves 3 more arithmetic operations. This can be generalized for a K x K matrix, which requires K^2 multiplication operations and K^2 addition operations. Therefore, total arithmetic operations involved would be [$2K^2$].

QUESTION B:

For this operation, the control module was modified in such a way that values of Vector B are initialised to the accumulation register at the beginning of each (Row X Column) MAC Operations. This is instead of resetting the accumulation register to 0, we will reset it to the appropriate value from the B memory. This approach keeps the arithmetic operations per MVM unchanged, and therefore no additional arithmetic cost is involved compare to previous part.

QUESTION C: PART 02 – SYNTHESIS REPORTS

Target Frequency (GHz)	0.833
Clock Period (ns)	1.20
Combinational Area (μm^2)	1087.67
Non Combinational Area (μm^2)	1174.12
Total Cell Area (μm^2)	2261.80
Total Dynamic Power (mW)	1.28
Cell Leakage Power (uW)	44.44
Total Power (mW)	1.33
Energy/Cycle Operation (pJ)	1.59
Timing Report (Slack)	0.00
	MET
Critical Part - Start Point	data_out_reg [5]
Critical Part - End Point	overflow_reg
Output File Name	Output02.txt

QUESTION D:

Number of Clock Cycles for the system takes to load one set of inputs, compute one matrix-vector multiplication of size k=3, and output the result. Assuming s_valid & m_ready as always asserted	⇨ **69 Clock Cycles** at 833 MHz
Delay of System ⇨ Clock Period X Number of Clock Cycles ⇨ 1.20ns X 69	⇨ 82.80 ns

QUESTION E:

Average Delay Product ⇨ Delay X Total Cell Area ⇨ 2261.80μm^2 X 82.80ns x 10^{-3}	⇨ 187.28 $m^2 ns$

QUESTION F:

Energy Consumed per Matrix Vector Multiplication ⇨ Clock Period X Total Power X #Clock Cycles ⇨ 1.20ns X 1.33mW X 69	⇨ 109.91 pJ

QUESTION G:

Arithmetic Operation Count	⇨ 18
Energy Consumed per Arithmetic Count ⇨ Energy Consumed / Operation Count ⇨ 109.91pJ / 18	⇨ 6.11 pJ

PART 3

QUESTION A:

Changing Design from 3 x 3 to 4 x 4

For 4 x 4 Matrix vector multiplication system the changes involved are simple. Changing the Counter limits, address line widths and memory sizes would suffice. The FSM in Control Module remains unchanged, and therefore the system is designed in a generalised manner which reduces the overall complexity when scaling the matrix sizes. The only practical limit for maximum value of K would be dependent on the memory availability and Hardware constraints.

QUESTION B: PART 02 – SYNTHESIS REPORTS

Target Frequency (GHz)	0.625
Clock Period (ns)	1.60
Combinational Area (μm^2)	1581.90
Non Combinational Area (μm^2)	1611.43
Total Cell Area (μm^2)	3194.93
Total Dynamic Power (mW)	1.34
Cell Leakage Power (uW)	66.15
Total Power (mW)	1.41
Energy/Cycle Operation (pJ)	2.25
Timing Report (Slack)	0.00
	MET
Critical Part - Start Point	data_out_reg [2]
Critical Part - End Point	overflow_reg
Output File Name	Output03.txt

QUESTION C:

Number of Clock Cycles for the system takes to load one set of inputs, compute one matrix-vector multiplication of size k=3, and output the result. Assuming s_valid & m_ready as always asserted	⇨ **65 Clock Cycles** at 625 MHz
Delay of System ⇨ Clock Period X Number of Clock Cycles ⇨ 1.60ns X 65	⇨ 104.00 ns

QUESTION D:

Average Delay Product ⇨ Delay X Total Cell Area ⇨ 3194.93μm^2 X 104.00ns x 10^{-3}	⇨ 332.27 m^2ns

QUESTION E:

Energy Consumed per Matrix Vector Multiplication ⇨ Clock Period X Total Power X #Clock Cycles ⇨ 1.60ns X 1.41mW X 65	⇨ 146.20 pJ

QUESTION F:

Arithmetic Operation Count	⇨ 32
Energy Consumed per Arithmetic Count ⇨ Energy Consumed / Operation Count ⇨ 146.20pJ / 32	⇨ 4.57 pJ

PART 4

QUESTION A:

Input-Output Overlapping
In order to boost the speed of the MVM System, the control module was modified to perform input output overlapping. That is while the MVM system is outputting data on **data_out**, it can begin taking in new data on **data_in**. This effectively saves clock cycles and brings in a significant improvement in maximum frequency of operation that is 36.8. (ie. 625 MHz to 855 Mhz)

QUESTION B: POST SYNTHESIS DATA COMPARISON

Part of Project		Part 01	Part 02	Part 03	Part 04
Question	Target Frequency (GHz)	0.833	0.833	0.625	0.855
D	Clock Period (ns)	1.20	1.20	1.60	1.17
	Combinational Area (um^2)	1035.27	1087.67	1581.90	1588.29
	Non Combinational Area (um^2)	990.85	1174.12	1611.43	1639.62
	Total Cell Area (um^2)	2026.12	2261.80	3194.93	3227.91
	Total Dynamic Power (mW)	1.17	1.28	1.34	1.79
	Cell Lekage Power (uW)	40.71	44.44	66.15	67.62
	Total Power (mW)	1.21	1.33	1.41	1.86
	Energy per Cycle Operation (pJ)	1.45	1.59	2.25	2.17
	Timing Report (Slack)	0.00	0.00	0.00	0.00
		MET	MET	MET	MET
	Critical Part - Start Point	data_out_reg [3]	data_out_reg [5]	data_out_reg [2]	data_out_reg [3]
	Critical Part - End Point	overflow_reg	overflow_reg	overflow_reg	overflow_reg
	Output File Name	Output01.txt	Output02.txt	Output03.txt	Output04.txt
E	Number of Clock Cycles / MVM	37.00	69.00	65.00	56.00
	Delay (ns)	44.40	82.80	104.00	65.52
F	Area Delay Product (m^{2}ns)	89.96	187.28	332.27	211.49
G	Energy Consumed / MVM (pJ)	53.71	109.91	146.20	121.65
H	Operation Count of System	18.00	18.00	32.00	32.00
	Energy Consumed / Arithmetic Operation (pJ)	2.98	6.11	4.57	3.80

QUESTION C:

The final optimized design clocks 230 MHz higher, occupies 32.98μm^2 more area and consumes slightly more power than the design in Part 3. Therefore, the area delay product for this new design is comparatively lower than the design in part 3. This is because the result is delivered in fewer clock cycles for the same number of inputs. As the total energy consumed by this design is comparatively less, the net energy consumed per arithmetic operation is also reduced.

QUESTION D:

Theoretically speaking Energy-per-operation can be optimized by **increasing the number of operations, decreasing the frequency, decreasing the number of clock cycles.**

For our system, increasing the number of operations consumes more power while decreasing the frequency increases the power consumed. Therefore, decreasing the number of clock cycles with a perfect trade-off between frequency and number of operations would be the best approach to minimize energy.

QUESTION E:

Since the number of input/output ports is constrained, the only way to make the design operate faster would be to read and display multiple values at the same time, which is adding additional ports to the system. This will allow us to reduce the delay of the system and more efficiently make use of clocks.

QUESTION A – TRIALS:

Design-Ware Components
In order to further boost the speed of MVM, a 2 Stage DesignWare Adder was implemented. Due to time constrains, we weren't able to complete our design due to timing issues. Though we did manage to get partial successful outputs.

END OF REPORT