ESE 507

Project 3

By

Charan Lellaboyena 111218815

Abhishek S Yellanki 111095603

1.

1. Firstly, the addresses used to access W-Rom or B-Rom depend on the size of the input M*N vector.
    a. The size of W-Rom is given by the no. of bits used to represent the product of no. of rows and no. of columns of the input M*N vector.
    b. The size of B-Rom is given by the no. of bits used to represent the no. of rows in the M vector that is loaded into the B rom. This has to be M to make the vector multiplication and addition possible.
    c. The size of memory X is given by the number of bits used to represent N, where N is same as the no. of columns of the W-vector. Here if X has to have size N to make MVM operation valid.
2. Secondly, the number of bits used to represent the value stored in the ROMs W and B and RAM X is given by the parameter ‘bits’ which is directly parsed to the respective memories.
3. Then,
    a. In our design the valid levels of parallelism is the set consisting of factors of M, where M represents the number of rows of W vector.
    b. The top module instantiates P data-paths, where P is the levels of parallelism. Each data-path computes the output in parallel.
    c. We used a multiplexer at the output stage and used a counter whose maximum is set to P, to count the number of outputs mapped to the actual output of the system.
    d. In the output state, we check if all outputs from the parallel levels are output. This was done keeping in mind the m_ready signal.
4.
    a. We did not define any limits on M and N. But during simulation we found that there were errors while computing the output for M=5, N=2, P=5, bits=9. There was no failure for any other set of parameters.
    b. We checked for a design with M=512, N=512. The simulation was successful.

2.

1. Since the input vectors W and B are ROMs the only input is the vector X which has to be written into the RAM memory. The combinational part of the FSM decides on what has to be done in a particular state. The sequential part, along with stating what has to be done, it decides when to change between states.
2. There are 5 states in our FSM. RESET state sets all signals and addresses to 0.
3. INITIAL state sets all signals to 0 and addresses to their initial values depending on the levels of parallelism. At each level of parallelism, address of vector W is incremented by N so that each parallel MAC points to next row.

4. WRITE_X state is used to write into the X memory.
5. MAC state passes address values to all the data-paths. It keeps a count of the values in X memory so that the current output can be outputted. Once the limit of X is reached, all addresses pointing to W vector are incremented [N*(P – 1) + 1] if P != M. If P = M all addresses are incremented by only 1. It doesn't really matter because, all the outputs are computed at a time and there is no need to increment the addresses.
6. In the OUTPUT state, the system waits for m_ready then asserts m_valid and sends output. That is done until results from all the parallel levels is outputted.
7. As the values of M and N grow the size of each memory location of all the memory units, changes. As the value of M increases the scope of parallelizing the system also increases. In our design the levels of parallelism can be any factor of M, where M is the number of rows in W vector.
8. ReLU is implemented using a signal en_check which is passed to the data-path. When this signal is high and the respective data-path output negative, the result is corrected to 0.

3.

There are 5 components that change if P > 1. They are Address of W, Address of B, Address of X memories, counter value that triggers the multiplexer to send results of each MAC to the output of the system and number of data-path instantiations.

1. P address values are initialized for W and B vectors. Address values for W vector are incremented by a value of N depending on the levels of parallelism (P). That is because each data-path reads a new row of input (each consisting of N columns).
2. Address values of B vector are incremented by value P. That is because during first computation P values of B vector are added to P results.
3. Address X remains same since that vector is repeatedly multiplied with the W vector.
4. The counter value depends on P. If there are P levels of parallelism the system has to wait until all computed results are outputted before accepting new set of inputs. The counter keeps track of this.
5. There will be P data-path instantiations and they all run in parallel to compute results. These results are multiplexed and mapped to the output.
6. **Optimization** – all the address values are limited by the size of W, X, B vectors so that memories are not larger than needed. Instead of using buffers to save results of the parallel MACs we have used a multiplexer where the output of each MAC waits until its turn comes only to be mapped to the output of the system. Data is read from the same memories by all parallel levels which saves a lot of memory resources.

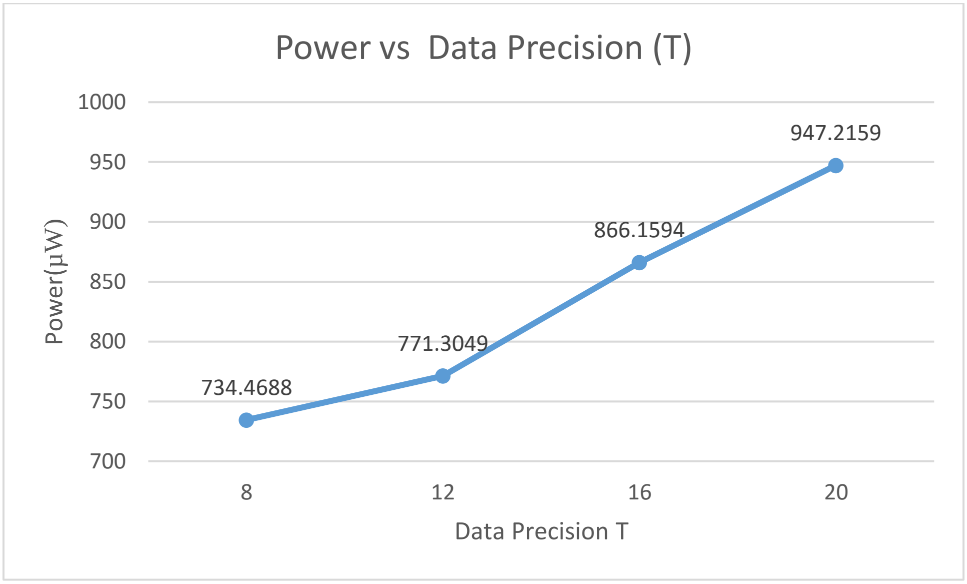
4.

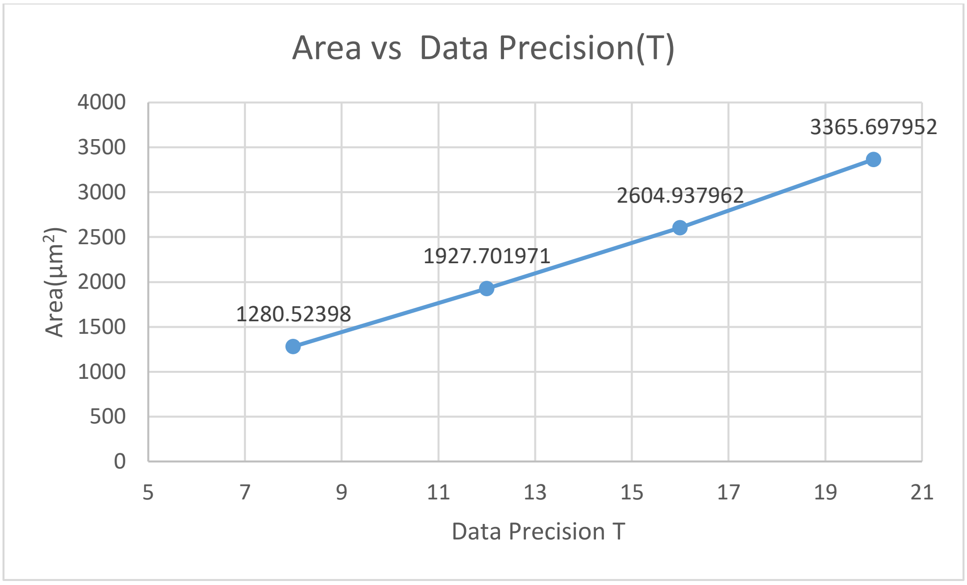
If the problem size increases that will imply that sizes of memories have to be increased which will increase cost. It will result in decrease in performance as the memory accesses increase, time for each computation increases. Precision increases. 1 32-computation on a 16-bit system is not at all precise compared to the same computation on 32-bit system.

1. If the values of M or N or T are increased keeping P=1, it implies that the problem size has been increased. And there will be increase in cost, decrease in performance and increase in precision.
2. If there are multiple levels of parallelism (P>1) that is if the value of P increases, the cost increases since there will be multiple MACs used. The performance of the system increases. Greater the value higher will be the cost and better the performance.
3. There is a tradeoff between cost and performance that can be found if P increases.

5. For (M, N, T) = (8,8,1) the changes in the area and power with respect to the changes in data precision T are given the table below.

T	**Power (μW)**	**Area(μm^2)**	**Critical Path**
8	734.4688	1280.524	Startpoint: data1/rom_w/z_reg[0] Endpoint: data1/M/f_reg[2]
12	771.3049	1927.702	Startpoint: data1/rom_w/z_reg[2] Endpoint: data1/M/f_reg[1]
16	866.1594	2604.938	Startpoint: data1/mem_x/data_out_reg[3] Endpoint: data1/M/f_reg[0]
20	947.2159	3365.698	Startpoint: data1/rom_w/z_reg[1] Endpoint: data1/M/f_reg[0]

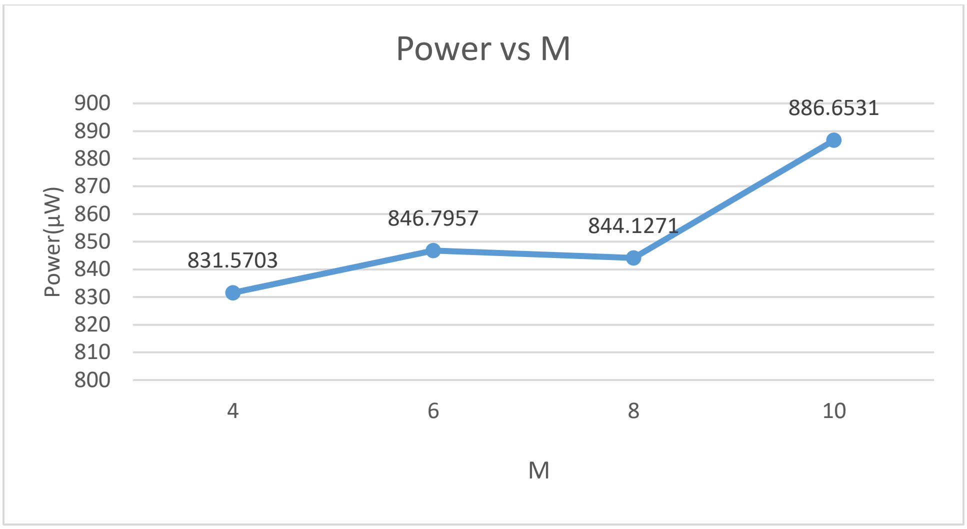


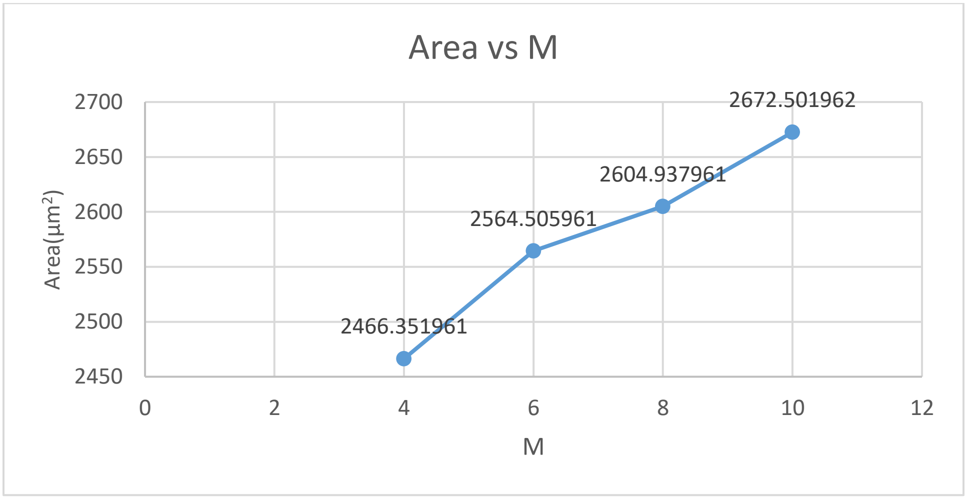


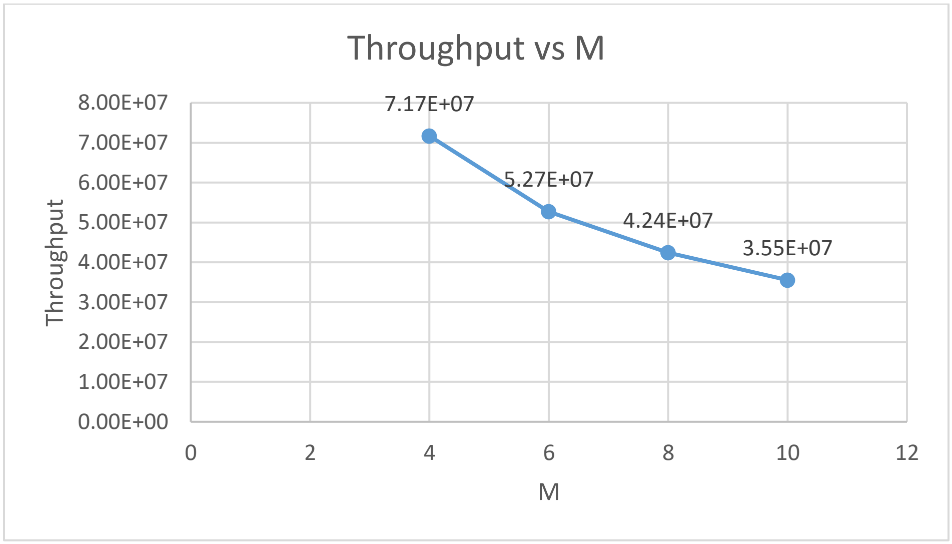
6. Throughput, power, and area are evaluated for M = 4,6,8 and 10 with N = 8, P = 1 and T = 16. Respective values are given in the table.

M	**Power(µW)**	**Area(µm²)**	**C (# cycles to process N input words)**	**Throughput**	**Clock Period(ns)**
4	831.5703	2466.352	62	7.17E+07	1.84
6	846.7957	2564.506	83	5.27E+07	1.83
8	844.1271	2604.938	103	4.24E+07	1.83
10	886.6531	2672.502	123	3.55E+07	1.83

For all the designs generated the critical path is between W_ROM register to accumulator register f_reg. Hence the location of critical path doesn't change as M changes. c, the number of cycles required to process N inputs is calculated at the input by checking the number of cycles the design took between starting points of two consecutive vector reads i.e., period between two consecutive vector reads is calculated and divided with clock cycle period (in GUI 10 ns) by verifying the waveforms.

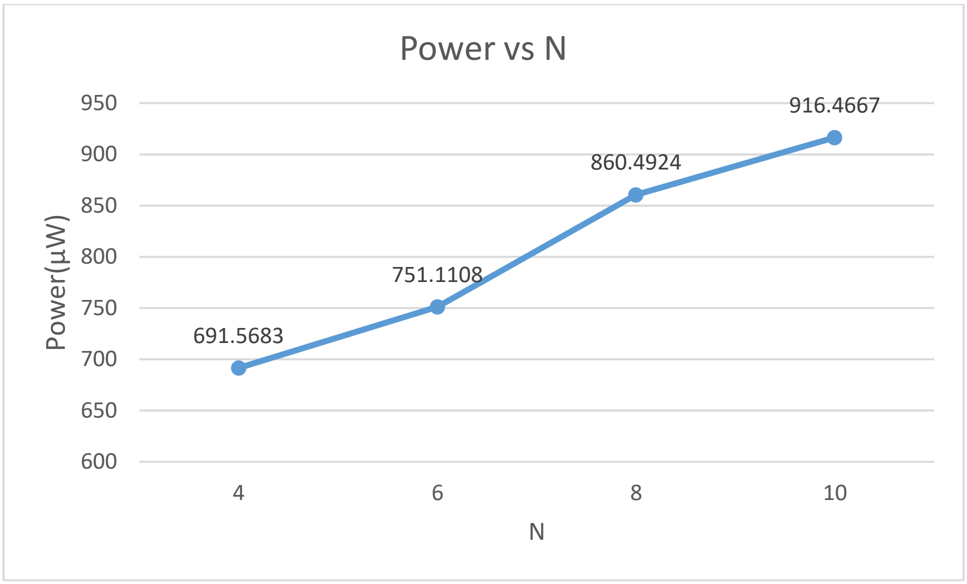

Power vs M
Power(μW)
900
890
880
870
860
850
840
830
820
810
800
831.5703
846.7957
844.1271
886.6531
4
6
8
10
M

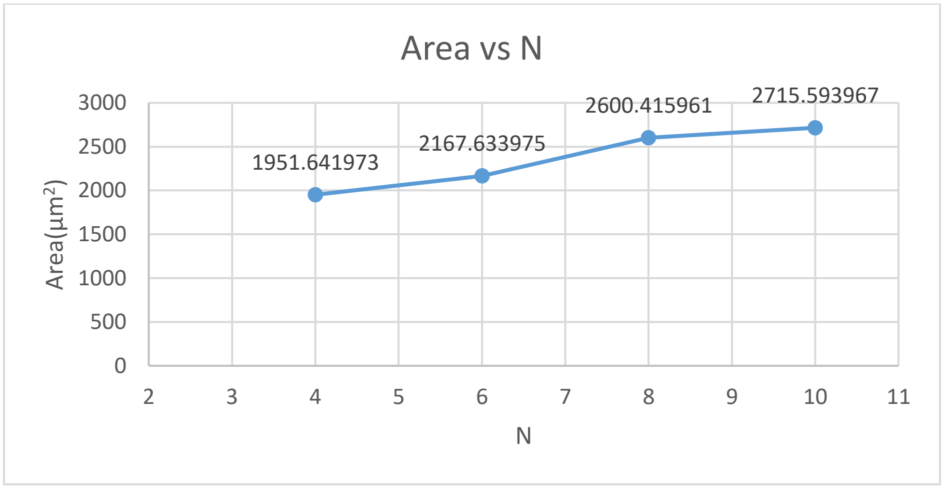

Area vs M
Area(μm2)
2700
2650
2600
2550
2500
2450
2466.351961
2564.505961
2604.937961
2672.501962
0
2
4
6
8
10
12
M

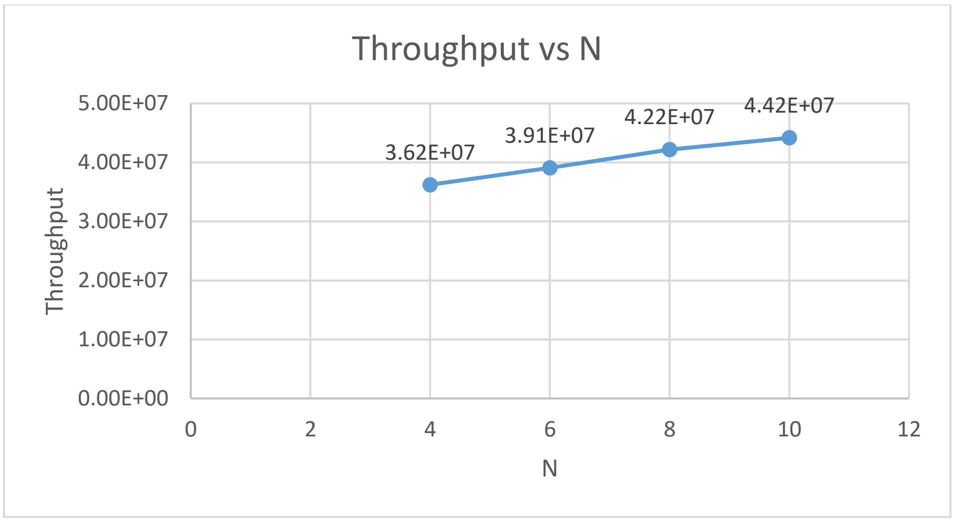

Throughput vs M
Throughput
8.00E+07
7.00E+07
6.00E+07
5.00E+07
4.00E+07
3.00E+07
2.00E+07
1.00E+07
0.00E+00
7.17E+07
5.27E+07
4.24E+07
3.55E+07
0
2
4
6
8
10
12
M

7. Throughput, power, and area are evaluated for N = 4,6,8 and 10 with M = 8, P = 1 and T = 16. Respective values are given in the table.

N	**Power(µW)**	**Area(µm^2)**	**C (# cycles to process N input words)**	**Throughput**	**Clock Period(ns)**
4	691.5683	1951.642	60	3.62E+07	1.84
6	751.1108	2167.634	83	3.91E+07	1.85
8	860.4924	2600.416	103	4.22E+07	1.84
10	916.4667	2715.594	123	4.42E+07	1.84



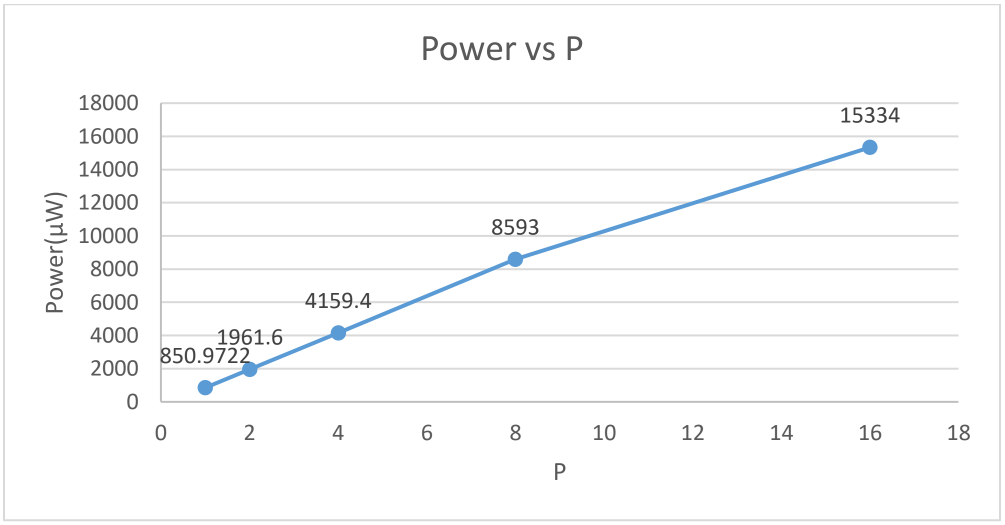


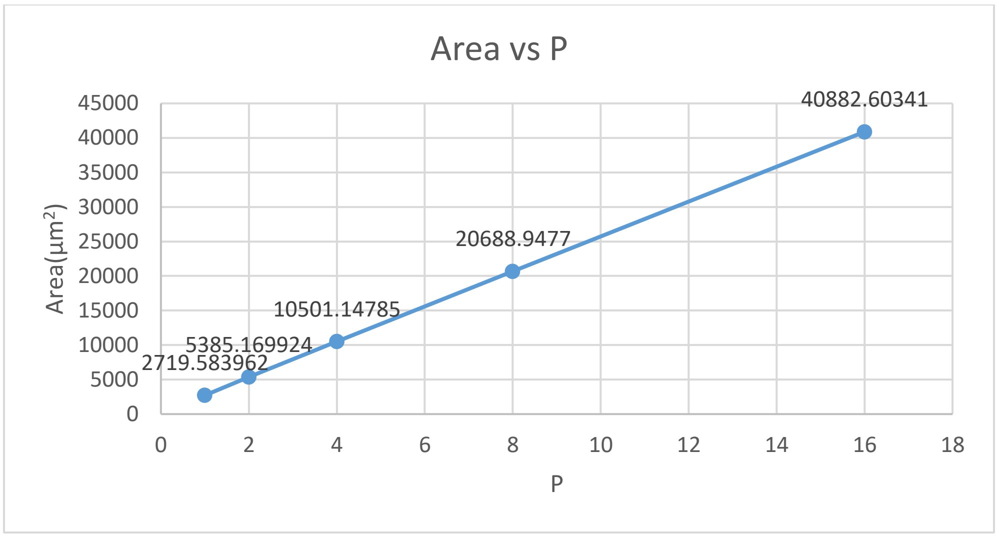


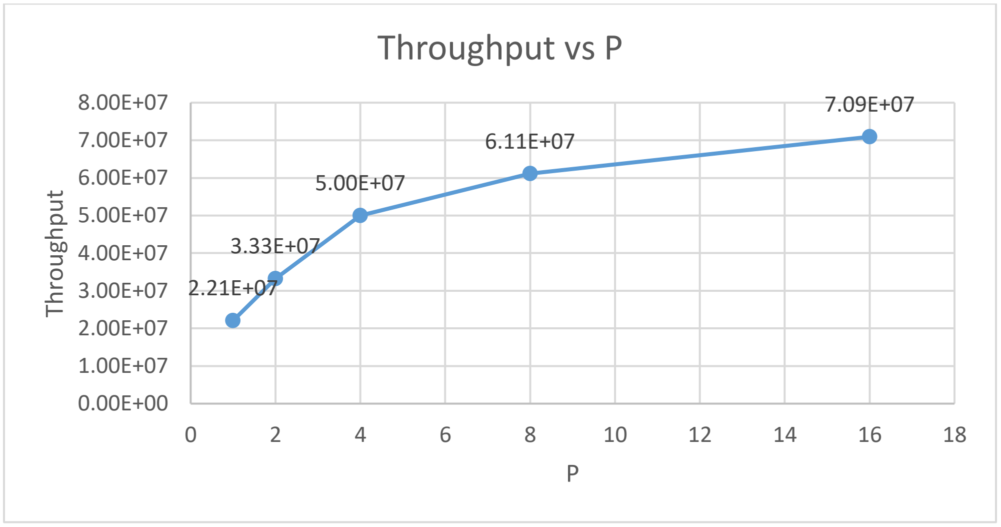
Out of the four designs three times the critical path is between W_ROM register to accumulator register f_reg. But once it's from SRAM for vector to accumulator register

8. Throughput, power, and area are evaluated for P = 1,2,4 and 8 with M = 16, N= 8 and T = 16. Respective values are given in the table.

P	**Power(µW)**	**Area(µm^2)**	**C (# cycles to process N input words)**	**Throughput**	**Clock Period(ns)**
1	850.9722	2719.584	193	2.21E+07	1.84
2	1961.6	5385.17	130	3.33E+07	1.85
4	4159.4	10501.15	86	5.00E+07	1.86
8	8593	20688.95	70	6.11E+07	1.87
16	15334	40882.6	60	7.09E+07	1.88







From the table and graphs above, as parallelism increases both power and area are increasing linearly but throughput is not increasing linearly. The factor of increase in throughput is decreasing as the parallelism P increases. For example, for P = 4 and P = 8 both power and area got almost doubled but not the throughput. Hence, as parallelism increases the cost is increasing significantly but the throughput is not increasing as much as cost. Considering this the most efficient design among these designs is parallelism P = 4. Because from design P = 2 to P = 4 the increase in throughput is as significant as the increase in both power and cost.

9. In part 2 we are applying parallelism for each MAC operation. Since we need N MAC operations we can only have at most N multipliers per layer. To parallelize further we can parallelize the multiplier in each MAC operation. All the multiplications required for each MAC operation are done parallelly and after doing all the multiplications the outputs are added together. This increases the parallelism even further. The drawback in this approach is the adder circuitry will become cumbersome compared to the previous simple adder. Also, since the number of multipliers are increasing significantly both area and power for the design increases very significantly.

10.

This is the approach we implemented:

1. Firstly we will assign 1 MAC unit to each of the layers which will set P1, P2 and P3 to 1.
2. The input values for each layer are identified and factorized. The factorized values are stored in respective arrays in increasing order. The array contains the number (input size M3) also.
3. Starting from layer3 we compared values of 'L - 3' with each of the factors from highest to lowest.
4. If (L-3) is greater than any one of the factors (say M3_3), that 'factor – 1' is added to P3 and the loop exits. The last step before exiting the loop is decreasing (L-3) with (M3_3 – 1).
5. The above 2 steps are repeated for layer2 and layer1 as well.

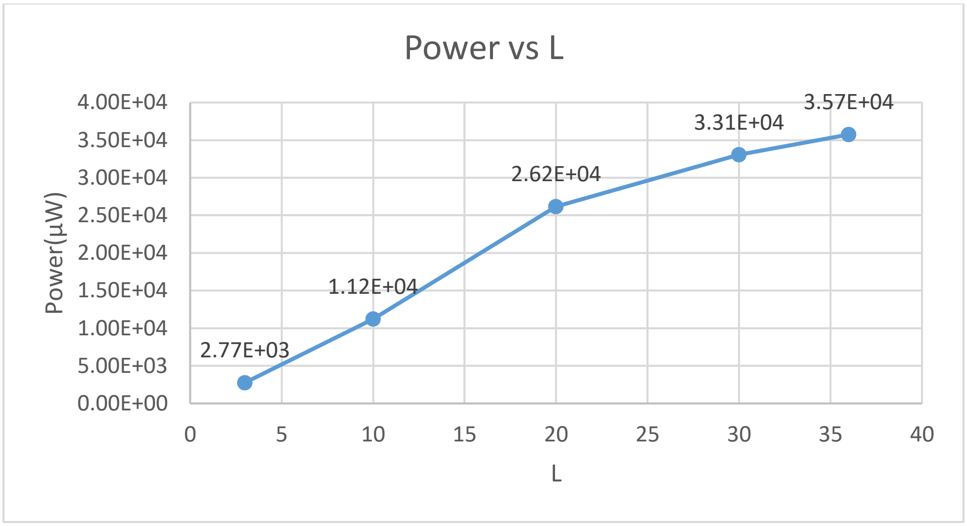
This is the approach we wanted to implement:

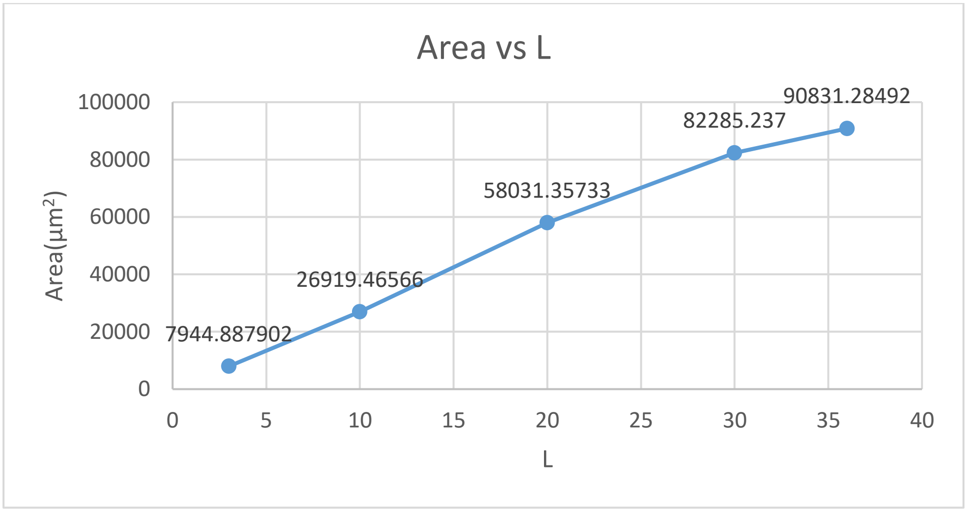
1. Firstly we will assign 1 MAC unit to each of the layers which will set P1, P2 and P3 to 1.
2. The input values for each layer are identified and factorized. The factorized values are stored in respective arrays in increasing order. The array contains the number (input size M3) also.
3. Then, the layer with most number of inputs (say M1) is considered. If 'L-3' is greater than the inputs to that layer, 'M1-1' MAC units are assigned to that layer. If L-3 is not greater than M1, 'L-3' is compared to the 2nd greatest input size (say M2). If L-3 is greater than M2, M2-1 MACs from L-3 are assigned to that layer. Otherwise the next greatest input size (M3) is checked.
4. If 'L-3' is lesser than all input sizes, then highest factor corresponding to each input value is compared to L-3 from factors of M1 to M3. If 'L-3' is lesser than all 3 'highest factors' then do the same for 2nd highest factors. If for any of the values of the factors, L-3 is greater, assign the number of MACs equal to the value of the factors, to that layer. And consider only the other two layers from now on. Remove the value from L-3.
5. Do the above step until all the factors for the input sizes >= 1 are examined.

The drawback for both the approaches is that there can be a MAC that is left out from being used. In that case we might end up wasting MAC units.

11. Power, and area are evaluated for P = 1,2,4 and 8 with M = 16, N= 8 and T = 16. Respective values are given in the table.

L	**Power(μW)**	**Area(μm^2)**	**Clock Period(ns)**
3	2.77E+03	7944.888	1.87
10	1.12E+04	26919.47	1.87
20	2.62E+04	58031.36	1.9
30	3.31E+04	82285.24	1.9
36	3.57E+04	90831.28	2





12.

1. There are 4 signals that control the operation of every layer. S_valid, m_ready, s_ready, m_valid
   To design a network of multiple layers, every pair of consecutive layers act as master – slave system.
   a. The master sends data using m_valid as a control signal which tells the slave that there is input data available. If the slave is ready it will assert an s_ready signal and receive data.
   b. Once data is processed 'slave' in the above situation acts as master to the next layer in the network and asserts an s_valid signal and waits for m_ready which act as m_valid and s_ready signals for the adjacent layer.
   c. Data_out one layers becomes data_in for the next layer in the network.
   d. In our design, we implement the same logic but only for three layers. This logic has to be implemented for M layers.

   The challenges that we might face are

   a. The data out of the network has to wait until m_valid for the last layer is asserted. Until then there won't be any valid output.

b. The output size of each layer has to be equal to the number of columns in the W vector of the next layer.