Project 3: Hardware Generation Tool

Karthik Raj	ID # 111675685
Reynerio Rubio	ID # 109899097

ESE 507: Advanced Digital System Design and Generation

Fall 2017

12/01/2017

Table of Contents

Question 1

We made our generator capable of handling flexibility for matrix sizes (*M* and *N*), parallelism (*P*), and number of bits (*T*) for a system that implements the basic function *“y=f(Wx +b)”*. Scripts were used to automatically generate layers of system. As a result, these scripts allowed us to efficiently implement and test numerous amounts of different parameters.

We designed our generator to handle various matrix sizes. This was accomplished by knowing that W is a matrix with *M* rows and *N* columns, *x* is an *N*-length column vector, and y and b are both *M*-length column vectors. With this knowledge, implementing flexibility for matrix sizes was just a matter of carefully thought-out logic.

In addition, we designed our generator to handle parallelism *(P)*. In order to output values faster, *P* outputs were calculated at once. Parallelism was the most difficult to implement, as we had to take our existing logic for matrix sizes and make that each level/rank of parallelism addressed the correct inputs for calculation.

Lastly, we also designed out generator to be handle different number of bits *(T)*. This was the easier one of the bunch to implement - where all the data in and data out of the system has the same number of bits, where $4 \leqq T \leqq 32$. Although there were no maximum defined values on *M* and *N*, it is important to keep in mind that since the maximum number of bits is *T = 32*, large values of *M* and *N* will see a lot of outputs as 0. For a 32-bit signed integer**,** values range from **−2,147,483,648 to 2,147,483,647.** If a system needs to support data that is outside of this range, another system would be more practical.

Question 2

Our control module was designed to handle changes to *M* and *N*. In our system, we used *ROMs* (read-only memory) to hold the values for our *W* matrix and *B* vector. The values for the *W* matrix and *B* vectors are loaded from an automatically generated text file.

In order to handle various values for *M,* we had to make sure that the values addressed for *W* and *B* aligned correctly. In addition, we also wrote the input data to memory and read the data as values for our *x* vector. The values for x were taken from input, written into memory, and finally read from memory. In order to handle various values for *N,* we had to make sure that the values addressed for *x* were properly being accessed.

The system is split into two stage of computation: not computing and computing. During the “not computing” stage, the system is writing values into memory for the *x* vector. Here, this stages grows linearly as *N* grows. During the “computing” stage, the system reads values from the ROMs for the *W* matrix and *B* vector and works on computing an output. This stage is constrained by both *M and N.* Here, there are *M* number of outputs, where each output has to do an *N*-amount of multiply-accumulate operations. Therefore, this stage has an *M* * *N* scaling relationship as *M* and *N* grow*s.*

Question 3

Q3: Describe how you implemented the parallel (P>1) designs. Explain your structure. What extra logic and storage elements did you need to add? Did you find any clever optimizations to reduce cost?

A3: The design uses a single ROM for the weights and bias instead of having multiple ROM's. The addresses of W (weight matrix) is traversed according to the rank of a MAC and this follows an arithmetic progression. By implementing the design using one single ROM can decrease the total cost that is incurred during the synthesis.

In our design multiple MACs (depending on the value of P) operate simultaneously on one particular set of inputs, until all the outputs corresponding to that input have been given. After this the second set of inputs are fed to the MAC's. The MAC's does not have a read cycle after the first time when the inputs have been given until they give M number of outputs. A few read cycles when computing goes to zero are saved by this kind of implementation. There were many extra logic elements added to the system from part 2. The traversal address of X and W and also the block on which every MAC works are some key logic that were added. The datapath remained the same from part 2 of the project.

Question 4

Q4: Our design parameters M, N, P, and T allow various tradeoffs between:

- problem size (e.g., how big of a neural network layer we can compute) • costs (e.g., area, power, energy)
- precision (i.e., how many bits of precision are used), and
- performance (e.g., throughput and latency).

Explain how these tradeoffs work at a conceptual level. In other words, explain how changing the four parameters listed above affects these four metrics. Be specific and think carefully.

A4: By increasing the problem size the area and the power of the system would increase very rapidly. Depending on the (M/P) ratio some layers can even harm the performance of the system by decreasing the overall speed. Having a less M/P value would mean that the layer has less arithmetic intensity. (although it is working on the same number of inputs each time the total floating point operations would be less). On the contrary having a good (M/P) value certainly adds speed to the system.

The total parallelism of one particular layer always depends on and is constrained by the number of outputs it's giving.

Throughput of the system would increase with the number of inputs N. Although the changes are less significant for lesser values of N. Throughput is inversely dependent on the number of outputs , i.e by having huge number of outputs a layer could be working on the same set of inputs until the desired output set is obtained. This reduces the throughput very much. By having a decreased throughput and less (M/P) value the latency of that particular layer increases. Having a layer work on one set of inputs for a long time to get all the outputs will certainly cause delays for the outputs to reach the next layer. The over efficiency of the systems decreases with increase in latency.

It should also be noted that when given a particular multiplier budget (such as in part 3) it is wise to invest the parallelism to the layer which is the most slowest. Even with a well pipelined code the slowest layer in the system will bottleneck the total speed of the system. Hence it is required to give more priority to the layer that involves more arithmetic processing.

Question 5

We evaluated the area and power of a layer change as data precision *T* changes. ***Table 5.0*** below shows the values obtained after synthesis for *T* = 8, 12, 16, 20 while *M*, *N*, and *P* are constant at values *M* =8, *N* = 8, and *P* = 1. Values for max clock period, total area, total power, and the critical path were obtained for each configuration.

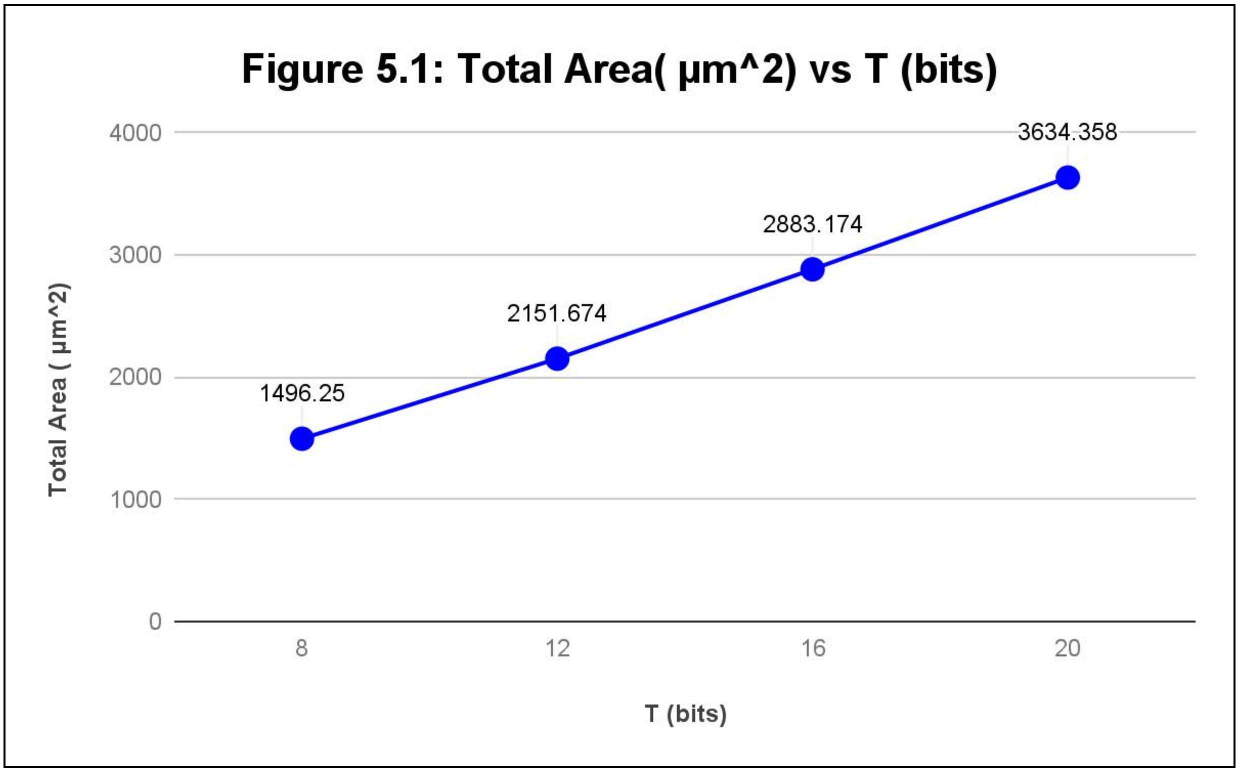
As *T* increases, we observed that the critical path for the system was initially the logic in the datapath, but became the logic in the control. We thought that the critical path would have stayed in the datapath since the system has to perform more operations on increasing number of bits.

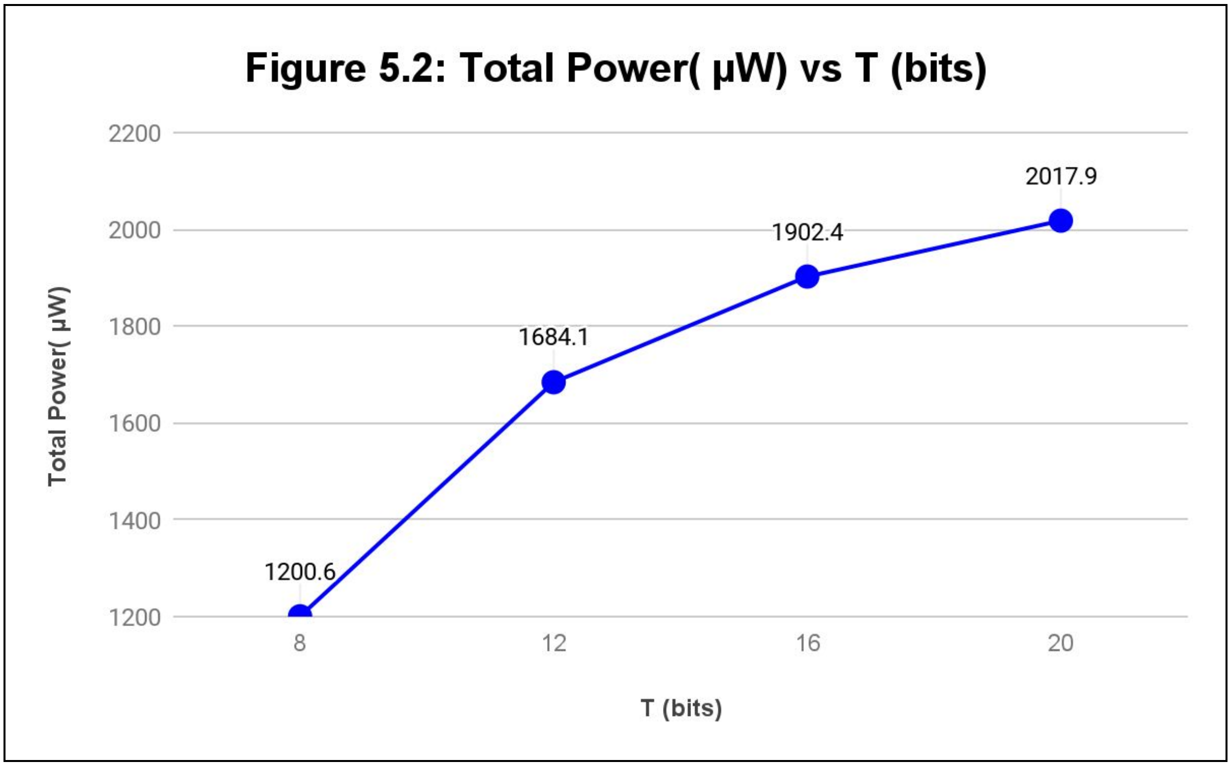
Table 5.0: Evaluating Data Precision T

M	N	P	T	Max Clock Period (ns)	Total Area (µm^2)	Total Power (µW)	Critical Path
8	8	1	8	0.87	1496.250	1200.600	mvma_8_8_1/dpth/data_out_reg[0] to mvma_8_8_1/dpth/data_out_reg[1]
8	8	1	12	0.89	2151.674	1684.100	mvma_8_8_1/dpth/rom_w/z_reg[0] to mvma_8_8_1/dpth/mult_out_reg[11]
8	8	1	16	1.04	2883.174	1902.400	mvma_8_8_1/dpth/mem_x/data_out_reg[7]to mvma_8_8_1/dpth/mult_out_reg[15]
8	8	1	20	1.23	3634.358	2017.900	mvma_8_8_1/ctrl/input_buffer_reg[1] to mvma_8_8_1/ctrl/input_buffer_reg[19]

Figures 5.1 below shows the relationship between *Total Area (µm^2) vs T (bits)*. From the graph, we can see the linear relationship between these two variables. ***Figures 5.2***

below shows the relationship between *Total Power(μW) vs T (bits*). From the graph, we can also observe the non- linear relationship between these two variables.





Question 6

We evaluated the area and power of a layer change as *M* changes. ***Table 6.0*** below shows the values obtained after synthesis for *M* = 4, 6, 8, 10 while *N*, *P*, and *T* are constant at values *N* =8, *P* = 1, and *T* = 16. Values for max clock period, total area, total power, and the critical path were obtained for each configuration.

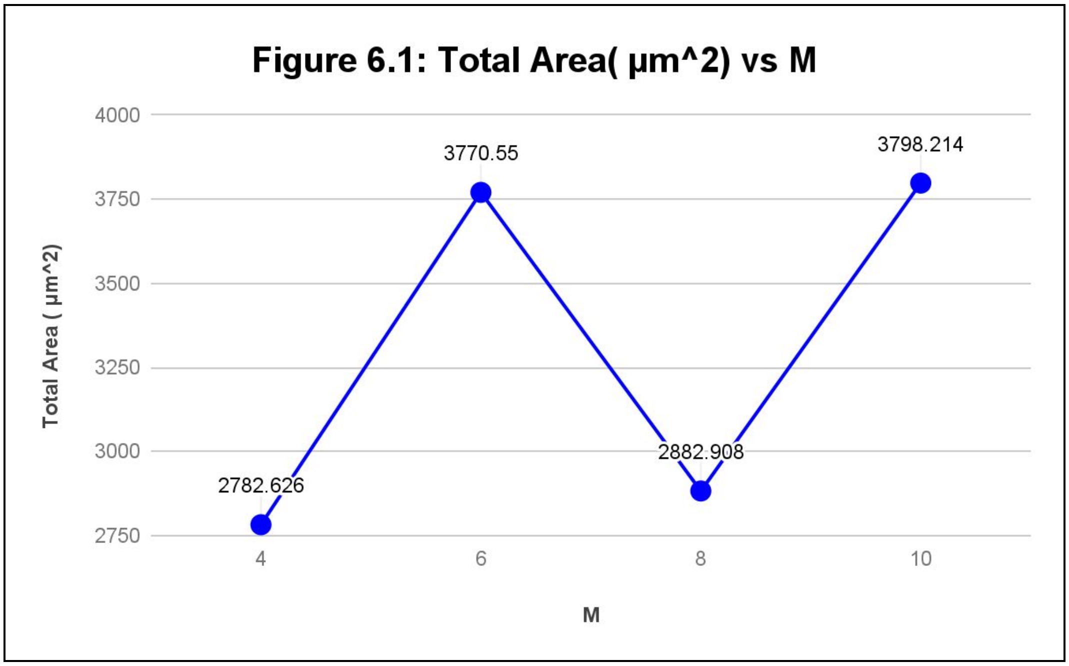
As *M* varied, we observed that the critical path for the system varied between our control logic and datapath. We thought that the critical path would have stayed in the datapath. When the critical path was calculated to be in the control logic, the system's max clock period was a lot higher than usual, which caused a slow clock rate. This clock slow clock rate caused total power to be much lower than expected.

Table 6.0: Evaluating M

M	N	P	T	Max Clock Period (ns)	Total Area (µm^2)	Total Power (µW)	Critical Path
4	8	1	16	1.02	2782.626	1917.400	mvma_4_8_1/dpth/mem_x/data_out_reg[1]to mvma_4_8_1/dpth/mult_out_reg[15]
6	8	1	16	1.57	3770.550	1296.500	mvma_6_8_1/ctrl/input_buffer_reg[11] mvma_6_8_1/ctrl/input_counter_reg[1]
8	8	1	16	1.04	2882.908	1912.100	mvma_8_8_1/dpth/rom_w/z_reg[0] to mvma_8_8_1/dpth/mult_out_reg[15]
10	8	1	16	1.53	3798.214	1362.600	mvma_10_8_1/ctrl/input_buffer_reg[9] to mvma_10_8_1/ctrl/input_counter_reg[0]]

Figures 6.1 below shows the relationship between *Total Area (µm^2) vs M*. From the graph, we can see an unexpected relationship between these two variables.

Figures 6.2 below shows the relationship between *Total Power(µW) vs M*. From the graph, we can also observe the unexpected relationship between these two variables.



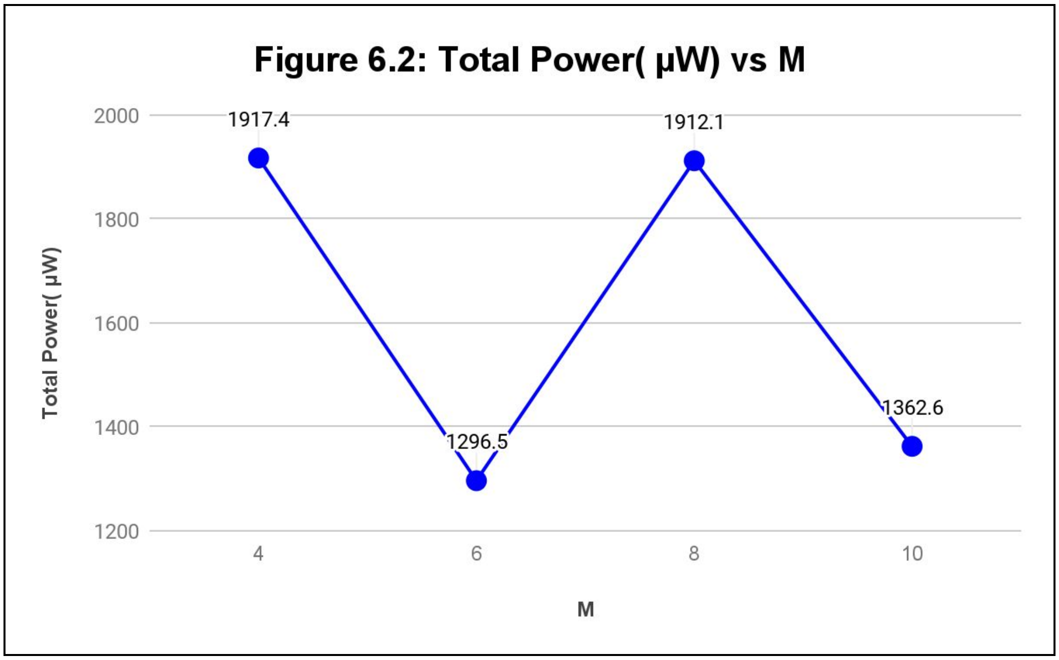
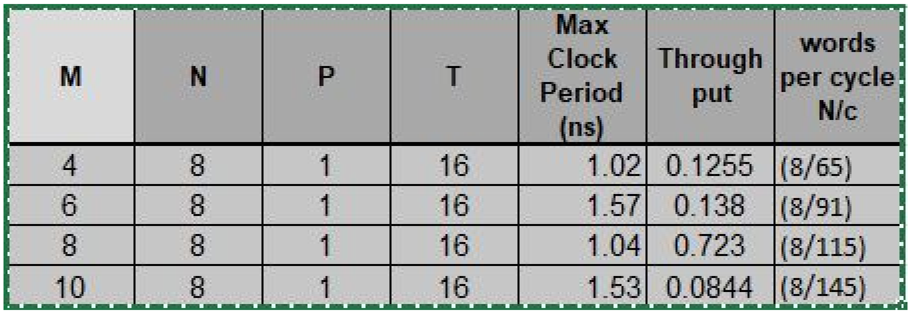


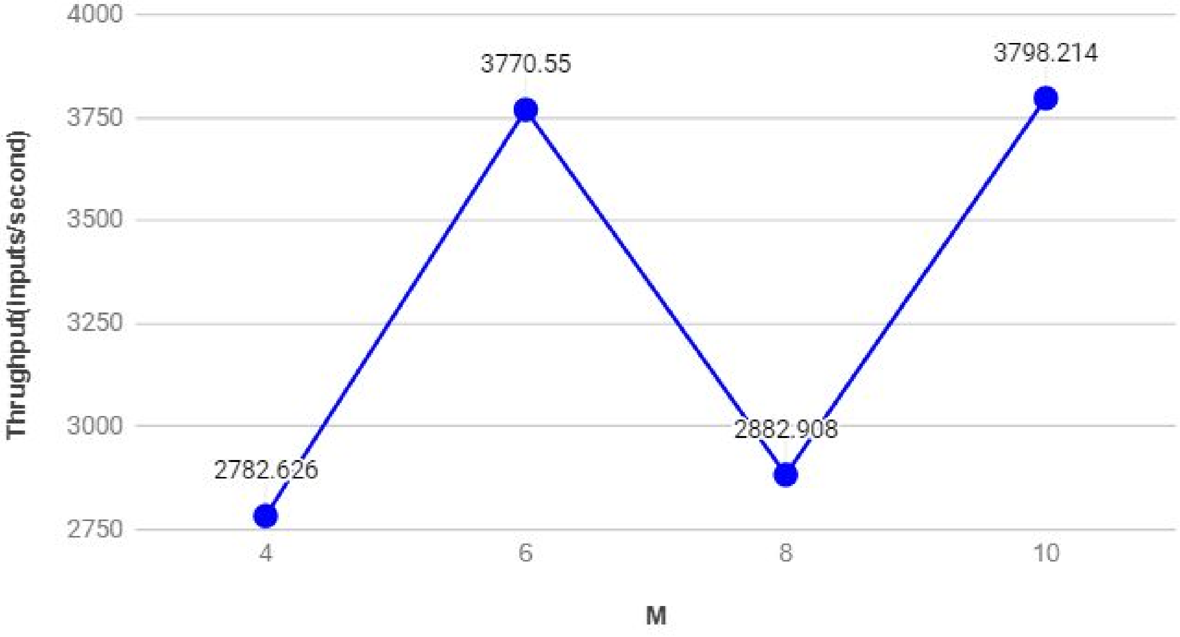
Figure 6.2: Total Power(µW) vs M

Throughput vs M : The following table represents the values of throughput varied with the number of outputs. As the number of outputs increase the total time taken to process one set of inputs increase. This decreases the throughput gradually. The results obtained follow the trend of the assumption. *Figure 6.3* below shows Throughput vs. M.



M	N	P	T	Max Clock Period (ns)	Through put	words per cycle N/c
4	8	1	16	1.02	0.1255	(8/65)
6	8	1	16	1.57	0.138	(8/91)
8	8	1	16	1.04	0.723	(8/115)
10	8	1	16	1.53	0.0844	(8/145)

Figure 6.3: Throughput(inputs/second) vs M



Question 7

We evaluated the area and power of a layer change as *N* changes. ***Table 7.0*** below shows the values obtained after synthesis for *N* = 4, 6, 8, 10 while *M*, *P*, and *T* are constant at values *M* = 8, *P* = 1, and *T* = 16. Values for max clock period, total area, total power, and the critical path were obtained for each configuration.

As *N* varied, we observed that the critical path for the system was mostly datapath. However, when *N* > *M*, we see that the critical path is in the control logic. This is in part because we have to spend more time reading and writing the "x" data from memory.

Table 7.0: Evaluating N

M	N	P	T	Max Clock Period (ns)	Total Area (μm^2)	Total Power (µW)	Critical Path
8	4	1	16	1.01	2264.192	1613.700	mvma_8_4_1/dpth/mem_x/data_out_reg[7] to mvma_8_4_1/dpth/mult_out_reg[15]
8	6	1	16	1.04	2515.030	1727.900	mvma_8_6_1/dpth/mem_x/data_out_reg[1] to mvma_8_6_1/dpth/mult_out_reg[15]
8	8	1	16	1.04	2881.578	1919.300	mvma_8_8_1/dpth/rom_w/z_reg[0] to mvma_8_8_1/dpth/mult_out_reg[15]
8	10	1	16	1.06	3066.980	1955.000	mvma_8_10_1/ctrl/addr_w_reg[6] to mvma_8_10_1/dpth/rom_w/z_reg[0]

Figures 7.1 below shows the relationship between *Total Area (μm^2) vs N*. From the graph, we can see a slightly linear relationship between the two variables. ***Figures 7.2*** below shows the relationship between *Total Power(µW) vs N*. From the graph, we can also we can see a slightly linear relationship between the two variables.

Figure 7.1: Total Area(µm^2) vs N

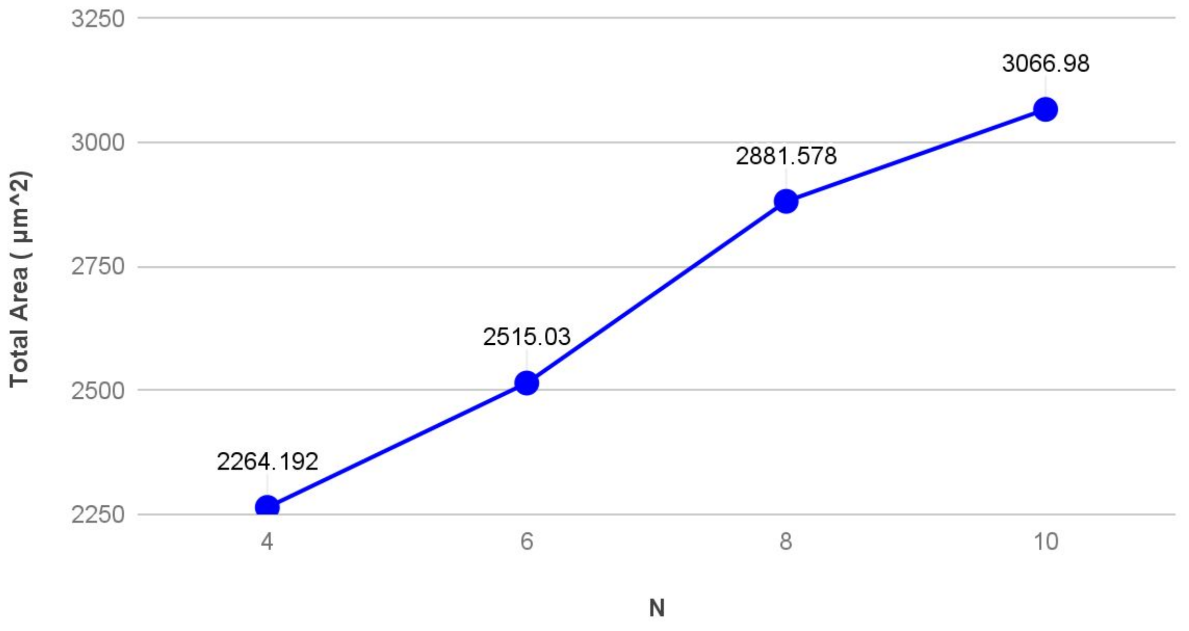
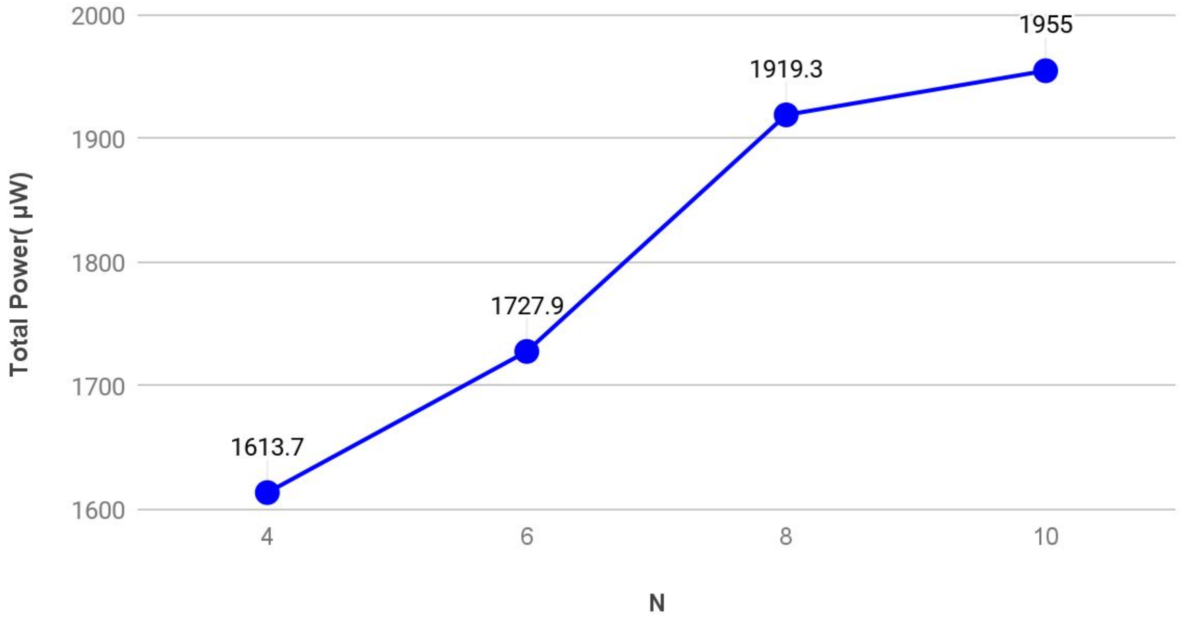


Figure 7.2: Total Power(µW) vs N



Question 8

We evaluated the area and power of a layer change as *P* changes. ***Table 8.0*** below shows the values obtained after synthesis for *P* = 1, 2,4, 8 while *M*, *N*, and *T* are constant at values *M* = 16, N = 8, and *T* = 16. Values for max clock period, total area, total power, and the critical path were obtained for each configuration.

As *P* varied, we observed that the critical path for the system stayed in the datapath. As *P* increases clock frequency increases a system becomes more expensive - taking up more area and consuming more power.

Our results show that a system does indeed become more expensive as parallelism increases. However, we did see a drastic increase in the clock frequency of our system.

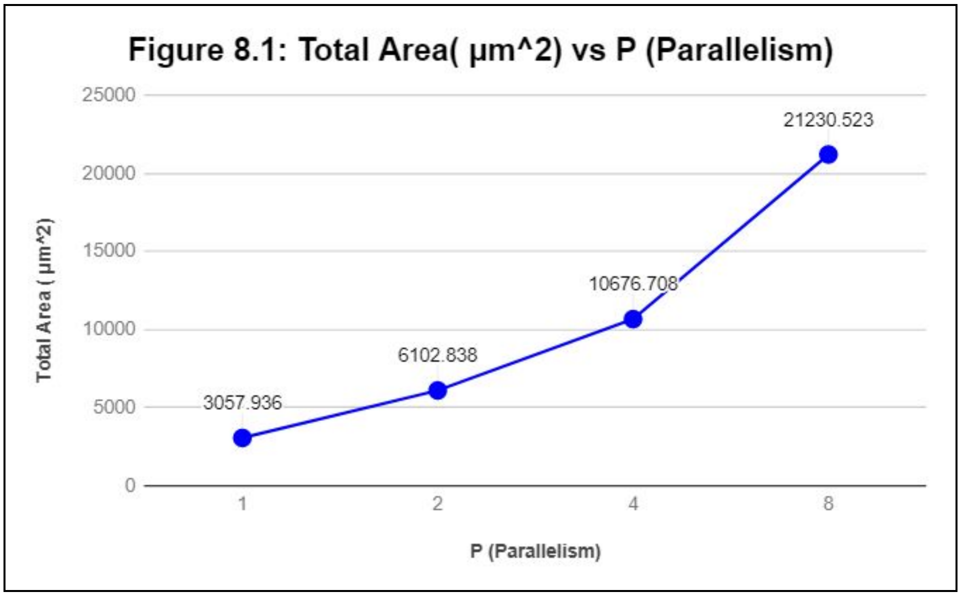
The best design out of this system would be when *P* = 4. This configuration achieved higher clock frequency than when *P = 1* and *P = 2. When P = 8,* our system did not see an improvement in clock frequency from *P = 4*, therefore was the least efficient of the bunch.

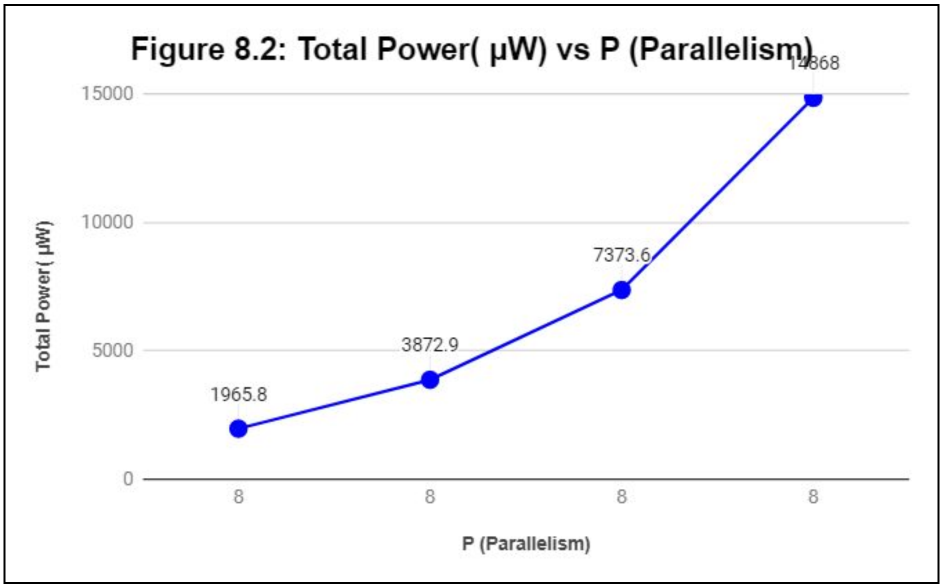
Table 8.0: Evaluating Parallelism (P)

M	N	P	T	Max Clock Period (ns)	Total Area (μm^2)	Total Power (μW)	Critical Path
16	8	1	16	1.03	3057.936	1965.800	mvma_16_8_1/dpth/mem_x/data_out_reg[5] to mvma_16_8_1/dpth/mult_out_reg[15]
16	8	2	16	1.04	6102.838	3872.900	mvma_16_8_1/dpth/mem_x/data_out_reg[3] to mvma_16_8_1/dpth/mult_out_reg[15]
16	8	4	16	0.95	10676.708	7373.600	mvma_16_8_3/dpth/mem_x/data_out_reg[1] to mvma_16_8_3/dpth/mult_out_reg[14]

16	8	8	16	0.95	21230.524	148680.000	mvma_16_8_5/dpth/mem_x/data_out_reg[3] to mvma_16_8_5/dpth/mult_out_reg[15]

Figures 8.1 below shows the relationship between *Total Area (μm^2) vs P*. From the graph, we can see that as *P* increases. ***Figures 7.2*** below shows the relationship between *Total Power(μW) vs M*. From the graph, we can also observe the unexpected relationship between these two variables.





Question 9

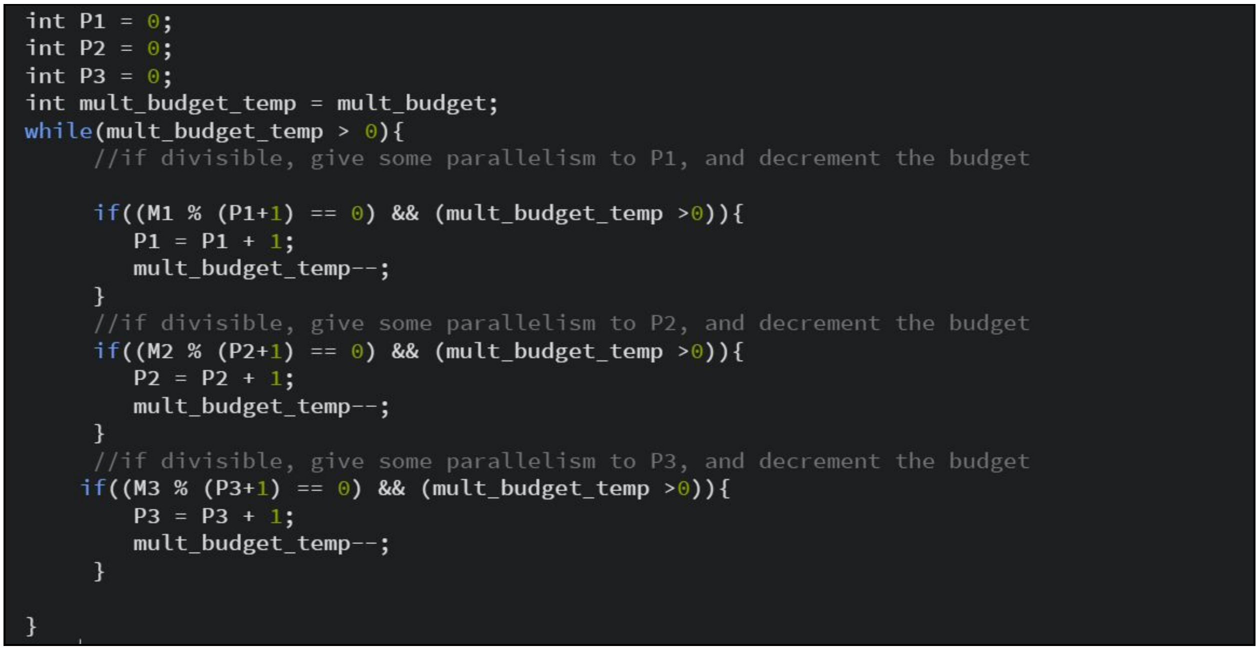
Q9: In Part 2, we defined parallelism for this system as having P parallel multiplyaccumulate units operating on P outputs concurrently. However, this limits us to using only N multipliers per layer. If you wanted to parallelize further, what could you do?

A9: In order to implement parallelism, our code had to generate multiple matrix-vector-multiplication-and-addition (mvma) modules. Our code depended heavily on the parameter, P, in order to function correctly. Each mvma module shared the same *data_in*, *s_valid, and m_ready* inputs but their own outputs for *s_ready, m_valid,* and *data_ou*t. The top module *layer_M_N_P_T* had to instantiate *P* amount of *mvma* modules. An *output_count* was used in order to keep track of which *mvma* module to output at a time. If we wanted to parallelize further,

Question 10

Q10: In Part 3, you were tasked with figuring out how to best optimize the parallelism parameters for your three-layer network, given a multiplier budget L. Explain how you did this, and why this is a good solution. Do you see any drawbacks to your approach?

A10: In order to optimize the parallelism parameters for the three layer network, we had to distribute the amount of multipliers among budget L. *Figure 10* below shows the logic we used to implement this. Here, we initialize P1, P2, and P3 to 0 and a temporary value that will be used to decrement through the multiplication budget. Using a while loop, we check if M1 is evenly divisible by the next P1 value, if it is divisible, we give add some parallelism to P1 and decrement the multiplication budget. We then check if M2 is evenly divisible by the next P2 value, if it is divisible, we give add some parallelism to P2 and decrement the multiplication budget. We then check if M3 is evenly divisible by the next P3 value, if it is divisible, we give add some parallelism to P3 and decrement the multiplication budget. We do this until the multiplication budget is depleted.



```
int P1 = 0;
int P2 = 0;
int P3 = 0;
int mult_budget_temp = mult_budget;
while(mult_budget_temp > 0){
     //if divisible, give some parallelism to P1, and decrement the budget

     if((M1 % (P1+1) == 0) && (mult_budget_temp >0)){
        P1 = P1 + 1;
        mult_budget_temp--;
     }
     //if divisible, give some parallelism to P2, and decrement the budget
     if((M2 % (P2+1) == 0) && (mult_budget_temp >0)){
        P2 = P2 + 1;
        mult_budget_temp--;
     }
     //if divisible, give some parallelism to P3, and decrement the budget
    if((M3 % (P3+1) == 0) && (mult_budget_temp >0)){
        P3 = P3 + 1;
        mult_budget_temp--;
     }

}
```

Figure 10: Multiplication Budget Distribution

Question 11

Q11: Lastly, we will evaluate the optimization method you developed for Part 3. For this experiment, set N=4, M1=8, M2=12, and M3=16. Set T=16. Then, generate and synthesize five different designs, with L = 3, 10, 20, 30, and 36. Graph (1) power versus L, (2) area versus L, and (3) throughput versus L. When you calculate throughput here, be careful to measure the time once the system has filled the pipeline. In other words, your design is a large pipeline; at the very start of execution, the pipeline is entirely empty, so the design may be faster at this time than during normal operating conditions. A good test is to also measure the amount of time that elapses between the system producing outputs. If you count Page 15 that a new input vector can start every c cycles, you should also see that the system should produce a new output vector every c cycles as well.

A11:

Table 10.0: Evaluating L

N	M1	M2	M3	T	L	Max Clock Period (ns)	Total Area (μm^2)	Total Power (μW)	Critical Path
4	8	12	16	16	3				
4	8	12	16	16	10				
4	8	12	16	16	20				
4	8	12	16	16	30				

Question 12

Q12: In Part 3, your system assumed it would produce a network with exactly three layers. How would you adapt your generator to create a system with an arbitrary (user-specified) number of layers? What challenges would it create?

A12: In order to produce exactly three layers, we had to rename the modules accordingly. Since everything is generated into one file, we had to name the modules according to the parameters it will be called with. Depending on the input parameters N, M1, M2, M3, L, and T, we had to name each module systematically. This included the mvma, control, datapath, and memory modules. Prior to part 3, these modules shared the same name. Since each layer has its own mvma, control, datapath, and memory modules to instantiate, each of these modules had to be renamed.