ESE 507 – Proj3

-Aditya Sharma (112676654)
-Mukul Javadekar (112961738)

1. In hardware generators, scalability and flexibility can be difficult. In your report, explain how you made your generator capable of handling the flexibility required by the parameters (vector sizes M and N, parallelism P, number of bits T). Were any of these particularly easy or difficult to support? How did you handle the fact that there were no maximum defined values on M and N? Are there practical limits on these values? Why or why not?

   Answer:
   For handling flexibility required by the parameters (vector sizes M and N, parallelism P, number of bits T) we have used parameterization. The .sv file is generated from the .cc code which accommodates input given by the user in the generated code. The current design is a generic design, once it was working for a constant set of parameters it was easy to extend it. Tool takes value for M and N from the user and generates the code, in .cc file M and N are of integer type therefore maximum value the user can give is maximum of integer type.

2. Describe how you designed your control logic. How did you structure the design so that it can be changed as the *M* and *N* parameters change? Explain how the structure changes as these parameters grow.

   Answer:
   Our current design is an extension of project 2 which had a separate data path and control path for doing 1-D convolution. For Part-1 of this project, a control logic for saturation arithmetic and ReLU is introduced in multiply and accumulate unit and control logic for loading memory f is removed. For Part-2, parallelism is added, a new control signal "op_done" is introduced to keep track when an output is completed, and this control logic is part of the data path. By changing value of M and N parameters there is not much change in the generated code as all these parameters and address bits for M and N are parameterized. As M and N will grow, bigger memory, rom, address bits will be required for the design.

3. 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?

   Answer:
   For the parallel (P>1) designs, as P increases the number of memories for x, MAC units increase. The design follow the same approach as for P = 1 along with that "op_done" signal and "counter2" is introduced. Since outputs for different MAC's is generated together but we have only one output port, this signal indicates if the output of current computation is completed, the counter counts how many outputs are completed. One optimization which we could think of to reduce memory cost is to use a single

memory for x, the memory can be modified to have many data out ports which can be controlled through separate address registers.

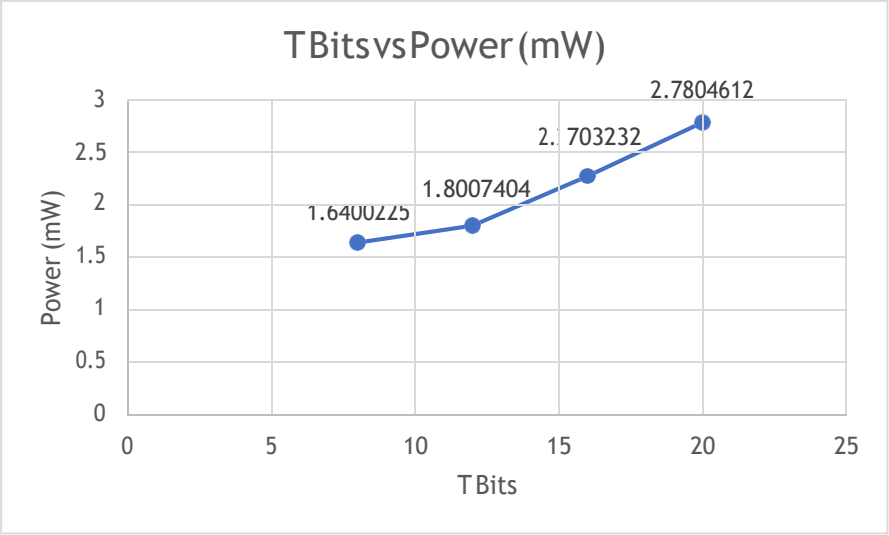
4. Our design parameters *M*, *N*, *P*, and *T* allow various tradeoffs between:
   - problem size *(e.g., how big of a convolution we can compute)*
   - costs *(e.g., area, power, energy)*
   - precision (*i.e., how many bits are used to represent numbers)*, and
   - performance (*e.g., throughput and 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.

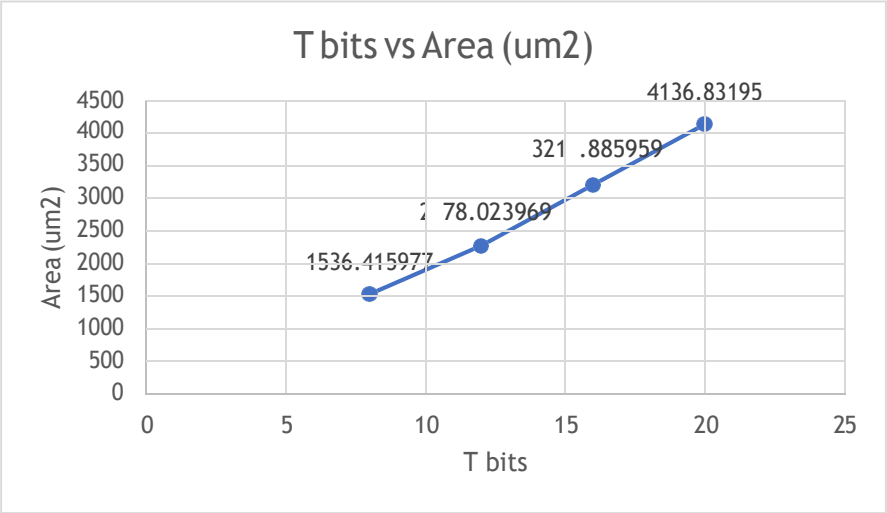
   Answer:
   As M and N are increased the problem size, area, power increases but the performance (throughput) reduces. As P is increased the performance (throughput) improves but power and area is increased significantly. As T is increased precision is improved but area, power increases.

5. Now, we will use synthesis to evaluate how the area and power of a convolution module change as the data precision *T* changes. Use your generator to produce four designs (Part 1) with *T* = 8, 12, 16, and 20, while you keep the other parameters constant: (*N*, *M*, *P*) = (16, 4, 1). Then produce two graphs that illustrate: (1) power versus *T* and (2) area versus *T* for these designs. Describe where the critical path is located in each design.

   Answer:
   (1) T v/s Power:



(2) T v/s Area:



Designs:
(1) conv_16_4_12_1:
Critical path :
Startpoint: d/rom/z_reg[2] (rising edge-triggered flip-flop clocked by clk)
Endpoint: d/m1/mul_temp_reg[17](rising edge-triggered flip-flop clocked by clk)
The path goes from the rom ending to mul_temp.

(2) conv_16_4_16_1:
Critical path:
Startpoint: d/x_mem1/data_out_reg[11] (rising edge-triggered flip-flop clocked by clk)
Endpoint: d/m1/mul_temp_reg[22] (rising edge-triggered flip-flop clocked by clk)
Here, the critical path is from the data_out of memory1 till mul_temp.

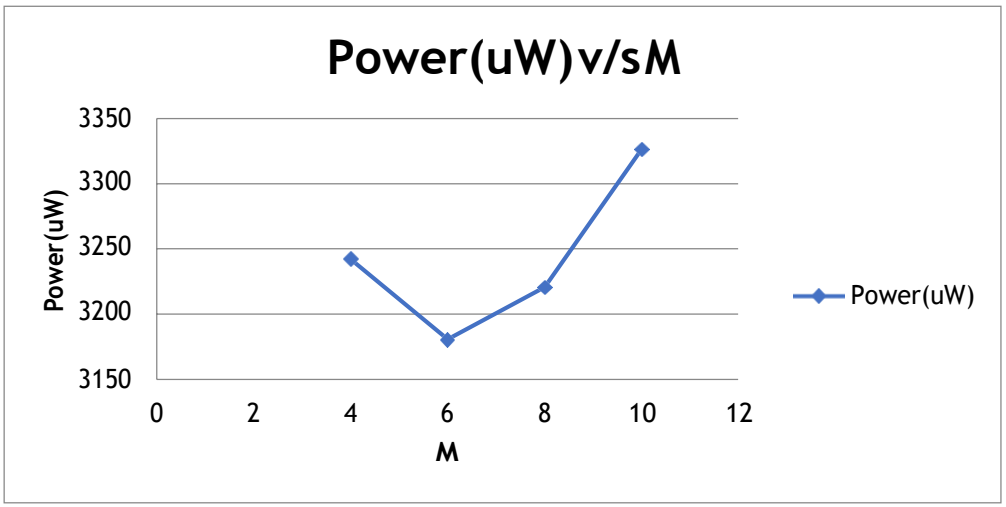
(3) conv_16_4_20_1:
Critical path:
Startpoint: d/x_mem1/data_out_reg[3] (rising edge-triggered flip-flop clocked by clk)
Endpoint: d/m1/mul_temp_reg[22] (rising edge-triggered flip-flop clocked by clk)
Here, the critical path is from the data_out of memory1 till mul_temp.

(4) conv_16_4_8_1:
Critical path:
Startpoint: d/rom/z_reg[0] (rising edge-triggered flip-flop clocked by clk)
Endpoint: d/m1/mul_temp_reg[11] (rising edge-triggered flip-flop clocked by clk)
The path goes from the rom, passing through the MAC and finally ending to mul_temp.

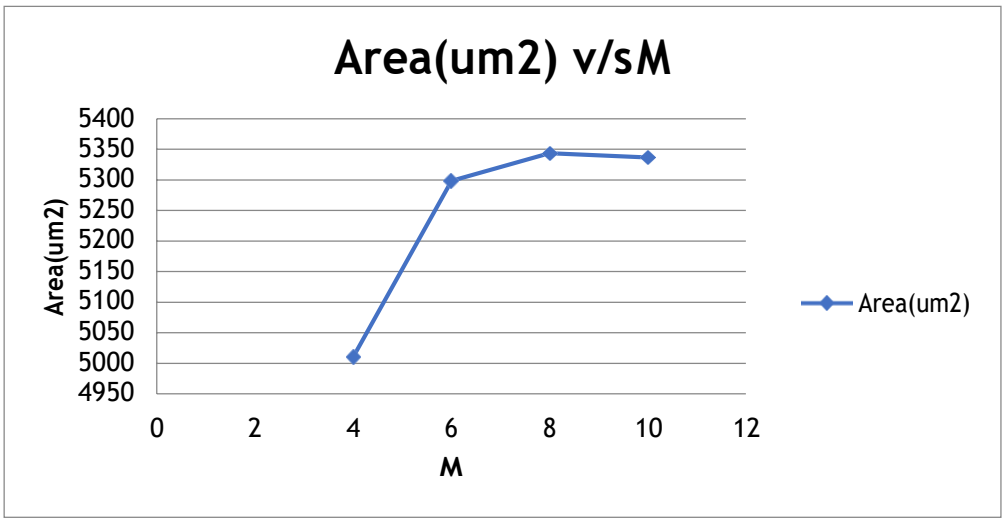
6. Next, we will evaluate how throughput, area, and power scale as M changes. Use your generator to produce four designs with $M = 4, 6, 8$, and 10, while you keep N=32, P=1 and T=16. Synthesize each design, and graph: (1) power versus M, (2) area versus M, and (3) throughput versus M. Does the location of the critical path change as M changes? In your report, include the values of c that you found for each design, and explain how you found them.

Answer:

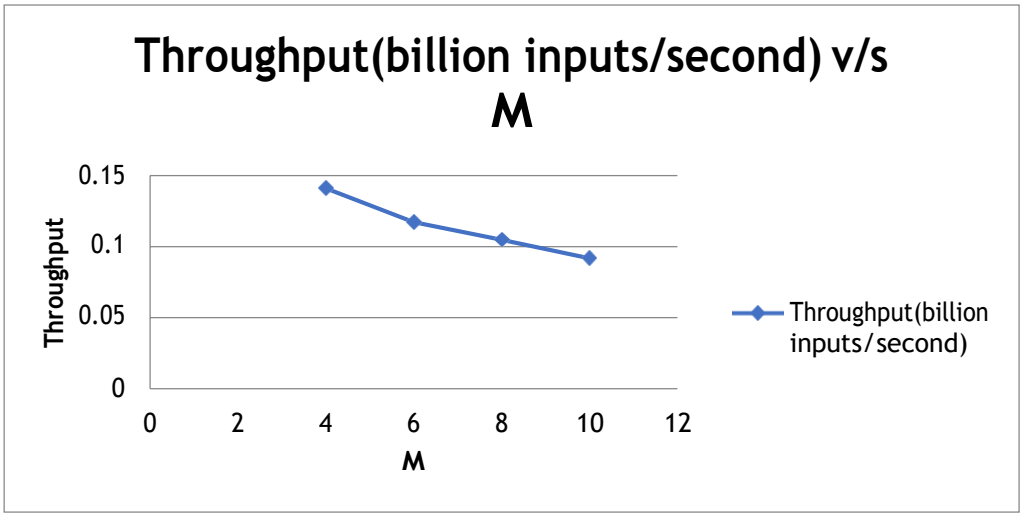
(1) Power v/s M:



(2) Area v/s M:



(3) Throughput v/s M:



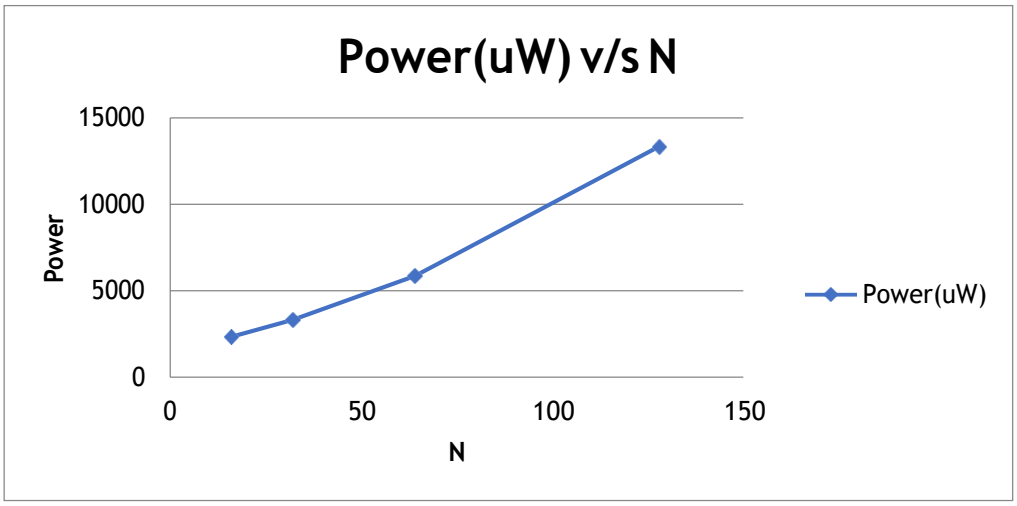
The location of critical path does not change even though we change the values of M.

'c' value for designs:
(1) conv_32_4_16_1 : c = 236
(2) conv_32_6_16_1 : c = 276
(3) conv_32_8_16_1 : c = 308
(4) conv_32_10_16_1 : c = 362
We made changes in the testbench by making the signals s_valid_x (line 42) and m_ready_y (line 55) as 1 to get the accurate values. For calculating c, we tracked time taken from inputting first set of N words to second set of N words and dividing it by the time period i.e 10nn.

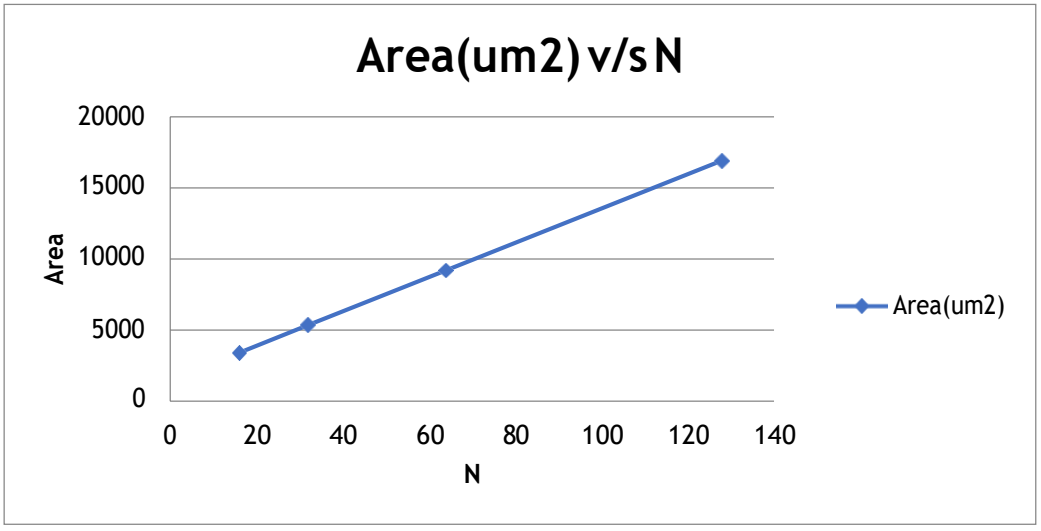
7. Now, we will repeat the previous question while *N* changes. Use your generator to produce four designs with N=16, 32, 64, and 128. Set *M*=8, *P*=1 and *T*=16. Synthesize each design, and graph: (1) power versus *N*, (2) area versus *N*, and (3) throughput versus *N*. Does the location of the critical path change as *N* changes?

Answer:

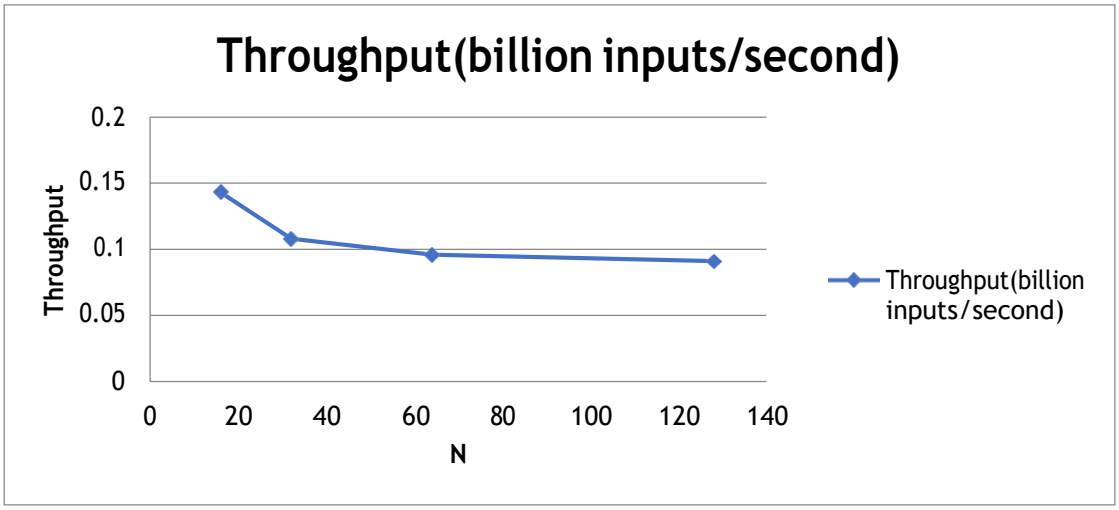
(1) Power v/s N:



(2) Area v/s N:



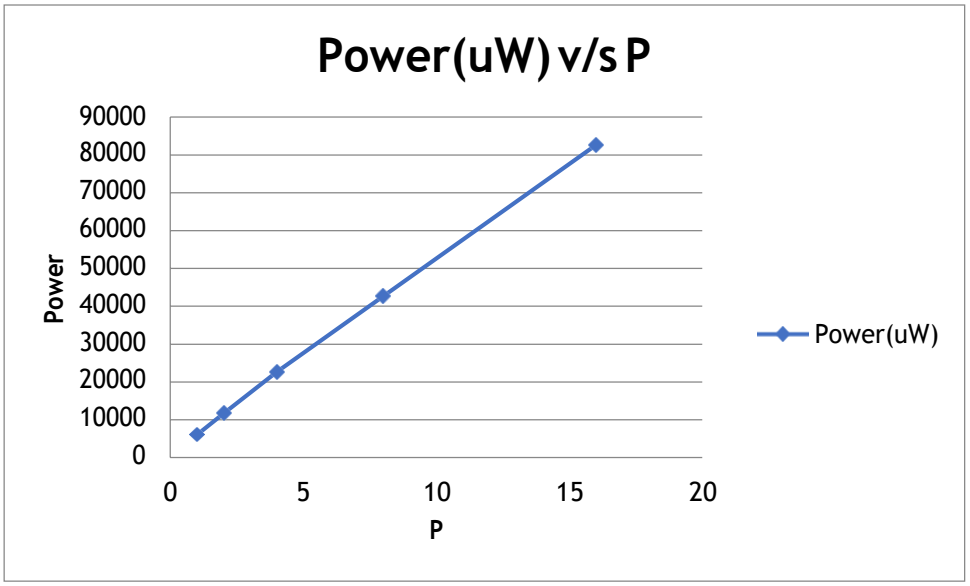
(3) Throughput v/s N:



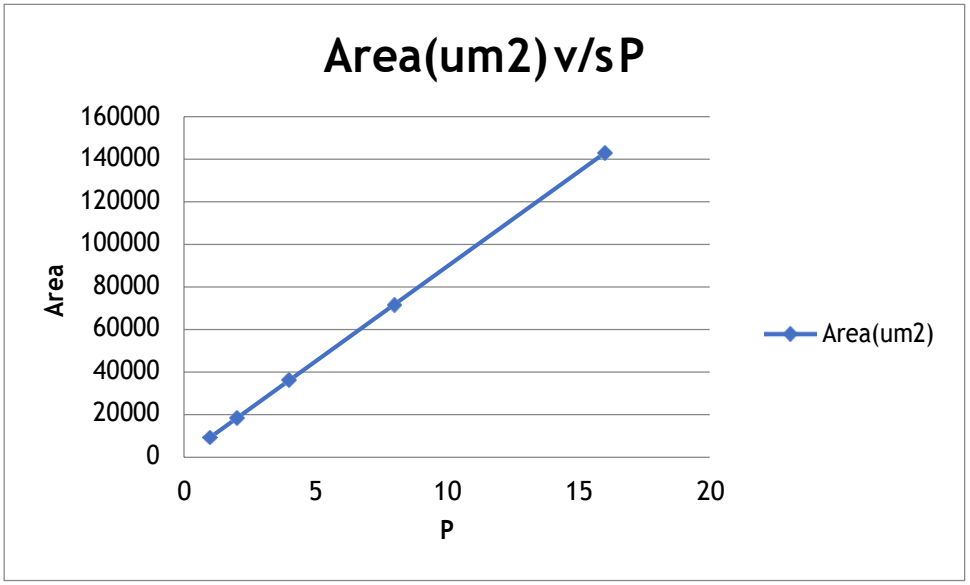
The location of critical path does not change even though we change the values of N.

8. Evaluate how the designs change when you increase parallelism, by evaluating P=1, 2, 4, 8, 16, with N=64, M=33, and T=16. Graph: (1) power versus P, (2) area versus P, and (3) throughput versus P.
As parallelism increases, the designs should get faster but also more expensive. Which of these designs is most efficient? Justify your answer quantitatively.

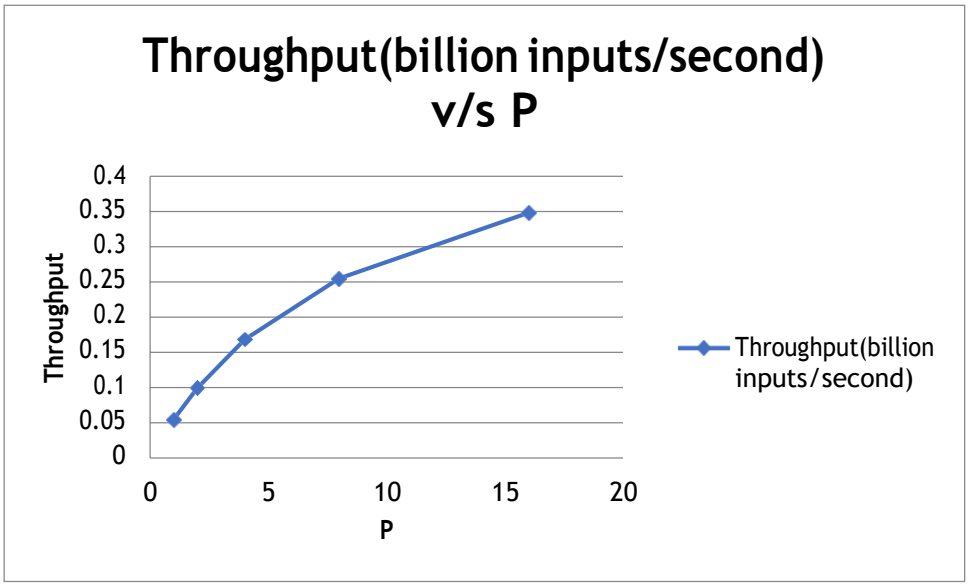
Answer:
(1) Power v/s P:



(2) Area v/s P:



(3) Throughput v/s P:



The design with P=16 is most efficient as the throughput calculated in this design maximum when compared with the other designs.

9. In Part 2, we defined parallelism for this system as having *P* parallel multiply-accumulate units operating on *P* outputs concurrently. However, this limits us to using only *L=N-M+1* multipliers per layer. If you wanted to parallelize further, what could you do?

   Answer:
   The design can be further parallelized by optimizing the individual components and introducing parallelism in those components such as multiplier and adder.

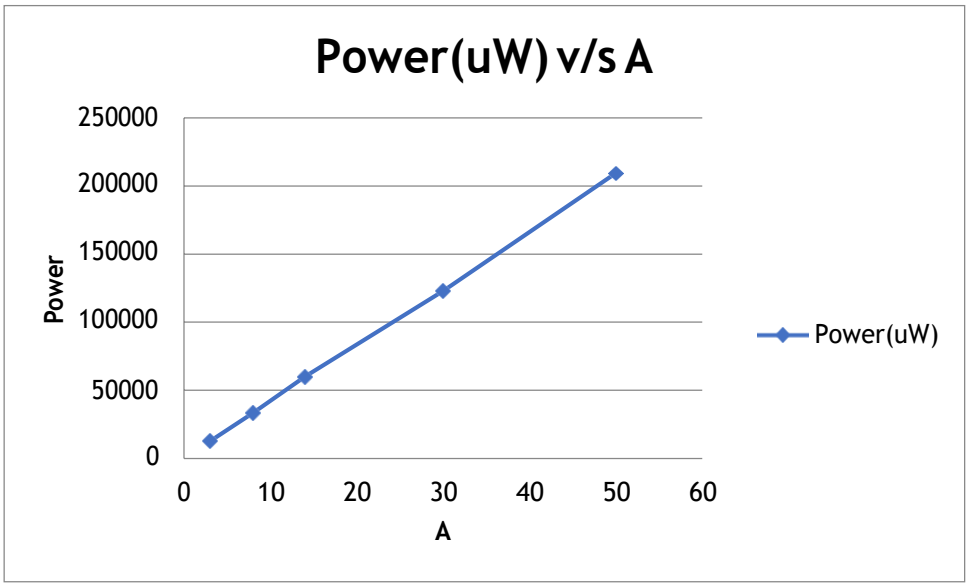
10. In Part 3, you were tasked with figuring out how to best optimize the parallelism parameters for your three-layer design, given a multiplier budget A. Explain how you did this, and why your approach is a good solution. Do you see any drawbacks to your approach? How did you evaluate whether or not your system worked well?

    Answer:
    For optimizing the parallelism in three-layer design we should be able to distribute maximum of A multipliers to the 3 layers while maintaining the L/P ratio for all the layers. In our approach first we are allocating maximum of L1/P of A-2 multipliers to layer one suppose x_a, then maximum of L2/P of A-x_a–1 multipliers to layer two suppose x_b and maximum of L3/P of A-x_a-x_b multipliers to layer three. In most cases we were able to assign maximum multipliers.

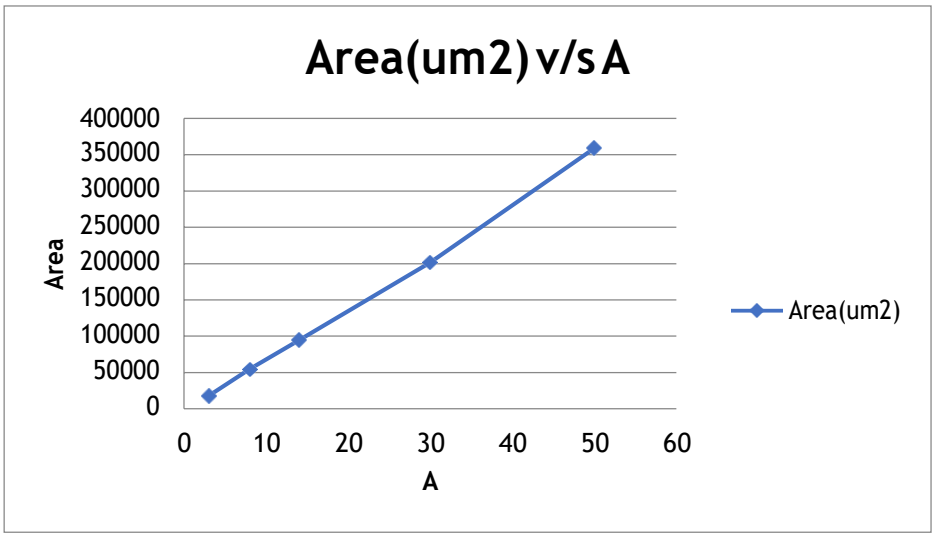
11. Lastly, we will evaluate the optimization method you developed for Part 3. For this experiment, set *N*=64, *M1*=33, *M2*=9, and *M3*=10. Set *T*=16. Then, generate and synthesize five different designs, with *A* = 3, 8, 14, 30, and 50. Graph (1) power versus *A*, (2) area versus *A*, and (3) throughput versus *A*.

Answer:

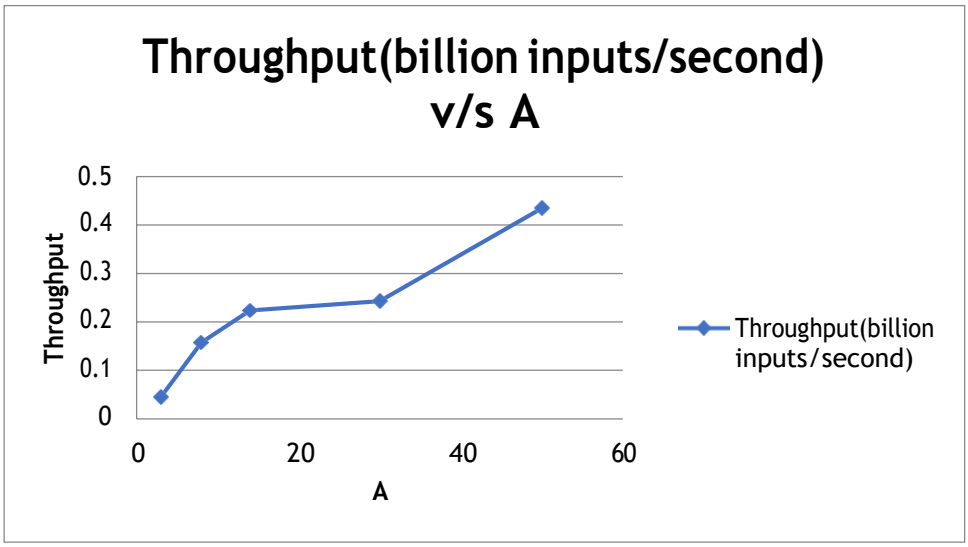
(1) Power v/s A:



(2) Area v/s A:



(3) Throughput v/s A:



12. In Part 3, your system assumed it would produce a system 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?

Answer:
For making the generator to create a system with an arbitrary number of layers the control logic for connecting has to be made generic and adapts to the number of layers entered by the user. The optimization for the number of multipliers will also has to be made compatible with the number of layers entered by the user.

13. If you worked with a partner: please explain each partner's contribution to the project. Be specific about what each partner did.

Answer:
Aditya Sharma (112676654) - Coding, debugging. Question 1, 2, 3, 4, 5, 9, 10, 12, and 13 of the report.

Mukul Javadekar (112961738): Debugging. Question 6, 7, 8, 11 of the report.