# Links

| | |
|---|---|
| Pynq Demo | https://drive.google.com/file/d/1mvL4MxIF26IdchtnAAkjzjug9ciUHM0C/view?usp=sharing |
| GitHub | https://github.com/JDRadatti/cse237c/tree/main |
| Throughput Calculations | https://github.com/JDRadatti/cse237c/blob/main/scripts/calc_throughput.py |

# Question 1:

**1.a.)** Now think about if you were to use a custom CORDIC algorithm to calculate cos() and sin() (you don't have to implement this). Would changing the accuracy of your CORDIC core make the DFT hardware resource usage change? How would it affect the performance?

Yes, changing the accuracy of the CORDIC core will change the DFT hardware resource usage under some circumstances. In order to increase accuracy, we need to increase the bit width. For instance, if we want to increase accuracy while keeping the function's latency—and thus the performance—approximately constant and the overhead small, then we need more resources. On the other hand, if we reuse some of the resources to not increase the resource utilization during calculation (low parallelism), the latency of CORDIC will increase, which degrades the performance.

# Question 2:

**2.a.)** Make a table that shows the change in resource utilization and performance between Question 1 and 2.

| | **Latency (Cycles) (Min, Max)** | **Interval (Cycles) (Min, Max)** | **Throughput (Hertz)** | **Resources (BRAM_18K\|DSP\|FF\|LUT )** |
|---|---|---|---|---|
| **Question 1** | 54884, 55908 | 54885, 55909 | 2.5 KHz | 2\| 53\| 5275\| 9111 |
| **Question 2** | 257, 257 | 258, 258 | 534.2 KHz | 66\| 640\| 59208\| 91730\| |

# Question 3:

**3.a.)** Why did we do this? Does it affect what optimizations you can perform?

By separating the input and output into separate arrays, we eliminate the need for a temporary array. This allows us to eliminate a loop, which results in a speed up. Also, separating the input and output into separate arrays allows for more parallelism because the input and output are independent of each other. This way, port bottlenecks can be eliminated. Reads and writes can be executed concurrently. Moreover, optimizing each array independently is possible including array partitioning.

**3.b.)** Make a table that shows the resource utilization and performance from before and after this change.

| | **Latency (Cycles) (Min, Max)** | **Interval (Cycles) (Min, Max)** | **Throughput (Hertz)** | **Resources (BRAM_18K\|DSP\|FF\|LUT )** |
|---|---|---|---|---|
| **Question 2 (Before)** | 257, 257 | 258, 258 | 534.2 KHz | 66\| 640\| 59208\| 91730 |
| **Question 3 (After)** | 221, 221 | 222, 222 | 620.8 KHz | 64\| 640\| 59519\| 91649 |

**3.c.)** Describe the results you see.

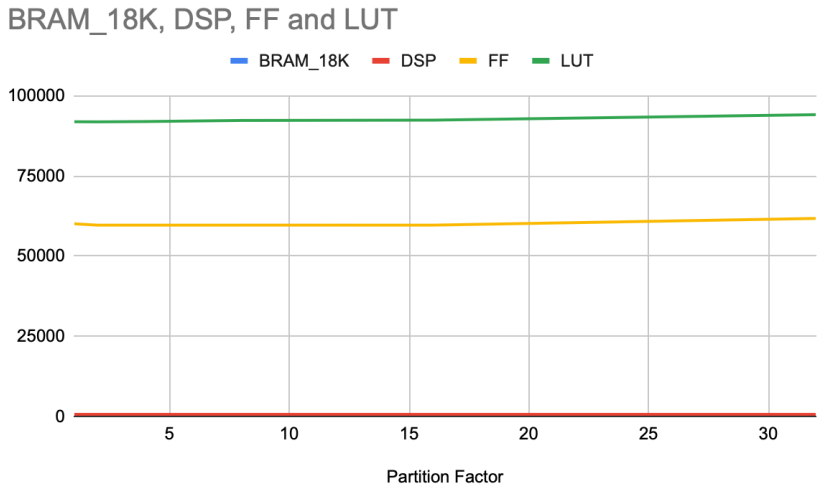
After splitting the input and output into separate arrays, we observed a slight speedup. This is mainly because we eliminated a for-loop and did not have to create temporary arrays. The FF usage slightly increased and the number of LUTs slightly decreased. The decrease in LUTs was likely due to the removed for-loop as well as slightly less logic. And most probably the FF number increased due to control overhead. Lastly, BRAM number decreased because hardware no longer has to create copies of certain data to read and write concurrently.

# Question 4

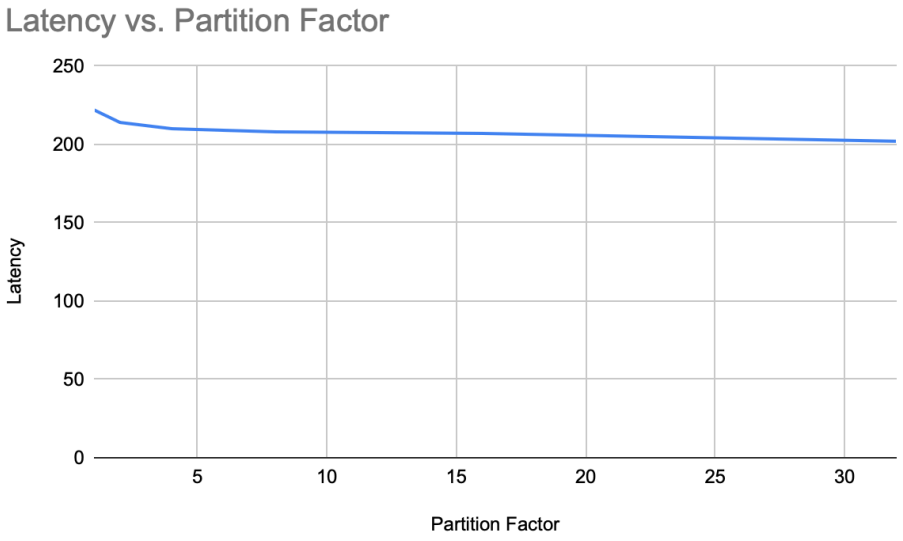
**4.a.)** Use block partitioning. Try factors of 1 (i.e. without partitioning), 2, 4, 8, 16, and 32. Make a table showing the achieved II, resource utilization, and performance of each of these implementations.

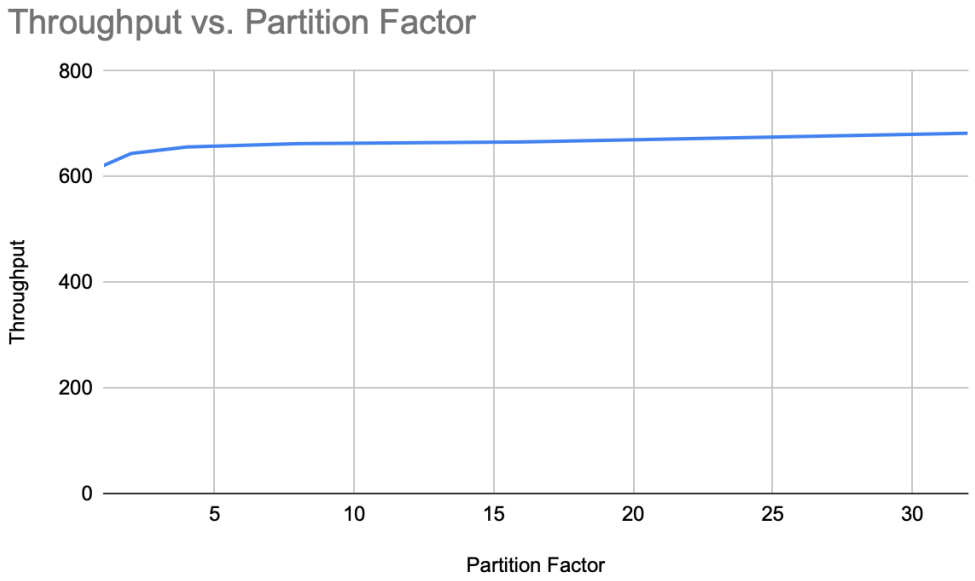
| Partition Factor | Achieved II (Cycles) (Min, Max) | Latency (Cycles) (Min, Max) | Interval (Cycles) (Min, Max) | Throughput (Hertz) | Resources (BRAM_18K\|DSP\|FF\|LUT ) |
|---|---|---|---|---|---|
| **1** | 32, 32 | 221, 221 | 222,222 | 620.8 KHz | 64\| 640\| 60081\| 91905 |
| **2** | 32, 32 | 213, 213 | 214, 214 | 644.0 KHz | 64\| 640\| 59650\| 91867 |
| **4** | 32, 32 | 209, 209 | 210, 210 | 656.3 KHz | 64\| 640\| 59654\| 91965 |
| **8** | 32, 32 | 207, 207 | 208, 208 | 662.6 KHz | 64\| 640\| 59660\| 92301 |
| **16** | 32, 32 | 206, 206 | 207, 207 | 665.8 KHz | 64\| 640\| 59659\| 92406 |
| **32** | 1, 1 | 203, 203 | 202, 202 | 682.3 KHz | 64\| 640\| 61743\| 94110 |

**4.b.)** Plot resource utilization vs the partition factor on one plot.



**4.c.)** Plot throughput & latency vs the partition factor on separate plots.



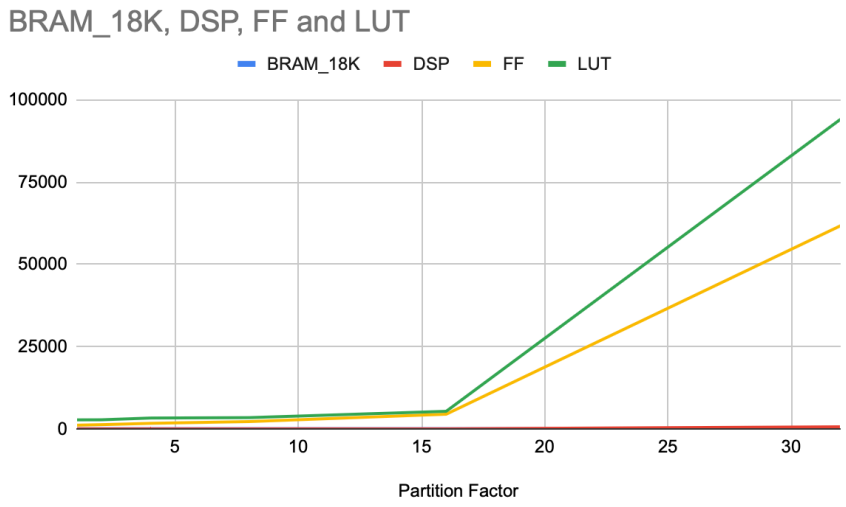



# Question 5

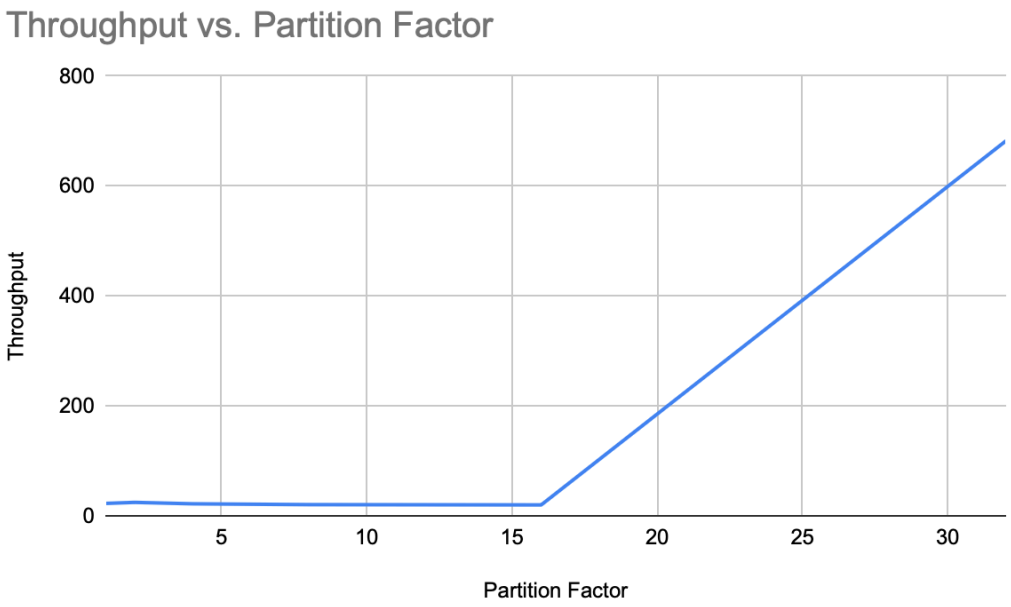
**5.a.)** Try factors of 1 (i.e. without unrolling), 2, 4, 8, 16, and 32. Make a table showing the achieved II, resource utilization, and performance of each of these implementations.

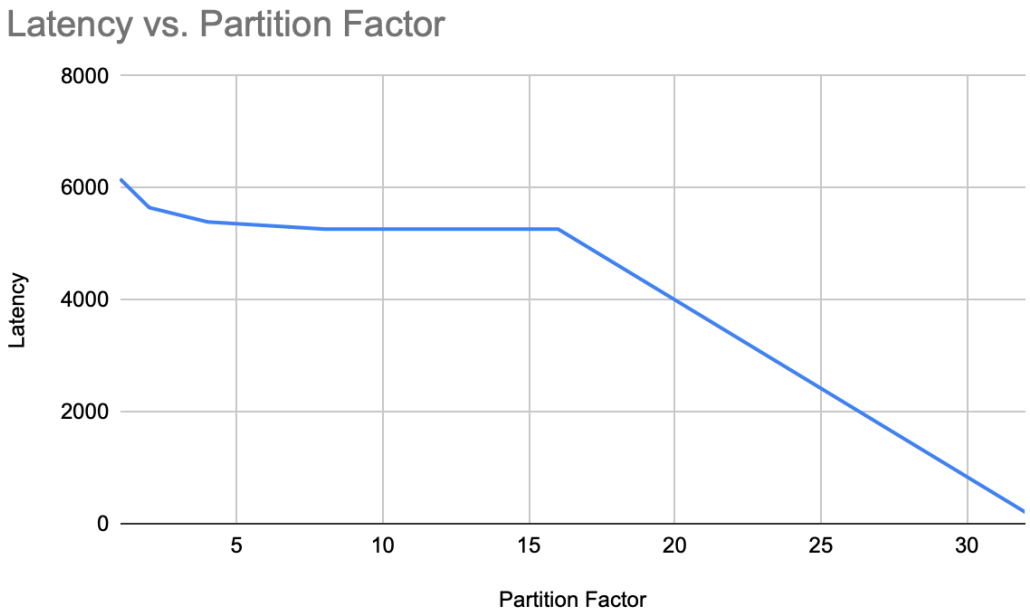
| Unrolling Factor | Achieved II (Cycles) (Min, Max) | Latency (Cycles) (Min, Max) | Interval (Cycles) (Min, Max) | Throughput (Hertz) | Resources (BRAM_18K\|DSP\|FF\|LUT ) |
|---|---|---|---|---|---|
| **1** | 6, 6 | 6159, 6159 | 6163, 6163 | 22.5 KHz | 64\| 5\| 1116\| 2740 |
| **2** | 11, 11 | 5647, 5647 | 5656, 5656 | 24.4 KHz | 64\| 5\| 1280\| 2760 |
| **4** | 21, 21 | 5391, 5391 | 5410, 5410 | 21.8 KHz | 64\| 7\| 1687\| 3299 |
| **8** | 41,1 (achieved, target) | 5263, 5263 | 5302, 5302 | 20.3 KHz | 64\| 7\| 2218\| 3415 |
| **16** | 41,1 (achieved, target) | 5263, 5263 | 5302, 5302 | 19.8 KHz | 64\| 7\| 4461\| 5341 |
| **32** | 1,1 | 203, 203 | 202, 202 | 682.3 KHz | 64\| 640\| 61743\| 94110 |

**5.b.)** Plot resource utilization vs the partition factor on one plot.



**5.c.)** Plot throughput & latency vs the partition factor on separate plots.



# Question 6

**6.a.)** Write a baseline DFT1024 using the sin() and cos() math functions. Do not apply any HLS pragmas. Report latency, throughput, and resource utilization.

| **Latency (Cycles) (Min, Max)** | **Interval (Cycles) (Min, Max)** | **Throughput (Hertz)** | **Resources (BRAM_18K\|DSP\|FF\|LUT )** |
|---|---|---|---|
| 91242497, 97533953 | 91242498, 97533954 | 1.51044 Hz | 16\| 197\| 13361\| 15801 |

**6.b.)** A full 2D lookup table no longer fits on a PYNQ-Z2 board. We can only store a 1D array of precomputed sin() and cos() values. Re-write the baseline DFT1024 to use the pre-computed values. Report the latency, throughput and resource utilization.

| **Latency (Cycles) (Min, Max)** | **Interval (Cycles) (Min, Max)** | **Throughput (Hertz)** | **Resources (BRAM_18K\|DSP\|FF\|LUT )** |
|---|---|---|---|
| 6291472, 6291472 | 6291476, 6291476 | 21.9 Hz | 4\| 5\| 1390\| 1430 |

# Question 7

**7.a.)** The baseline DFT1024 from Figure 4.15 of the textbook has data dependencies in the inside loop, which could limit parallelism. One way to tackle this issue is to interchange the two loops. Implement this change and report the latency, throughput, and resource.

| **Latency (Cycles) (Min, Max)** | **Interval (Cycles) (Min, Max)** | **Throughput (Hertz)** | **Resources (BRAM_18K\|DSP\|FF\|LUT )** |
|---|---|---|---|
| 101719041, 108010497 | 101719042, 108010498 | 1.275 Hz | 16\| 197\| 13298\| 15787 |

# Question 8

**8.a.)** Try any optimization techniques and describe your methodology.

Added array partitioning to all the arrays and pipelined the inner loop.

**8.b.)** Report the latency, throughput, and resource utilization of your best design. Your design must fit on the PYNQ-Z2 board, which means all resource utilizations must be less than 100%.

| **Latency (Cycles) (Min, Max)** | **Interval (Cycles) (Min, Max)** | **Throughput (Hertz)** | **Resources (BRAM_18K\|DSP\|FF\|LUT )** |
|---|---|---|---|
| 4194322, 4194322 | 4194323, 4194323 | 26.952 Hz | 32\| 5\| 1195\| 1703 |

# Question 9

9.a.) Report the latency, throughput, and resource utilization of your design. Resource utilization must be under 100%.

| **Latency (Cycles) (Min, Max)** | **Interval (Cycles) (Min, Max)** | **Throughput (Hertz)** | **Resources (BRAM_18K\|DSP\|FF\|LUT )** |
|---|---|---|---|
| 4195352, 4195352 | 4195353, 4195353 | 26.952 Hz | 64\| 5\| 1295\| 3096 |