Q1 a) Now think about if you were to use a custom CORDIC algorithim 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?

Higher accuracy would result in higher LUT usage to account for the pre-calculated rotation table. Lower accuracy would do the inverse, lower LUT usage.

How would it affect performance?

Higher accuracy, which leads to higher LUT usage, would lead to higher latency and lower throughput.

Q2 DFT32 Table Lookup Rewrite the code to eliminate these math function calls (i.e. cos() and sin()) by utilizing a table lookup. Use the provided *_2D.h file.
a) Make a table that shows the change in resource utilization and performance with between Question 1 and 2.

Implementation	Clock	RMSE r	RMSE i	Latency Min	Latency Max	Interval Min	Interval Max	Throughput (II*period)^-1 samples per sec	BRAM	DSP	FF	LUT	URAM
Baseline	1.00E-08	0.000143860568	0.000162503522	54884	55908	54885	55909	1.79E+03	2	53	5229	9127	0
Lookup Table	1.00E-08	0.000002198110	0.000001827443	257	257	258	258	3.88E+05	66	640	59208	91730	0

Q3 DFT32 Interface Change Modify the DFT function interface so that the input and outputs are stored in separate arrays. Modify the testbench to accommodate this change to DFT interface.

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

Separating the read from write operations allows us to pipeline more efficiently. Throughput improves.

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

Implementation	Clock	RMSE r	RMSE i	Latency Min	Latency Max	Interval Min	Interval Max	Throughput (II*period)^-1 samples per sec	BRAM	DSP	FF	LUT	URAM
Lookup Table	1.00E-08	0.000002198110	0.000001827443	257	257	258	258	3.88E+05	66	640	59208	91730	0
Interface Change	1.00E-08	0.000002198110	0.000001827443	221	221	222	222	4.50E+05	64	640	59381	91624	0

(c) Describe the results you see.

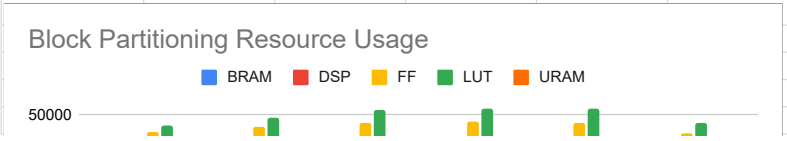
As expected, interval is reduced slightly and throughput improves.

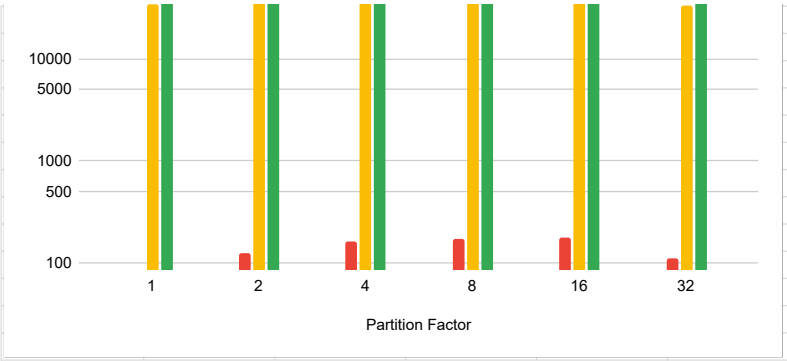
Q4 DFT32 Array Partitioning Experiment with array partitioning.
Partition all arrays in your implementation.
For now, make sure that your loops are being pipelined with a target II of 1.

(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.

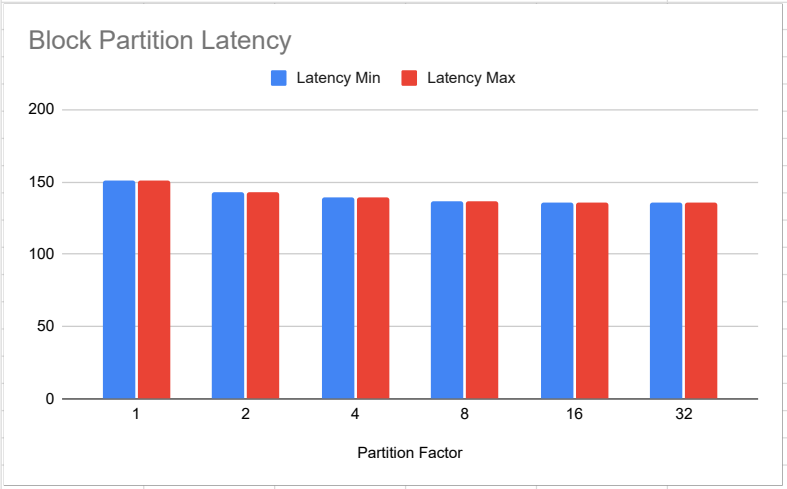
Partition Factor	Clock	RMSE r	RMSE i	Latency Min	Latency Max	Interval Min	Interval Max	Throughput (II*period)^-1 samples per sec	BRAM	DSP	FF	LUT	URAM
1	1.00E-08	0.000002198110	0.000001827443	151	151	152	152	6.58E+05	0	85	33759	39194	0
2	1.00E-08	0.000002198110	0.000001827443	143	143	144	144	6.94E+05	0	125	38523	46434	0
4	1.00E-08	0.000002198110	0.000001827443	139	139	140	140	7.14E+05	0	165	42459	55593	0
8	1.00E-08	0.000002198110	0.000001827443	137	137	138	138	7.25E+05	0	173	43341	57271	0
16	1.00E-08	0.000002198110	0.000001827443	136	136	130	130	7.69E+05	0	177	41537	57254	0
32	1.00E-08	0.000002198110	0.000001827443	136	136	129	129	7.75E+05	0	111	33011	42479	0

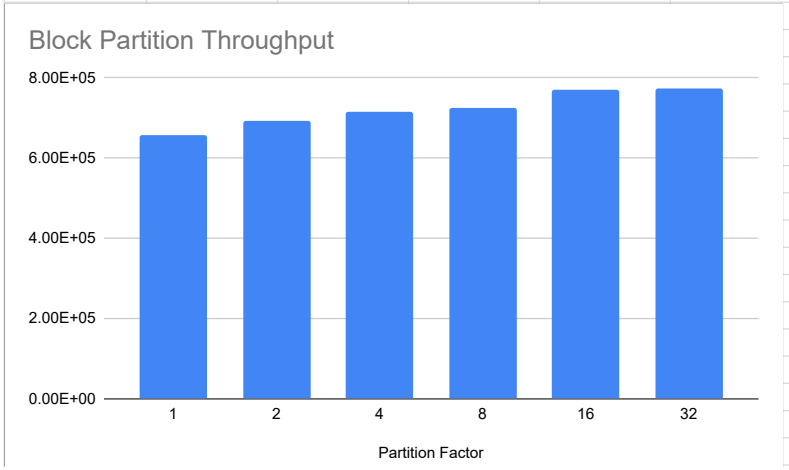
(b) Plot resource utilization vs the partition factor on one plot.





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



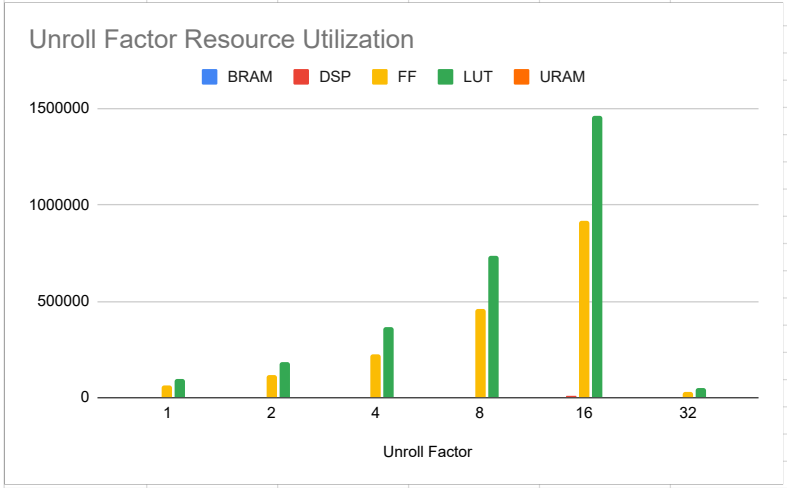


Q5 DFT32 Loop Unrolling Experiment with loop unrolling. Unroll the inner loop only.
Use the best array partitioning from Question 4. In order to prevent automatic unrolling by Vitis HLS, you will need to remove the #pragma HLS PIPELINE pragmas from both loops.
Double check that your explicit unrolling is being applied by looking at the synthesis report.

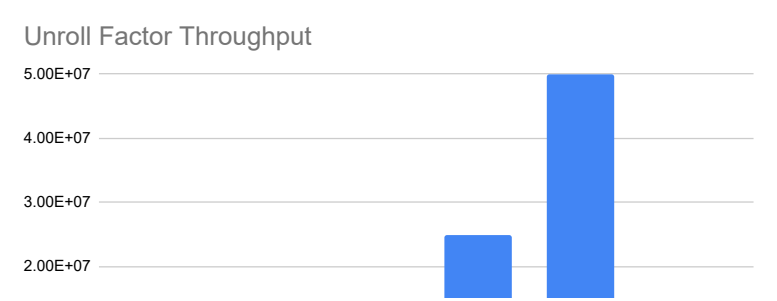
(a) Try factors of 1 (i.e. without unrolling), 2, 4, 8, 16, and 32.
Make a table showing the achieved II, resource utilization, and performance of each of these implementations.

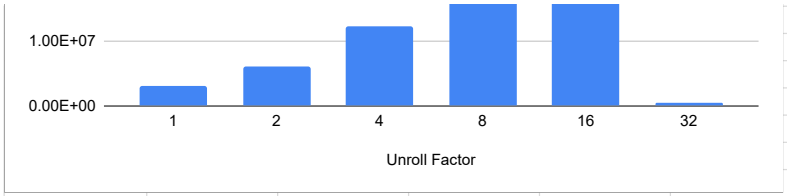
Unroll Factor	Clock	RMSE r	RMSE i	Latency Min	Latency Max	Interval Min	Interval Max	Throughput (II*period)^-1 samples per sec	BRAM	DSP	FF	LUT	URAM
1	1.00E-08	0.000002198110	0.000001827443	203	203	32	32	3.13E+06	64	640	61701	94174	0
2	1.00E-08	0.000002198110	0.000001827443	187	187	16	16	6.25E+06	64	1280	116795	185308	0
4	1.00E-08	0.000002198110	0.000001827443	179	179	8	8	1.25E+07	256	2560	227001	367720	0
8	1.00E-08	0.000002198110	0.000001827443	175	175	4	4	2.50E+07	0	5120	462513	732760	0
16	1.00E-08	0.000002198110	0.000001827443	173	173	2	2	5.00E+07	0	10240	918181	1462066	0
32	1.00E-08	0.000002198110	0.000001827443	208	208	209	209	4.78E+05	0	160	28387	48918	0

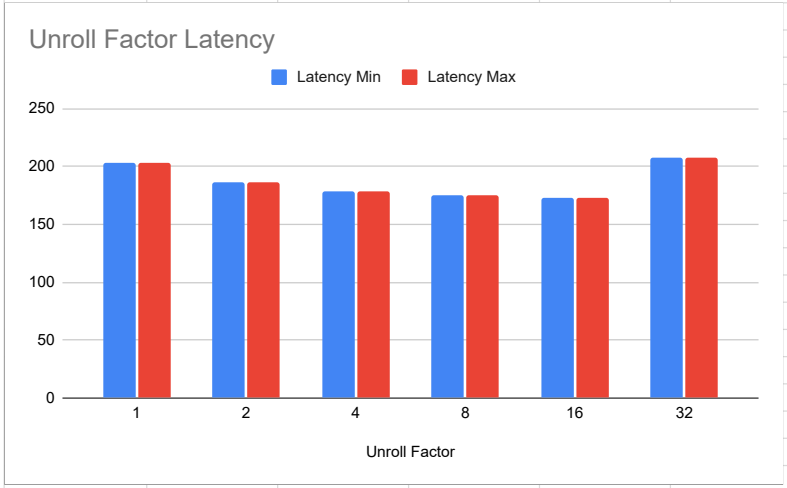
(b) Plot resource utilization vs the unroll factor on one plot.



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







Q6 DFT1024 Baseline You should refer to the baseline DFT code at Figure 4.15 of the textbook.

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

Clock	RMSE r	RMSE i	Latency Min	Latency Max	Interval Min	Interval Max	Throughput (II*period)^-1 samples per sec	BRAM	DSP	FF	LUT	URAM
1.00E-08	1.491802692	0.8630965352	55592961	56641537	55592962	56641538	1.77E+00	0	53	5237	8959	0

(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.

Clock	RMSE r	RMSE i	Latency Min	Latency Max	Interval Min	Interval Max	Throughput (II*period)^-1 samples per sec	BRAM	DSP	FF	LUT	URAM
1.00E-08	0.001853297697	0.001314980211	6291472	6291472	6291456	6291456	1.59E+01	4	5	1230	1415	0

Q7 DFT1024 Loop optimization 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.

Clock	RMSE r	RMSE i	Latency Min	Latency Max	Interval Min	Interval Max	Throughput (II*period)^-1 samples per sec	BRAM	DSP	FF	LUT	URAM
1.00E-08	0.001853297697	0.001314980211	1048596	1048596	1048597	1048597	9.54E+01	4	20	2181	3308	0

Q8 DFT1024 Best Design: Now that you have explored different optimizations for DFT32, we can go ahead and try their ideas on DFT1024.

(a) Try any optimization techniques and describe your methodology.

Initially I tried pipelining the outer loop, but because synthesis time was crazy long, I decided to only pipeline the inner loop.
I used the best outcome of the array partitioning.
I used the best outcome for loop unrolling.

During testing, I found the final pipeline value achievable for the inner loop was II = 16.
Using this, I increased loop unrolling and set pipeline to 16 for reduced synthesis time and higher throughput.

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

BRAM	DSP	FF	LUT	URAM
64	20	21435	44361	0

Q9 Streaming Interface Synthesis: Modify your design to allow for streaming inputs and outputs using hls::stream.
You must write your own testbench to account for the function interface change from DTYPE to proper hls::stream.
You can learn about hls::stream from the HLS Stream Library. You should also follow the Lab: Axistream Multiple DMAs example.
Report the latency, throughput, and resource utilization of your design. Resource utilization must be under 100%.
Using your optmized DFT1024 is optional, you can also just convert the baseline DFT1024 to streaming interface.

Using baseline

Clock	RMSE r	RMSE i	Latency Min	Latency Max	Interval Min	Interval Max	Throughput (II*period)^-1 samples per sec	BRAM	DSP	FF	LUT	URAM
1.00E-08	0.001853297697	0.001314980211	6293532	6293532	6293533	6293533	1.59E+01	12	5	1227	1788	0