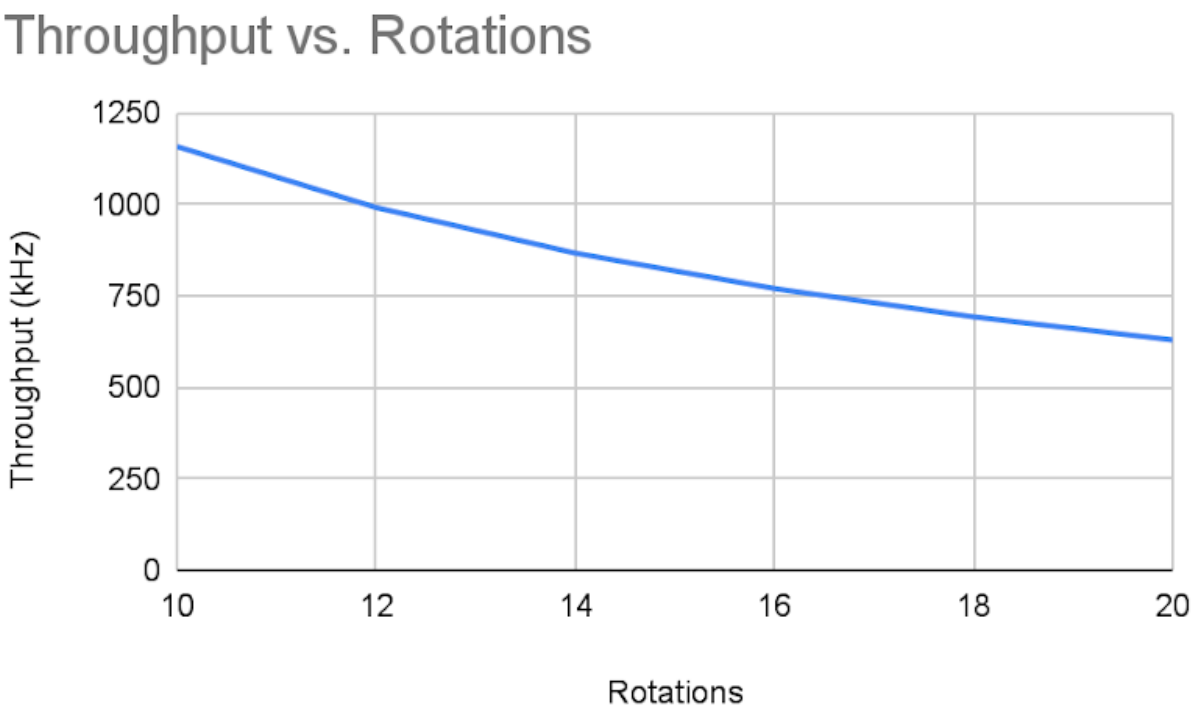
Project 2
Question 1:

1.A.)

Rotations	Resource Usage (BRAM|DSP|FF|LUT)	Clock Period [ns]	Interval (min, max) [cycles]	Throughput	Latency (min, max) [cycles]	RMSE (R, theta)
10	0| 21| 1893| 2921	7.256	4, 119	1.158 MHz	3, 118	0.000001100461873, 0.001291386433877
12	0| 21| 1893| 2921	7.256	4, 139	991.4 kHz	3, 138	0.000000049174467, 0.000282649387373
14	0| 21| 1893| 2921	7.256	4, 159	866.7 kHz	3, 158	0.000000084866343, 0.000078061922977
16	0| 21| 1894| 2922	7.256	4, 179	769.9 kHz	3, 178	0.000000084866343, 0.000016766502085
18	0| 21| 1894| 2921	7.256	4, 199	692.5 kHz	3, 198	0.000000084866343, 0.000005152625818

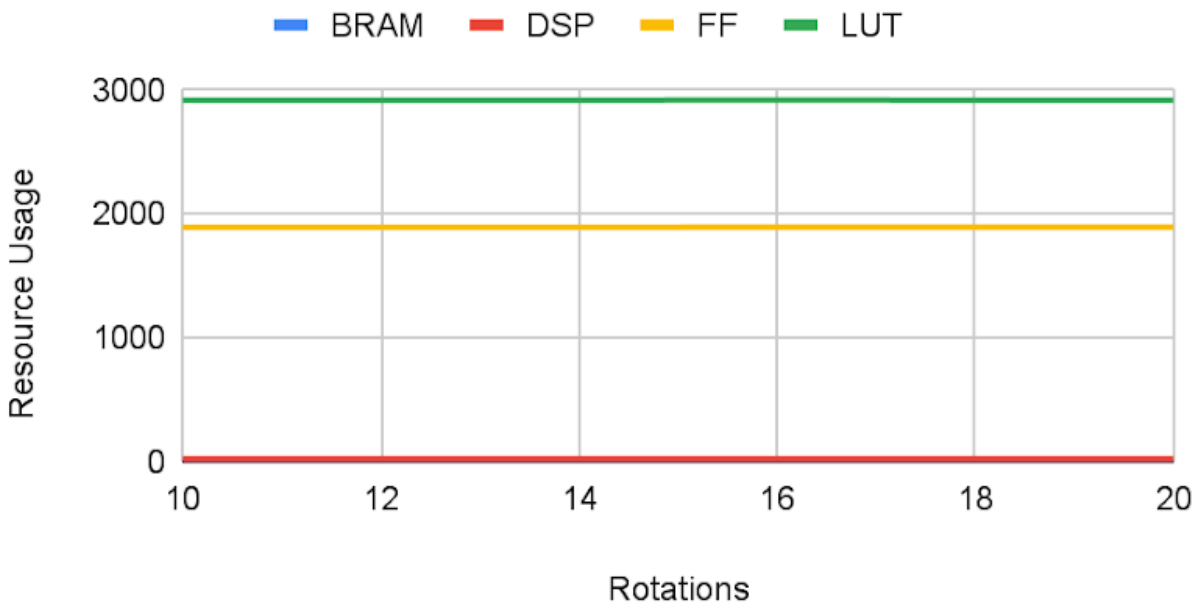
20	0| 21| 1894| 2921	7.256	4, 219	629.3 kHz	3, 218	0.000000084866343, 0.000000784642907

1.B.)

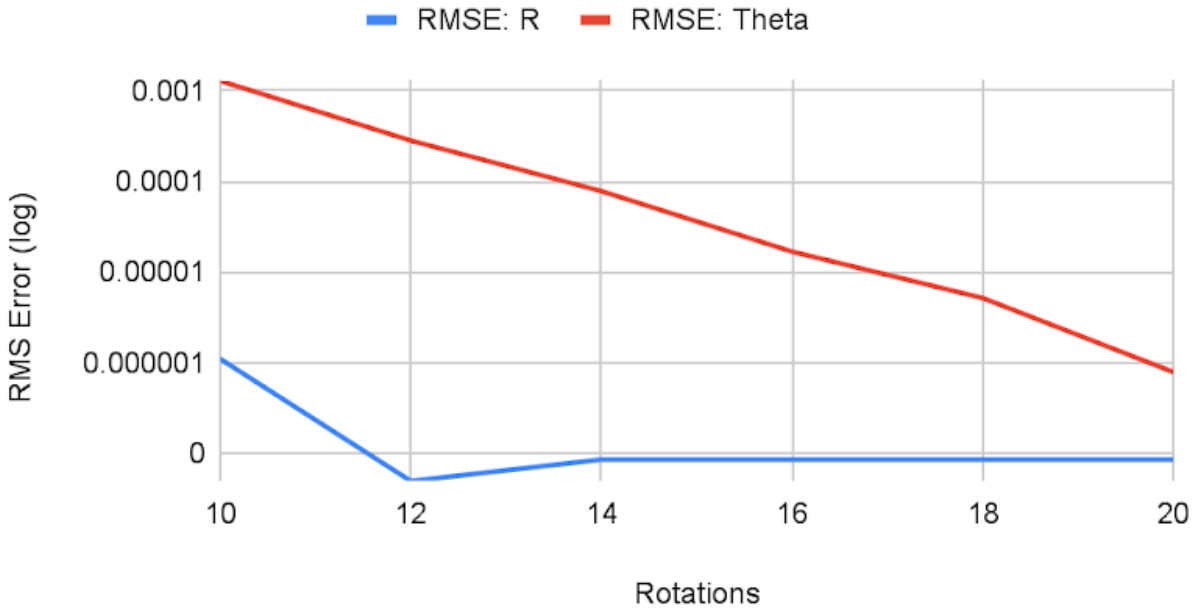


1.B.CONTINUED)

Resource Usage vs. Rotations



RMSE vs. Rotations



1.C.) The RMSE stops noticeably changing after 14 rotations.

Question 2:

2.A.)

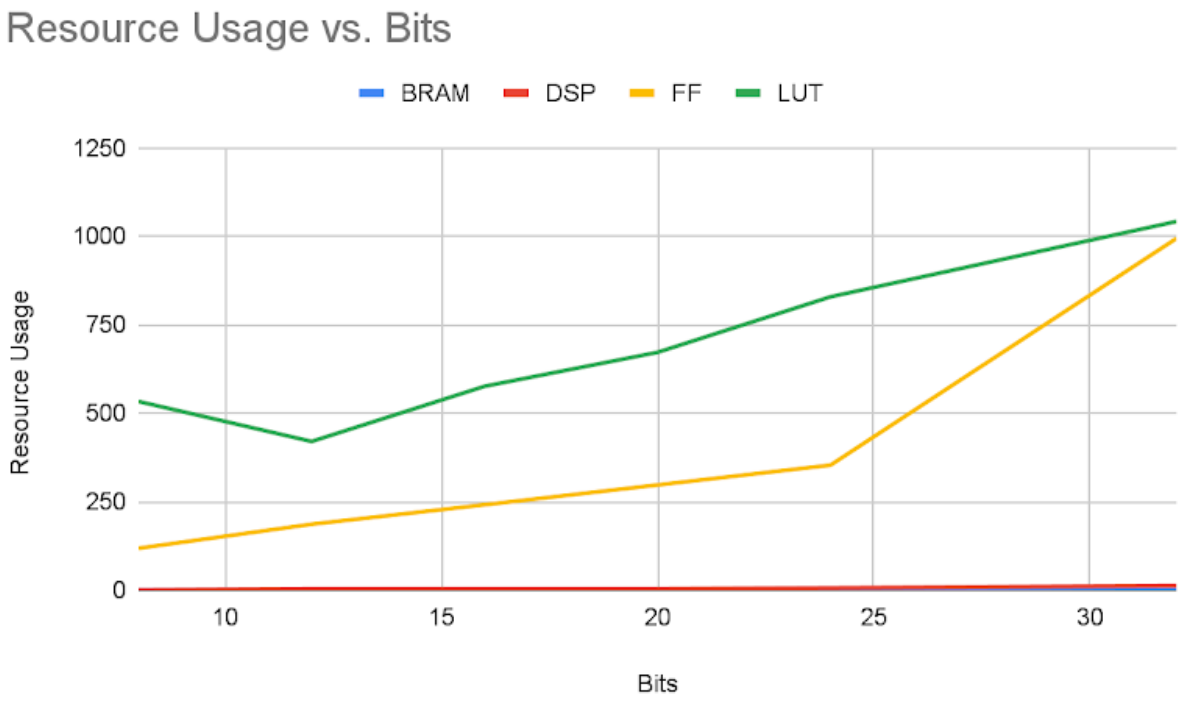
Variable	Range	Number of Bits
x	[-1.65, 1.65]	2
y	[-1.65, 1.65]	2
r	[0, 1.41]	2
theta	[-3.14, 3.14]	3

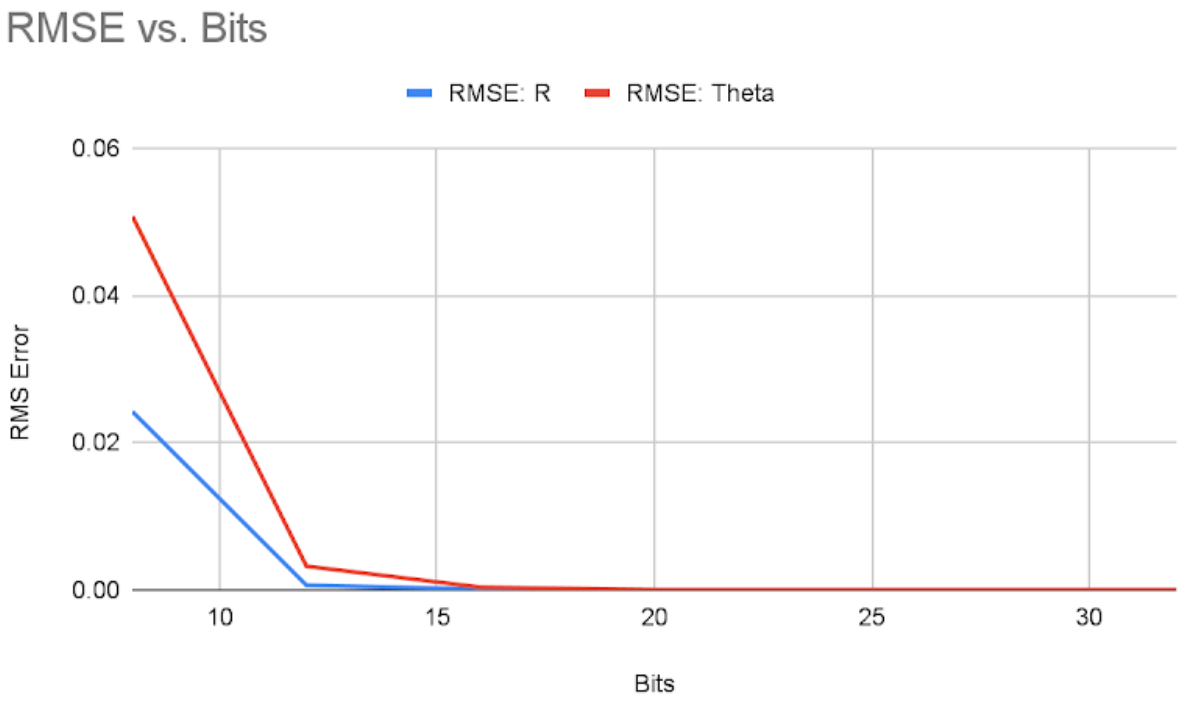
2.B.)

Bits	Resource Usage (BRAM|DSP|FF|LUT)	Clock Period [ns]	Interval (min, max) [cycles]	Throughput	Latency (min, max) [cycles]	RMSE (R, theta)
8	0| 0| 118| 533	7.073	2, 38	3.72 MHz	2, 37	0.024207258597016, 0.050719358026981
12	0| 3| 186| 420	7.167	2, 38	3.672 MHz	1, 37	0.000641623861156, 0.003221699735150

16	0 | 3| 241| 576	6.792	2,38	3.875 MHz	1, 37	0.000110776971269, 0.000376321928343
20	0| 3| 297| 672	6.814	2,38	3.862 MHz	1, 37	0.000004652670214, 0.000024490960641
24	0| 6| 353| 829	6.918	2, 38	3.804 MHz	1, 37	0.000000132 34978, 0.000016160758605
32	0| 12| 994| 1042	7.160	2,55	2.539 MHz	1,54	0.000000028665912, 0.000016693516955

2.B. CONTINUED)

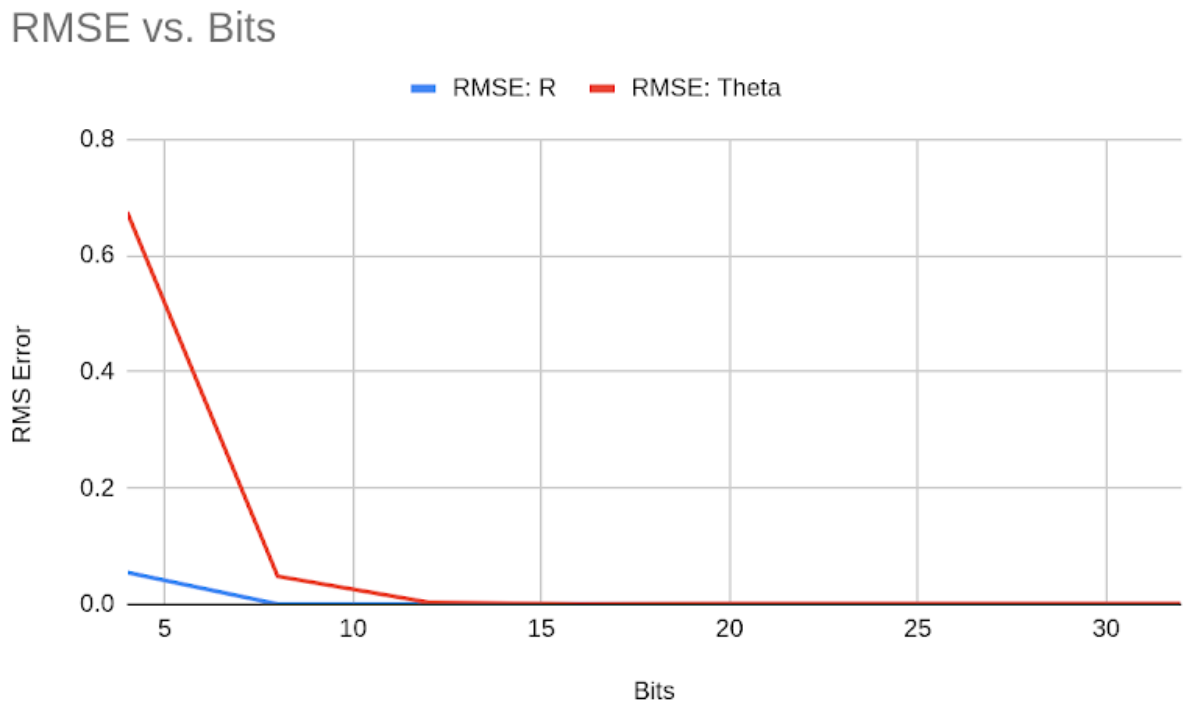




2.C.)

Bits	Resource Usage (BRAM|DSP|FF|LUT)	Clock Period [ns]	Interval (min, max) [cycles]	Throughput	Latency (min, max) [cycles]	RMSE (R, theta)
4	0| 1| 203| 568	6.792	2, 38	3.875 MHz	1, 37	0.055077131837606, 0.675097644329071
8	0| 3| 215| 570	6.792	2, 38	3.875 MHz	1, 37	0.000390634115320, 0.048388864845037

12	0| 3| 227| 573	6.792	2, 38	3.875 MHz	1, 37	0.000139058713103, 0.003590888110921
16	0| 3| 239| 576	6.792	2, 38	3.875 MHz	1, 37	0.000115331131383, 0.000483148935018
20	0| 3| 251| 586| 0|	6.911	2, 38	3.808 MHz	1, 37	0.000115331131383, 0.001135226106271
32	0| 5| 524| 715	7.267	2, 54	2.548 MHz	1, 53	0.000115331131383, 0.001135226106271



Question 3:

Since we used simpler operations instead of multiplication, our resource utilization decreased. Although the HLS tool optimizes computations by converting multiplications into shift operations, explicitly implementing this in the code further improved resource utilization efficiency.

3.A.)

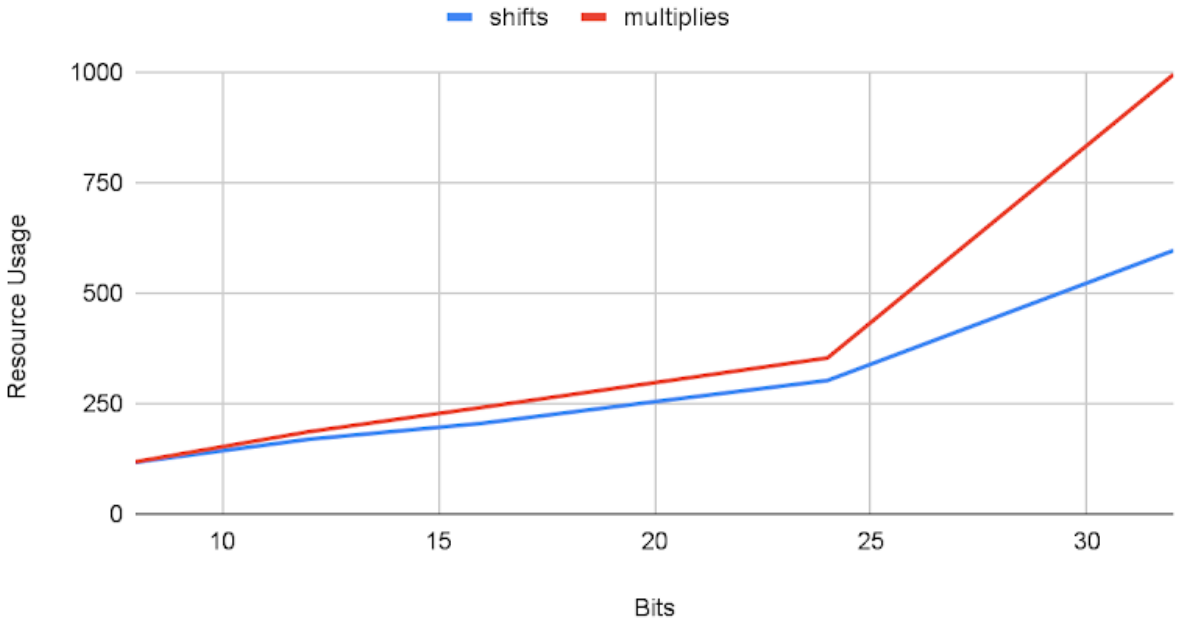
Bits	Resource Usage (BRAM|DSP|FF|LUT)	Clock Period [ns]	Interval (min, max) [cycles]	Throughput	Latency (min, max) [cycles]	RMSE (R, theta)
8	0| 0| 117| 474	6.981	2, 38	3.769 MHz	1, 37	0.024207258597016, 0.086958505213261
12	0| 1| 169| 464	6.287	2, 38	4.185 MHz	1, 37	0.001630143844523, 0.007799166254699
16	0| 1| 205| 620	6.792	2, 37	3.979 MHz	1,36	0.000110776956717, 0.000376321928343
20	0| 1| 254| 737	6.843	2, 37	3.949 MHz	1, 36	0.000110776956717, 0.000691107183229

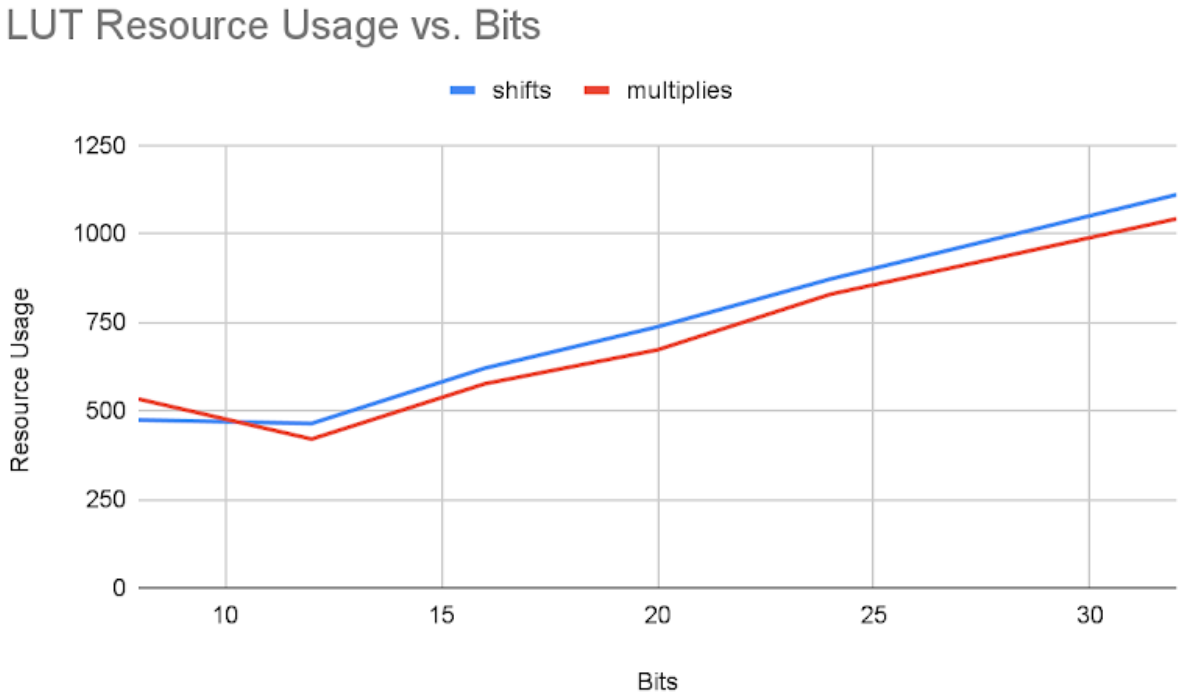
24	0| 2| 302| 872|	6.948	2, 37	3.889 MHz	1, 36	0.000000132349783, 0.000015608316971
32	0| 4| 596| 1110	7.201	2, 38	3.654 MHz	1, 37	0.000000028665912, 0.000016690997654

3.B.)

DSP Resource Usage vs. Bits

FF Resource Usage vs. Bits





Question 4:

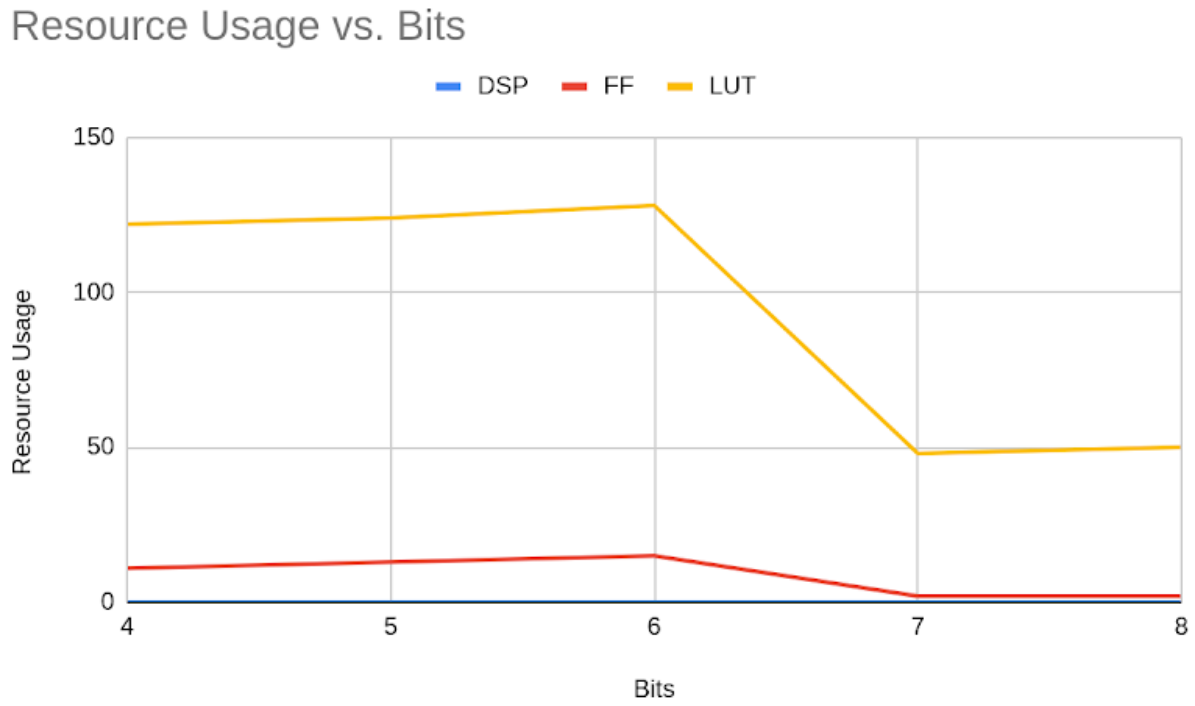
4.A.) The LUT size grows exponentially as the input size increases because of ap_uint<2*W> index. The number of output bits directly multiplies the total LUT memory but does not affect the number of entries. Small increases in input width W drastically increase LUT size, while increases in output width B scale LUT size linearly.

4.B.)

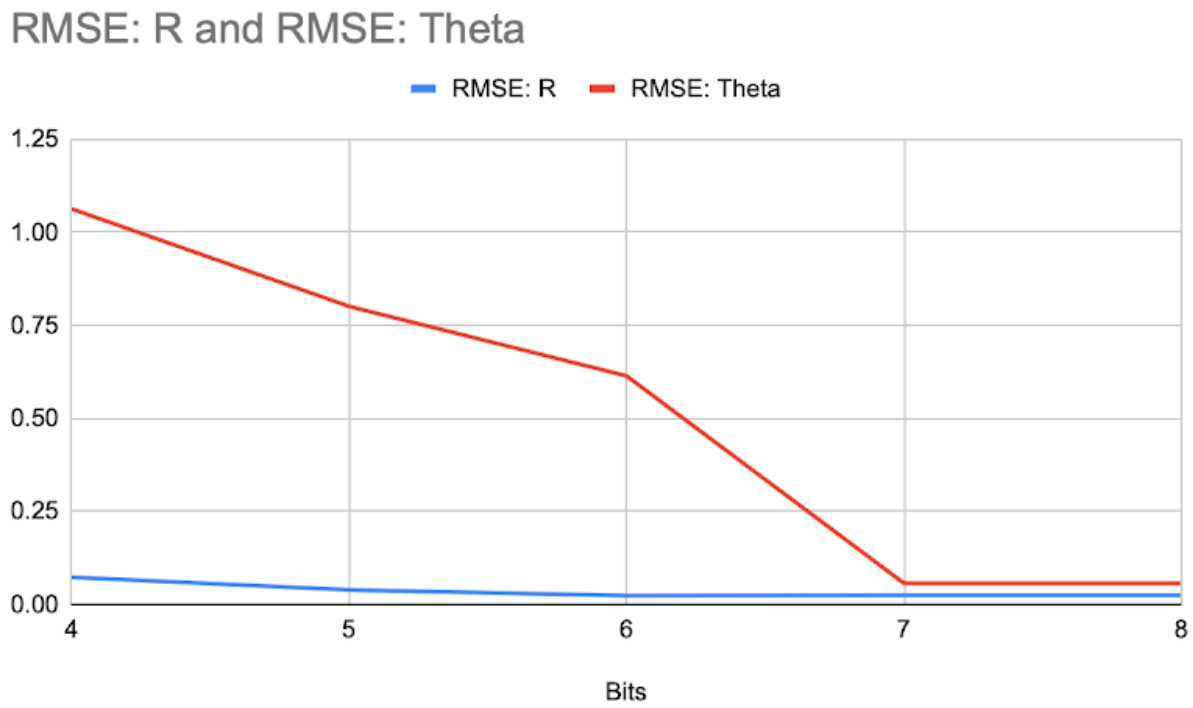
Bits	Resource Usage (BRAM|DSP|FF|LUT)	Clock Period (ns)	Interval (min, max)	Throughput	Latency (min, max) (cycles)	RMSE (R, theta)
4	2| 0| 11| 122	5.708	3, 3	58.39 MHz	2, 2	0.072758205235004, 1.063029170036316

5	2| 0| 13| 124	5.944	3, 3	56.07 MHz	2, 2	0.038790833204985, 0.801008105278015
6	4| 0| 15| 128	5.962	3, 3	55.90 MHz	2, 2	0.023380080237985, 0.614072918891907
7	15| 0| 2| 48	5.225	2, 2	95.69 MHz	1, 1	0.024072423577309 0.056591678410769
8	60| 0| 2| 50	5.480	2, 2	91.24 MHz	1, 1	0.024072423577309 0.056591678410769

4.C.)



4.D.)



4.E.) The advantages of using CORDIC is that it is much faster to synthesise while the disadvantages are that it uses a lot more resources and slower when compared to the LUT based approach. Moreover, LUT is not scalable.

Demo Recording:

https://drive.google.com/file/d/1LolDhcrNwJ3WK3jYkOgOT2e7zr9PqZt5/view?usp=sharing

(also found in README in https://github.com/JDRadatti/cse237c/tree/main)

Testing

We added the following tests to our testbench:

- run_test(0, 0.7003, 1.570796327, 0.7003);
- Tests starting from y-axis (positive)

- run_test(0, - 0.7003, -1.570796327, 0.7003);
- Tests starting from y-axis (negative)
- run_test(-0.485, 0, 3.14159265358979323846, 0.485);
- Tests starting from the x-axis (negative)