CSE237C - Project2

Question 1 - Number of Rotations: Answered with reference to baseline design. This design has the Cordic implementation with float variables to perform the coordinate rotation. Table I summarizes the root mean square error(RMSE) for both polar coordinates r and θ by varying the number of iterations(N). We observe that the RMSE for r remains the same regarding number of iterations because cordic gain settles beyond 6 iterations and varies very little for $N > 6$. On the other hand, RMSE for $theta$ stabilises decreases until $N = 14$ and for $N > 14$, the error stabilizes.

It can also be seen that the latency increases with the number of iterations and also the throughput is less than 1 MHz for the floating point Cordic implementation. As outlined in table II The floating point cordic uses 21 DSPs which is more than the other variations.

Baseline design	**RMSE_r**	**RMSE_theta**	**II (# of cycles)**	**Throughput (MHz)**
N = 10	3.93E-04	1.23E-03	164	0.83
N = 11	3.92E-04	5.20E-04	179	0.76
N = 12	3.92E-04	6.70E-04	194	0.70
N = 14	3.92E-04	4.56E-04	224	0.61
N = 16	3.92E-04	4.83E-04	254	0.54
N = 18	3.92E-04	4.87E-04	284	0.48
N = 20	3.92E-04	4.87E-04	314	0.43

TABLE I: Performance and Accuracy table for different number of iterations in baseline implementation

	BRAM_18k	**DSP**	**FF**	**LUT**
Baseline	0	21	1654	2798
optimized1	0	11	2145	5736
optimized2	0	11	1847	4769
optimized3	0	11	3462	5904
LUT_based	8	0	3	104

TABLE II: Resource usage table for different Cordic implementations

Question 2 - Fixed point implementation: Answered with reference to optimized1,2 and 3 designs. All these designs are modified to do the cordic rotation using fixed point datatypes. Considering the input range and output range, we can design the datatypes for these variables to decrease the computational load. Assuming the input x,y coordinates are normalized to [-1,1], this implies $r < \sqrt{2}$ and $-\pi < \theta < \pi$. So, to store θ without any overflow errors, atleast 3 integer bits must be allocated to the angle accumulator. Also, since $r = 1.414$ for the worst case when (x,y) = (-1,-1) and considering a cordic gain of 1.647. We must allocate integer bits to store both positive and negative numbers with magnitude upto $1.414 * 1.647 = 2.32$. This means atleast 3 integer bits are needed to store the coordinates in the process of rotation.

The number of fractional bits can be adjusted based on how much precision is needed. For example, if we need precision to the third decimal point, we need approximately 10 fractional bits. Optimized1 design uses 27 fractional bits and compares the RMSE errors for different number of cordic iterations in table III. The RMSE of θ stabilize beyond $N = 14$.

Optimized1 design	**RMSE_r**	**RMSE_theta**	**II (# of cycles)**	**Throughput(MHz)**
N = 11	3.92E-04	5.28E-04	38	3.6
N = 12	3.92E-04	6.73E-04	39	3.5
N = 14	3.92E-04	4.56E-04	41	3.3
N = 16	3.92E-04	4.83E-04	43	3.2
N = 18	3.92E-04	4.87E-04	45	3.07
N = 20	3.92E-04	4.87E-04	47	2.94

TABLE III: Performance and Accuracy table for different number of iterations with fixed-point implementation

Optimized 2 design uses lesser fractional bits 14 fractional bits and limit the number of iterations to 14 to further reduce the resource usage and it is tabulated in table II. In this case, the number of FFs and LUTs used decreased by 20% compared to optimized1 design. The RMSE of θ however increased slightly to 5.1e-4 and achieves throughput of 3.3MHz.

Optimized3 design enables function level pipelining to reduce the initiation interval(II) to 1 and achieves 138MHz throughput at the expense of slightly higher FF and LUT usage. Accuracy comparison of optimized 2 and 3 designs is shown in tableIV.

14 iterations	**RMSE_r**	**RMSE_theta**	**II (# of cycles)**	**Throughput(MHz)**
optimized2	2.8E-04	5.1E-04	40	3.3
optimized3	2.8E-04	5.1E-04	1	138

TABLE IV: Comparison of optimized 2 and 3 designs

Question 3: Answered with reference to optimized1,2,3 designs.

Using adds and shifts instead of multiply and divide operations resulted in less usage of DSPs because of the reduced computational load as can be seen from tableII. But the LUTs and FFs usage remain higher compared to the baseline floating point implementation. In terms of accuracy, fixed-point implementation error in optimized1 design is close to error in floating-point implementation of baseline design. But as the fractional bits are reduced to 14 in optimized2 design, the error is slightly higher than that of baseline.

Question4 -LUT design: Answered with respect to LUT based cordic design. The default code that is provided has a bit width of 8 for x,y coordinates as well as the r, θ coordinates. It doesn't use any DSPs since it is designed to just index into a table of values and do not need computations.

Fractional bitwidth	**RMSE_r**	**RMSE_theta**	**II (# of cycles)**	**Throughput(MHz)**
8	2.3E-02	5.1E-02	3	51.3
10	3E-3	5.1E-03	3	51.3

TABLE V: Performance and Accuracy for LUT based design

a) The size of LUT is equal to 2^{W_i} where W_i is the bitwidth of the x,y coordinates. Memory occupied by the Lookup table is equal to $2W_j * 2^{2W_i}$ bits where W_j is the bitwidth of the r, θ coordinates. The exponent term dominates the total memory and increasing the bitwidth of input increases LUT size exponentially whereas it changes linearly with the output bitwidth.
b) For x,y $\in$ [-1,1], we need at least 2 integer bits for fixed-point representation. Consequently, $r \in [0, \sqrt{2}]$ and $\theta \in [-\pi, \pi)$. To represent r, we need at least 2 integer bits because $r > 0$. But for θ, we need 3 integer bits because θ can be negative as well.
c) The provided code is modified to increase the fractional bit width to 10. In this case, the RMSE improved by an order of magnitude for both r and θ as presented in TableV.
d) The throughput and latency remained the same even after increasing the bitwidth. This happens because in Lookup table method, there is nothing to compute using DSPs and we are just indexing into a memory array.
e) Increasing the bitwdith of inputs/outputs can significantly increase the memory usage. As can be seen in Table VI, the BRAM usage went up by 16 times just by increasing fractional bits by 2 more bits. The advantage here is that, the latency and the initiation interval can be very small because of little to no computational load on the DSPs.

Bitwidth	**BRAM_18k**	**DSP**	**FF**	**LUT**
8	8	0	3	104
10	128	0	3	88

TABLE VI: Resource usage table for LUT implementation