CSE 237C Project 2

Ze Han, Zhijun Wang, Ziyi Zeng

October 23, 2025

1 Q1

1.1 (a)

Summarized in Table 1 and Table 2.

Table 1: Performance summary

Design	Latency (ns)	Throughput (MHz)	RMSE(R)	RMSE(Theta)
CORDIC (NO_ITER=10)	1610	0.62	0.000001239063636	0.001373776700348
CORDIC (NO_ITER=12)	1910	0.52	0.000000064680641	0.000274967635050
CORDIC (NO_ITER=14)	2210	0.45	0.000000129140460	0.000081084443082
CORDIC (NO_ITER=16)	2510	0.40	0.000000129140460	0.000016661457266
CORDIC (NO_ITER=18)	2810	0.36	0.000000129140460	0.000004772118245
CORDIC (NO_ITER=20)	3110	0.32	0.000000129140460	0.000000616681575

Table 2: Utilization summary

Design	BRAM	DSP	FF	LUT
CORDIC (NO_ITER=10)	0	10	1347	2246
CORDIC (NO_ITER=12)	0	10	1347	2246
CORDIC (NO_ITER=14)	0	10	1347	2246
CORDIC (NO_ITER=16)	0	10	1347	2246
CORDIC (NO_ITER=18)	0	10	1348	2246
CORDIC (NO_ITER=20)	0	10	1348	2246

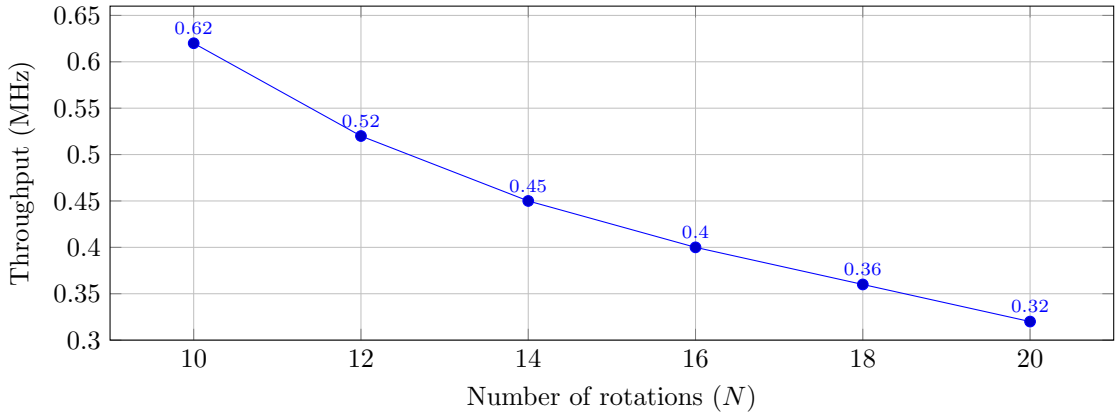

Figure 1: Throughput vs. number of rotations

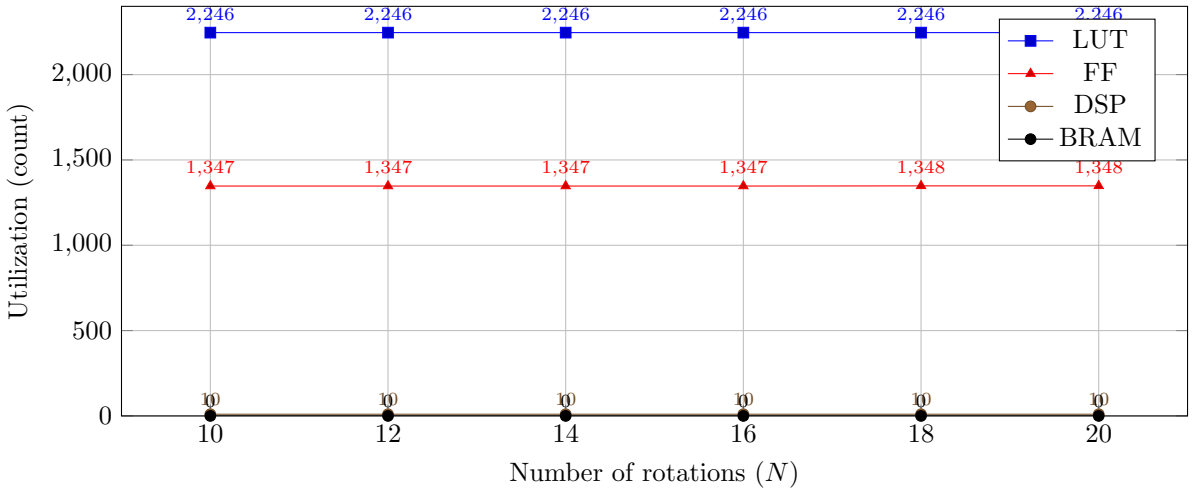

Figure 2: Resource usage vs. number of rotations

1.2 (b)

Summarized in Figure 1, Figure 2, and Figure 3.

1.3 (c)

14 rotations.

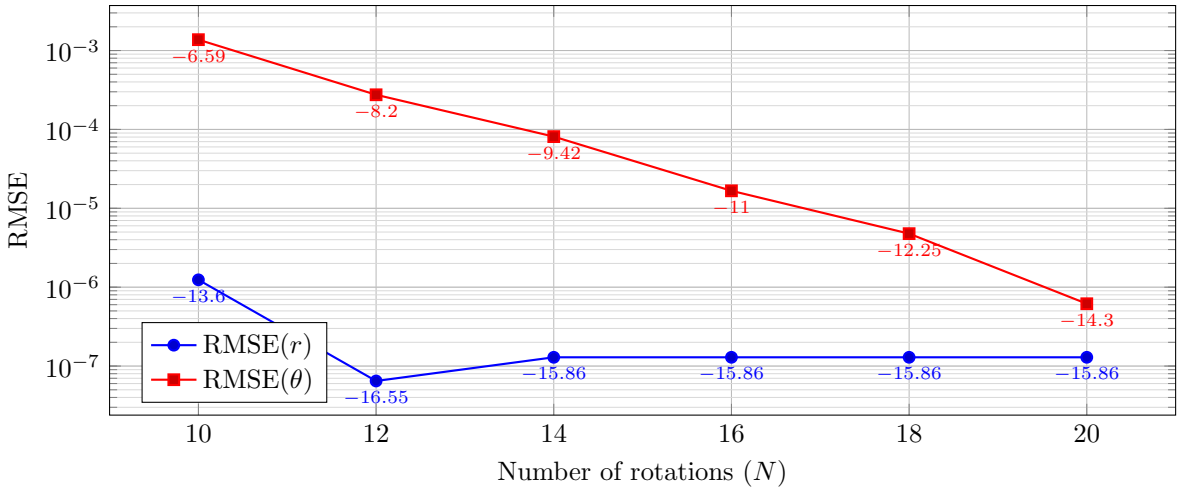

Figure 3: RMSE vs. number of rotations

2 Q2

2.1 Q2(a) Integer bits for signed `ap_fixed`

We use signed `ap_fixed`$\langle W, I\rangle$, where the integer-bit count I *includes* the sign bit. The representable range is $[-2^{I-1}, 2^{I-1})$. The testbench normalizes $x, y \in [-1, 1]$, with $r = \sqrt{x^2 + y^2}$ and $\theta = \text{atan2}(y, x)$. Choose the minimum I such that $2^{I-1} > \max|v|$ for each variable:

- x: range $[-1, 1] \Rightarrow I = 2$.
- y: range $[-1, 1] \Rightarrow I = 2$.
- r: range $[0, \sqrt{2}] \approx [0, 1.414] \Rightarrow I = 2$.
- θ: range $(-\pi, \pi] \approx (-3.142, 3.142] \Rightarrow I = 3$.

Answer: $I_x = I_y = I_r = 2,\ I_\theta = 3$.

2.2 (b)

Summarized in Table 3 and Table 4.

Table 3: Q2(b) CORDIC (multiply version): resource, throughput and latency (clock = 10 ns)

W	LUT	FF	DSP	BRAM	II	thr/clk	thr@100MHz	Lat(cyc)
8	2075	603	0	0	27	0.037037	3.7037	26
12	2023	663	3	0	28	0.035714	3.5714	27
16	2262	757	3	0	28	0.035714	3.5714	27
20	2440	834	3	0	28	0.035714	3.5714	27
24	2776	1056	6	0	28	0.035714	3.5714	27
32	4248	1982	12	0	32	0.031250	3.1250	31

2.3 Q2(c)

Table 5 and Fig. 4 summarize the results when all variables use `ap_fixed<16,3>` and only the CORDIC tables (K and angles) vary in total bits.

Table 4: Q2(b) CORDIC (multiply version): RMSE vs total bits

W	RMSE(r)	RMSE(θ)
8	0.0970888883	1.1478300095
12	0.0534359328	1.3125388622
16	0.0522289313	1.3237951994
20	0.0522061139	1.3238462210
24	0.0522038378	1.3238475323
32	0.0522035807	1.3238475323

Table 5: Q2(c): Only table width varies (W_{TBL}); resources, throughput, latency and RMSE (clock = 10 ns)

W_{TBL}	LUT	FF	DSP	BRAM	II	thr/clk	thr@100 MHz	Lat(cyc)	RMSE(r) / RMSE(θ)
4	2172	695	0	0	27	0.037037	3.7037	26	0.507104 / 1.168963
8	2200	709	0	0	27	0.037037	3.7037	26	0.036810 / 0.405302
12	2176	726	1	0	28	0.035714	3.5714	27	0.049163 / 0.289276
16	2176	726	1	0	28	0.035714	3.5714	27	0.051283 / 0.278630
20	2176	726	1	0	28	0.035714	3.5714	27	0.051407 / 0.278630
32	2176	726	1	0	28	0.035714	3.5714	27	0.051407 / 0.278630

3 Q3

3.1 (a)

Summarized in Table 6 and Table 7.

Table 6: Q3(a) Resources, throughput, and latency (clock = 10 ns)

W	LUT	FF	DSP	BRAM	II	Throughput (s/clk)	Throughput (MSPS)	Latency (cycles)
8	1942	596	0	0	27	0.037037	3.7037	26
12	1998	652	1	0	28	0.035714	3.5714	27
16	2255	742	1	0	28	0.035714	3.5714	27
20	2445	818	1	0	28	0.035714	3.5714	27
24	2751	1040	2	0	28	0.035714	3.5714	27
32	4261	1578	4	0	32	0.031250	3.1250	31

3.2 (b)

Summary. Results are shown in Figs. 5, 6, and 7.

Figures 5–7 compare the baseline Q2(b) *Multiply* implementation with the Q3(a) *Add+Shift* implementation over total bit-widths $W \in \{8, 12, 16, 20, 24, 32\}$.

From Fig. 5, LUT usage grows similarly with W for both implementations and the gap is small. Fig. 6 shows that *Add+Shift* uses significantly fewer DSPs at all W (often down to about one third to one quarter at large W), because CORDIC replaces general multipliers with additions and shifts. Fig. 7 indicates that *Add+Shift* also uses slightly fewer FFs, with a more visible gap at $W = 32$. Throughput and cycle latency remain the same, since both designs use the same II and pipeline depth. Overall, the *Add+Shift* CORDIC substantially reduces DSP usage with nearly unchanged LUTs and similar performance compared to the *Multiply* baseline.

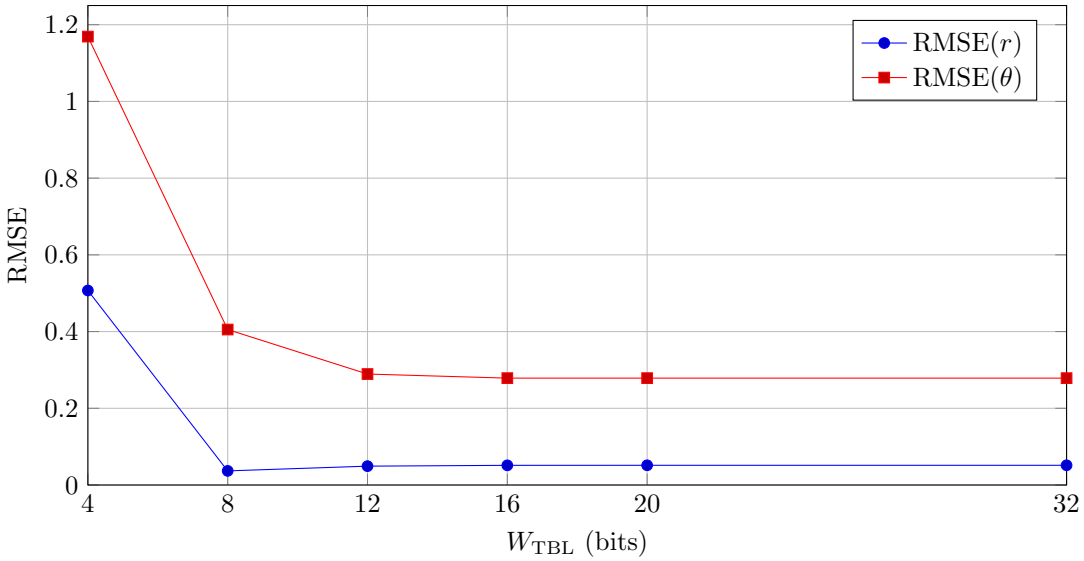

Figure 4: Q2(c): RMSE vs. table width W_{TBL} with all variables fixed at `ap_fixed<16,3>`.

Table 7: Q3(a) RMSE of r and θ

W	RMSE(r)	RMSE(θ)
8	1.0971163511	0.0247685537
12	0.0060032504	0.0030231269
16	0.0002107439	0.0005414842
20	0.0000825915	0.0003237652
24	0.0000682161	0.0003230932
32	0.0000679273	0.0003230377

4 Q4

4.1 (a)

Setup. Assume the LUT implementation quantizes inputs with `ap_fixed`$\langle W, I\rangle$ for both x and y, and stores outputs r and θ with types `ap_fixed`$\langle T_r, I_r\rangle$ and `ap_fixed`$\langle T_\theta, I_\theta\rangle$. (If both outputs share the same type `data_t` = `ap_fixed`$\langle T_{\mathrm{data}}, I_{\mathrm{data}}\rangle$, then $T_r = T_\theta = T_{\mathrm{data}}$.)

Input type ⇒ number of entries. The address is formed by concatenating the W-bit codes of x and y, so the address width is $2W$ and the number of LUT entries is

$$\#\text{entries} \;=\; 2^{\,2W}. \tag{1}$$

Thus, increasing the input total-bit width W grows the LUT *exponentially* (doubling W squares the entries).

Output type ⇒ bits per entry. Each entry stores both r and θ, whose total payload width is

$$\text{bits/entry} \;=\; T_r + T_\theta \;=\; \begin{cases} 2\,T_{\mathrm{data}}, & \text{if } r,\theta \text{ share } \texttt{data\_t}, \\ T_r + T_\theta, & \text{(general case)}. \end{cases} \tag{2}$$

Hence, increasing the output bit widths scales the memory *linearly* in bits per entry.

Total memory bits. Combining the two,

$$\text{Total LUT bits} \;=\; \left(2^{\,2W}\right) \times \left(T_r + T_\theta\right) \;=\; \begin{cases} 2^{\,2W} \cdot (2T_{\mathrm{data}}), & r,\theta \sim \texttt{data\_t}, \\ 2^{\,2W} \cdot (T_r + T_\theta), & \text{general}. \end{cases} \tag{3}$$

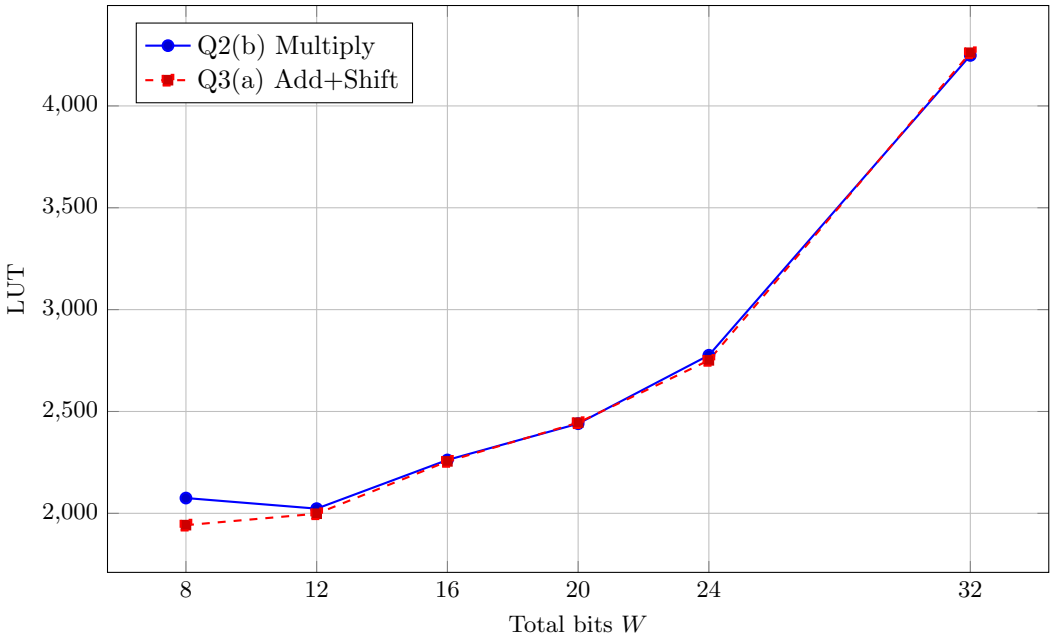

Figure 5: Q3(b): LUT vs total bits W (Multiply vs Add+Shift).

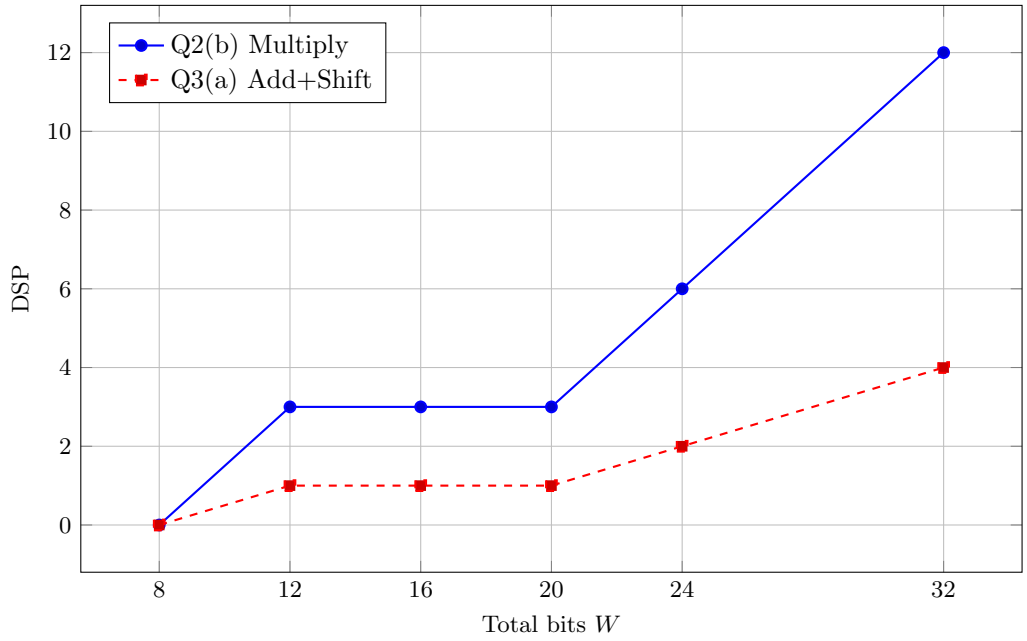

Figure 6: Q3(b): DSP vs total bits W (Multiply vs Add+Shift).

Practical corollaries. For Xilinx BRAM18 ($\approx$ 18,432 bits each),

$$\#\text{BRAM18} \;\approx\; \left\lceil \frac{2^{\,2W}\,(T_r + T_\theta)}{18{,}432} \right\rceil. \tag{4}$$

With a synchronous read, the LUT has $\approx$1-cycle latency and 1 result/cycle throughput. The dominant scaling term is the $2^{\,2W}$ factor from the *input* precision, while the *output* precision contributes linearly through $T_r + T_\theta$.

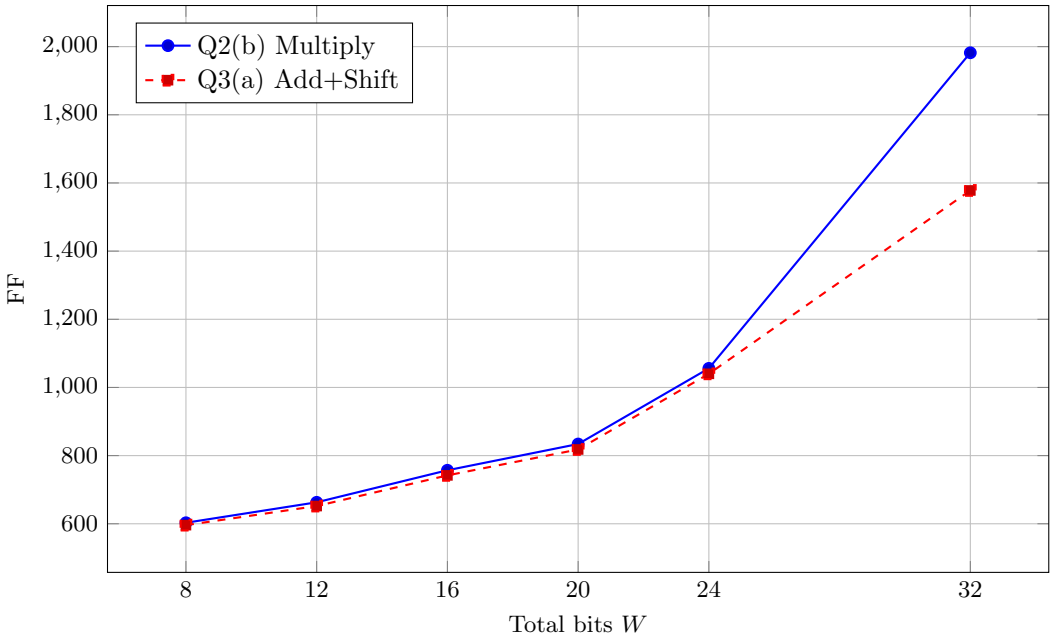

Figure 7: Q3(b): FF vs total bits W (Multiply vs Add+Shift).

4.2 (b)

Summarized in Table 8

Table 8: Q4(b) LUT: resource, throughput and latency (clock = 10 ns)

W	LUT	FF	DSP	BRAM	II	thr/clk	thr@100MHz	Lat(cyc)
4	23413	16292	12	1	256	0.003906	0.3906	329
5	23419	16300	12	1	1024	0.000977	0.0977	1097
6	23431	16372	12	3	4096	0.000244	0.0244	4169
7	23455	16380	12	13	16384	0.000061	0.0061	16457
8	23481	16386	12	57	65536	0.000015	0.0015	65609

4.3 (c)

Summarized in Figure 8,9,10

4.4 (d)

Summarized in table Figure 11

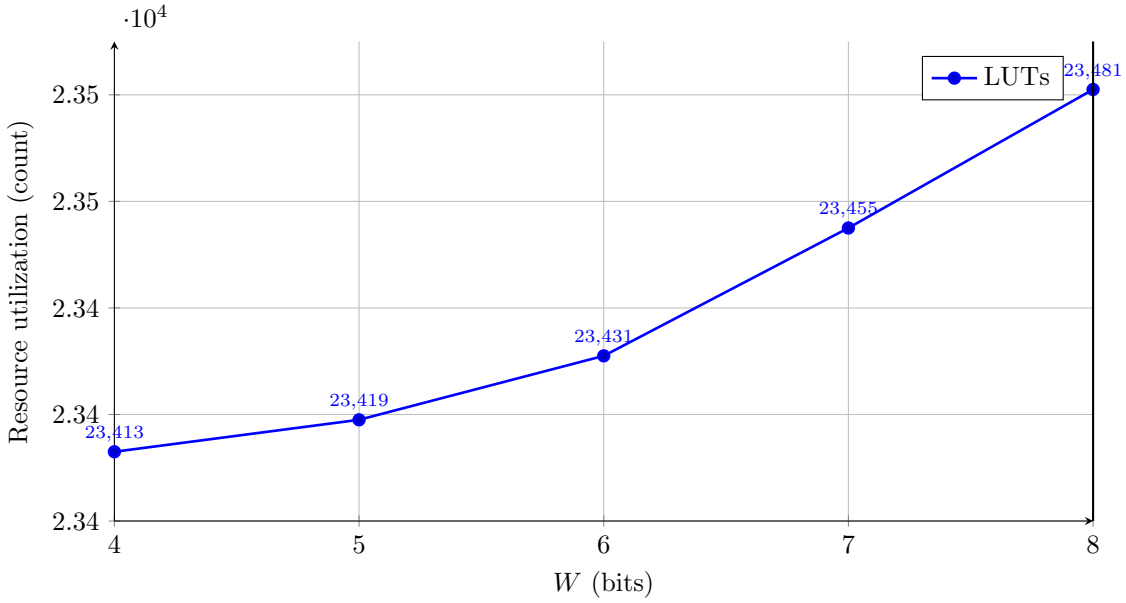

Figure 8: Q4(c): LUT utilization vs. total bits W (values marked; vertical borders on both sides).

4.5 (e)

Note: LUT (Q4) and CORDIC (Q1–Q3) ran on different machines; we compare *relative* on-chip trends only.

- **Accuracy:** CORDIC attains much lower error (e.g., θ RMSE $\sim 10^{-4}$–10^{-5} with more iterations), whereas LUT plateaus around RMSE$(r) \approx 2.31 \times 10^{-2}$ and RMSE$(\theta) \approx 5.10 \times 10^{-2}$ for $W \geq 7$.
- **Throughput/II trend:** In our reports, LUT throughput *decreases* sharply with W because the tool scheduled a large-II initialization loop (II$\approx 2^{2W}$), while CORDIC maintains a *nearly constant* II ($\sim$27–32) and similar cycle latency across W.
- **Latency (reported):** LUT shows very large reported latencies when the init loop is included (e.g., 329$\rightarrow$65,609 cycles for W=4$\rightarrow$8), whereas CORDIC is $\sim$26–31 cycles.
- **BRAM:** LUT memory grows *exponentially* with W (BRAM18K: $1, 1, 3, 13, 57$ for W=4..8); CORDIC uses ≈ 0 BRAM.
- **Logic (LUT/FF):** LUT-based design consumes $\sim$23k LUT / $\sim$16k FF with weak W-dependence; CORDIC uses an order of magnitude fewer (few thousand LUT, few hundred–couple thousand FF depending on W).
- **DSPs:** LUT shows a small fixed DSP count (12 in our build due to helper math during init); CORDIC uses few DSPs and scales gently with W (e.g., 0$\rightarrow$12 in the multiply variant).

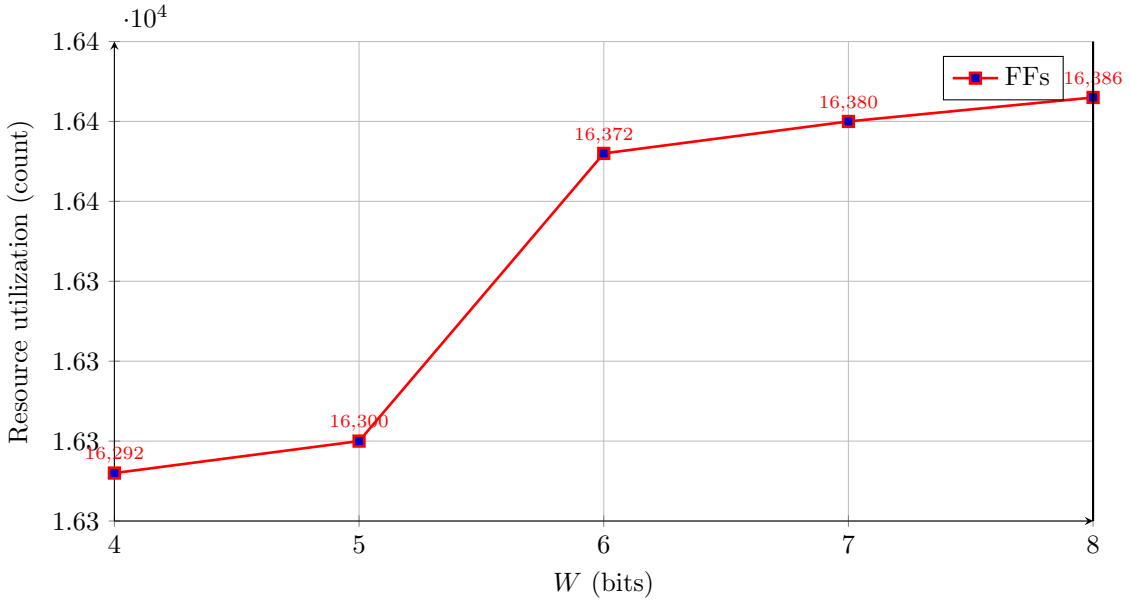

Figure 9: Q4(c): FF utilization vs. total bits W (values marked; vertical borders on both sides).

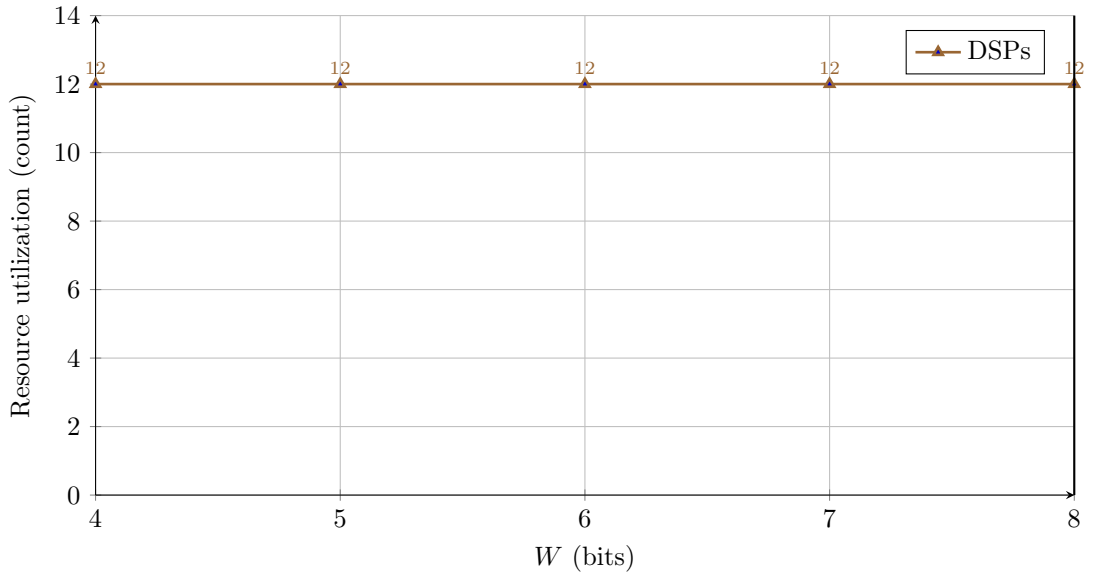

Figure 10: Q4(c): DSP utilization vs. total bits W (values marked; vertical borders on both sides).

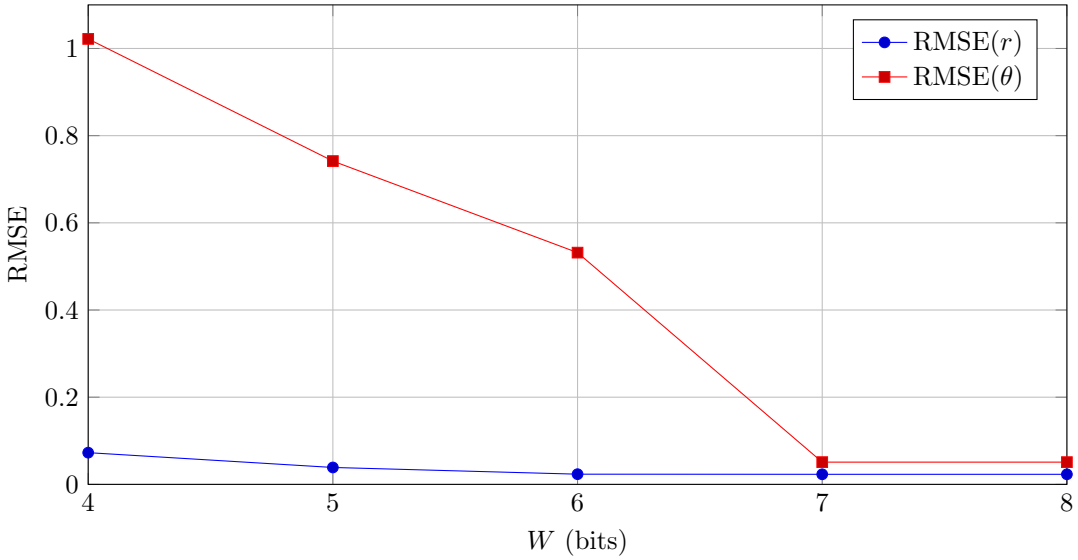

Figure 11: Q4(d): RMSE vs. total bits W for the LUT-based implementation (csim results).