MSFEA
EECE Department
EECE 423: Reconfigurable Computing
Project Report

Professor:

Dr. Mazen Saghir

Student:

Tarek Bshinnati/ID: 202400161

Soheil Haroun/ID: 202405922

December 2nd, 2025

1 Summary

This report describes the design, implementation, and verification of a ZedBoard FPGA-integrated CORDIC angle computation IP block. The accelerometer interface from Project 1 is extended by replacing software-based arctangent calculations with fixed-point arithmetic hardware.

With 13 cycles, the CORDIC IP block calculates tilt angles from ADXL345 accelerometer data in vectoring mode. The circuitry calculates angles with better accuracy than 0.2 degrees compared to floating-point reference implementations in Q3.12 fixed-point format (1 sign bit, 3 integer bits, 12 fractional bits).

2 Overall System Design

The system uses the Processing System (PS) and Programmable Logic of the Zynq SoC architecture.

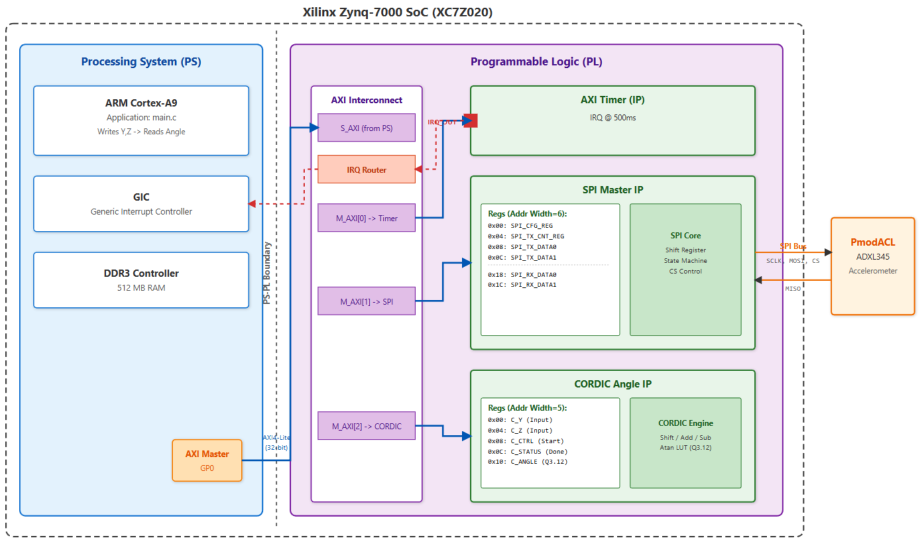


Figure 1: Complete system block diagram showing integration of CORDIC IP with existing SPI IP

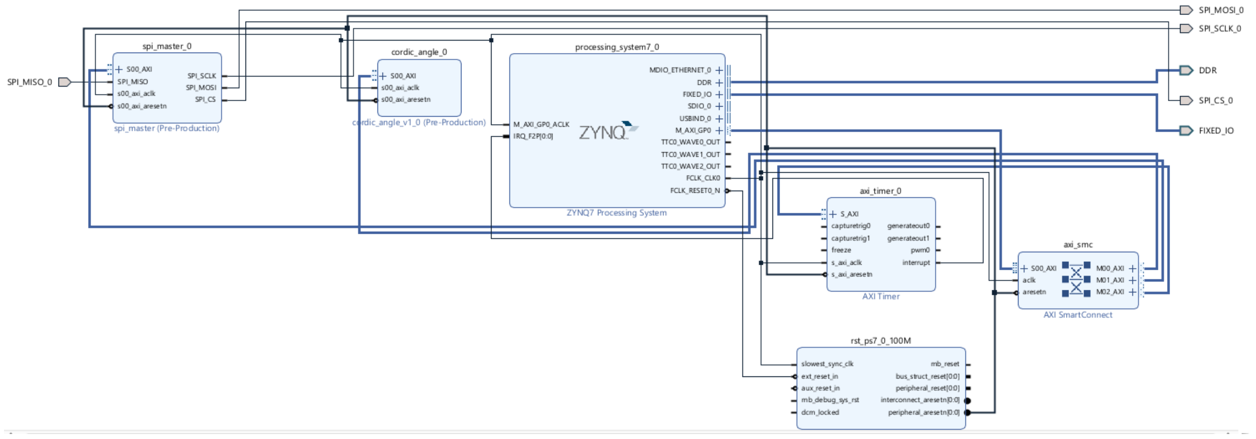


Figure 2: Block Diagram from Vivado

Operation of the system:

1. An AXI Timer generates periodic interrupts every 500 ms
2. The interrupt service routine reads X, Y, Z acceleration data from the ADXL345 via the SPI Master IP
3. The PS writes Y and Z values to the CORDIC IP through its AXI-Lite interface
4. The CORDIC hardware performs 13 iterations to compute the angle
5. The PS reads the result in Q3.12 fixed-point format
6. Software computes atan2f(Y, Z) for comparison and validation
7. Results are displayed

3 Fixed-Point Arithmetic Design

3.1 Numerical Representation

Fixed-point arithmetic provides a compromise between the simplicity of integer arithmetic and the dynamic range of floating-point, making it ideal for hardware implementations where floating-point units are resource-intensive.

3.1.1 Q3.12 Format

The CORDIC angle output uses Q3.12 fixed-point format, defined as follows:

- **Total bits:** 16

- **Sign bit:** 1 (bit 15)
- **Integer bits:** 3 (bits 14-12)
- **Fractional bits:** 12 (bits 11-0)

This representation supports angles in the range $[-4\pi, 4\pi)$ radians with a resolution of $2^{-12} \approx 0.000244$ radians, or approximately 0.014 degrees.

3.2 Conversion Between Formats

3.2.1 Floating-Point to Fixed-Point

To convert a floating-point angle θ to Q3.12 format:

$$\Theta_{\text{fixed}} = \text{round}(\theta \cdot 2^{12}) \tag{1}$$

For example, to represent $\pi/4 = 0.785398$ radians:

$$\Theta_{\text{fixed}} = \text{round}(0.785398 \times 4096) = 3217 \tag{2}$$

3.2.2 Fixed-Point to Floating-Point

To convert a Q3.12 value back to floating-point:

$$\theta \approx \frac{\Theta_{\text{fixed}}}{2^{12}} = \frac{\Theta_{\text{fixed}}}{4096} \tag{3}$$

3.3 Angle Look-Up Table

You need a look-up table (LUT) with arctangent values for each iteration of the CORDIC algorithm. These constants are calculated ahead of time and saved in the Q3.12 format. Table 1 shows the full LUT that was used in the implementation.

Table 1: CORDIC angle look-up table in Q3.12 format

Iteration i	**Formula**	α_i **(radians)**	$A[i]$ **(Q3.12)**
0	$\arctan(2^0)$	0.785398	3217
1	$\arctan(2^{-1})$	0.463648	1899
2	$\arctan(2^{-2})$	0.244979	1003
3	$\arctan(2^{-3})$	0.124355	509
4	$\arctan(2^{-4})$	0.062419	256
5	$\arctan(2^{-5})$	0.031240	128
6	$\arctan(2^{-6})$	0.015624	64
7	$\arctan(2^{-7})$	0.007812	32
8	$\arctan(2^{-8})$	0.003906	16
9	$\arctan(2^{-9})$	0.001953	8
10	$\arctan(2^{-10})$	0.000977	4
11	$\arctan(2^{-11})$	0.000488	2
12	$\arctan(2^{-12})$	0.000244	1

4 CORDIC Algorithm Implementation

The CORDIC algorithm provides an efficient method for computing trigonometric functions using only shift, add, and subtract operations. For this project, the vectoring mode is employed to compute $\arctan(Y/Z)$ directly from the accelerometer readings.

The algorithm rotates (U, V) toward the positive U-axis by smaller angles until the V component converges to zero in vectoring mode. The cumulative rotation angle is $\arctan(V_0/U_0)$.

The Important thing to notice is that multiplications by $\tan(\alpha_i) = 2^{-i}$ can be done without hardware multipliers as simple right shifts.

4.1 Datapath Variables

Three 16-bit signed registers maintain the algorithm state:

- U_i (cU): Horizontal vector component, initialized to Z
- V_i (cV): Vertical vector component, initialized to Y
- Θ_i (cTh): Accumulated angle in Q3.12 format, initialized to 0

The U and V components use Q15.0 format (pure integers) while the angle accumulator uses Q3.12 fixed-point representation.

4.2 Verilog Implementation

The CORDIC datapath is implemented within the AXI-Lite slave module. The complete user logic section is presented below.

Listing 1: CORDIC datapath implementation

```
reg signed [15:0] cU, cV;        // vector x,y
reg signed [15:0] cTh;           // angle acc
reg [3:0] cIter;                 // iter count
reg cBusy, cDone;
reg cStart_prev;
wire cStart_pulse;
reg signed [15:0] shftU, shftV;

// atan LUT in Q3.12
wire signed [15:0] atan_tbl [0:12];
assign atan_tbl[0]  = 16'd3217;
assign atan_tbl[1]  = 16'd1899;
assign atan_tbl[2]  = 16'd1003;
assign atan_tbl[3]  = 16'd509;
assign atan_tbl[4]  = 16'd256;
assign atan_tbl[5]  = 16'd128;
assign atan_tbl[6]  = 16'd64;
assign atan_tbl[7]  = 16'd32;
assign atan_tbl[8]  = 16'd16;
assign atan_tbl[9]  = 16'd8;
assign atan_tbl[10] = 16'd4;
assign atan_tbl[11] = 16'd2;
assign atan_tbl[12] = 16'd1;

// edge detect for start
always @(posedge S_AXI_ACLK) begin
    if (S_AXI_ARESETN == 1'b0)
        cStart_prev <= 1'b0;
    else
        cStart_prev <= slv_reg2[0];
end
assign cStart_pulse = slv_reg2[0] & ~cStart_prev;

// cordic fsm
always @(posedge S_AXI_ACLK) begin
    if (S_AXI_ARESETN == 1'b0) begin
        cU <= 16'd0;  cV <= 16'd0;  cTh <= 16'd0;
        cIter <= 4'd0;  cBusy <= 1'b0;  cDone <= 1'b0;
    end
    else begin
        if (cStart_pulse) begin
            cU    <= slv_reg1[15:0];  // load Z
            cV    <= slv_reg0[15:0];  // load Y
            cTh   <= 16'd0;
            cIter <= 4'd0;
            cBusy <= 1'b1;
```

```verilog
                cDone <= 1'b0;
            end
            else if (cBusy) begin
                if (cIter < 4'd13) begin
                    shftU = cU >>> cIter;  // arith shift
                    shftV = cV >>> cIter;

                    if (cV >= 0) begin
                        cU  <= cU + shftV;
                        cV  <= cV - shftU;
                        cTh <= cTh + atan_tbl[cIter];
                    end
                    else begin
                        cU  <= cU - shftV;
                        cV  <= cV + shftU;
                        cTh <= cTh - atan_tbl[cIter];
                    end
                    cIter <= cIter + 4'd1;
                end
                else begin
                    cBusy <= 1'b0;
                    cDone <= 1'b1;
                end
            end
        end
end

assign slv_reg3 = {31'd0, cDone};                // status: bit0 =
    done
assign slv_reg4 = {{16{cTh[15]}}, cTh};          // angle sign-
    extended
```

5 Verification Methodology

5.1 Simulation-Based Verification

Simulation was done using a testbench to verify before running the design on hardware.

5.1.1 Testbench Architecture

The Verilog testbench instantiates the complete CORDIC IP including the AXI-Lite wrapper. It performs the following operations:

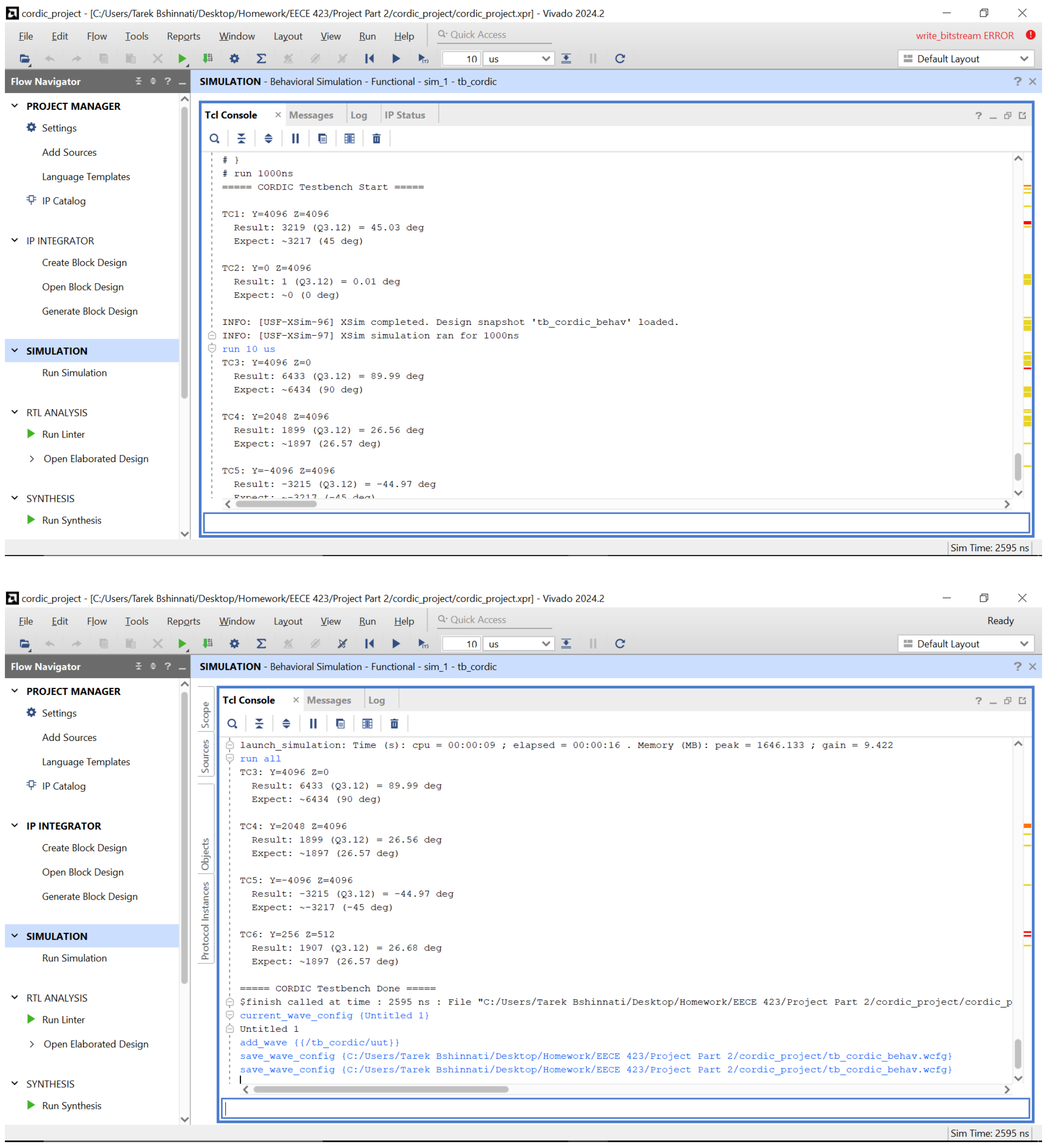
1. Initialize all inputs and reset the design

2. For each test case:

- Write Y and Z values via AXI-Lite write transactions
- Assert the start control bit
- Monitor the done status bit
- Read and verify the computed angle
- Compare against expected Q3.12 value
- Calculate error in both radians and degrees

3. Check pass/fail for each test case

5.1.2 Simulation Results and Waveform



```
# }
# run 1000ns
===== CORDIC Testbench Start =====

TC1: Y=4096 Z=4096
  Result: 3219 (Q3.12) = 45.03 deg
  Expect: ~3217 (45 deg)

TC2: Y=0 Z=4096
  Result: 1 (Q3.12) = 0.01 deg
  Expect: ~0 (0 deg)

INFO: [USF-XSim-96] XSim completed. Design snapshot 'tb_cordic_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
run 10 us
TC3: Y=4096 Z=0
  Result: 6433 (Q3.12) = 89.99 deg
  Expect: ~6434 (90 deg)

TC4: Y=2048 Z=4096
  Result: 1899 (Q3.12) = 26.56 deg
  Expect: ~1897 (26.57 deg)

TC5: Y=-4096 Z=4096
  Result: -3215 (Q3.12) = -44.97 deg
  Expect: ~-3217 (-45 deg)
```

Sim Time: 2595 ns

```
launch_simulation: Time (s): cpu = 00:00:09 ; elapsed = 00:00:16 . Memory (MB): peak = 1646.133 ; gain = 9.422
run all
TC3: Y=4096 Z=0
  Result: 6433 (Q3.12) = 89.99 deg
  Expect: ~6434 (90 deg)

TC4: Y=2048 Z=4096
  Result: 1899 (Q3.12) = 26.56 deg
  Expect: ~1897 (26.57 deg)

TC5: Y=-4096 Z=4096
  Result: -3215 (Q3.12) = -44.97 deg
  Expect: ~-3217 (-45 deg)

TC6: Y=256 Z=512
  Result: 1907 (Q3.12) = 26.68 deg
  Expect: ~1897 (26.57 deg)

===== CORDIC Testbench Done =====
$finish called at time : 2595 ns : File "C:/Users/Tarek Bshinnati/Desktop/Homework/EECE 423/Project Part 2/cordic_project/cordic_p
current_wave_config {Untitled 1}
Untitled 1
add_wave {{/tb_cordic/uut}}
save_wave_config {C:/Users/Tarek Bshinnati/Desktop/Homework/EECE 423/Project Part 2/cordic_project/tb_cordic_behav.wcfg}
save_wave_config {C:/Users/Tarek Bshinnati/Desktop/Homework/EECE 423/Project Part 2/cordic_project/tb_cordic_behav.wcfg}
```

Sim Time: 2595 ns

Figure 3: Simulation results

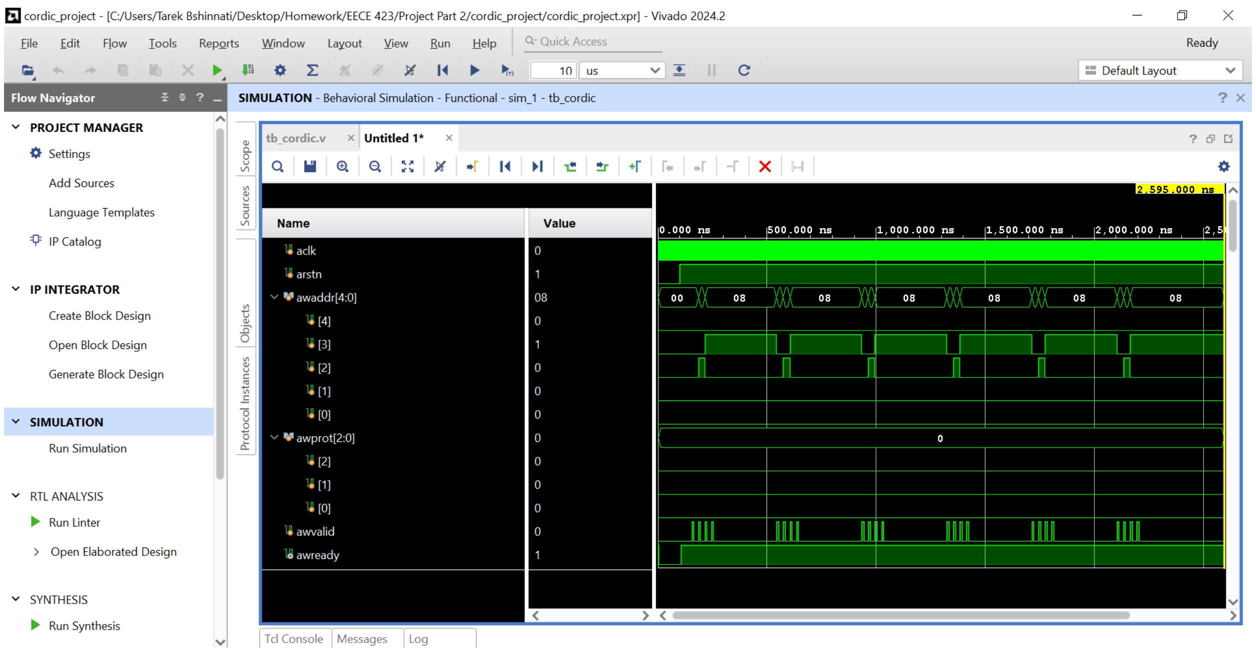


Figure 4: Simulation waveform

5.2 Hardware Verification

Following successful simulation, the design was implemented on the ZedBoard for hardware validation.

5.2.1 Test Methodology

Hardware testing used a simplified approach with known input values found in the cordic_test functions before reading on variable accelerometer readings:

1. Modified C code to use fixed Y, Z test values
2. Compared hardware CORDIC output against C math library atan2f()
3. Recorded results for the same test cases used in simulation
4. Read values from accelerometer and record results for both software and hardware
5. Calculated error between hardware and software implementations

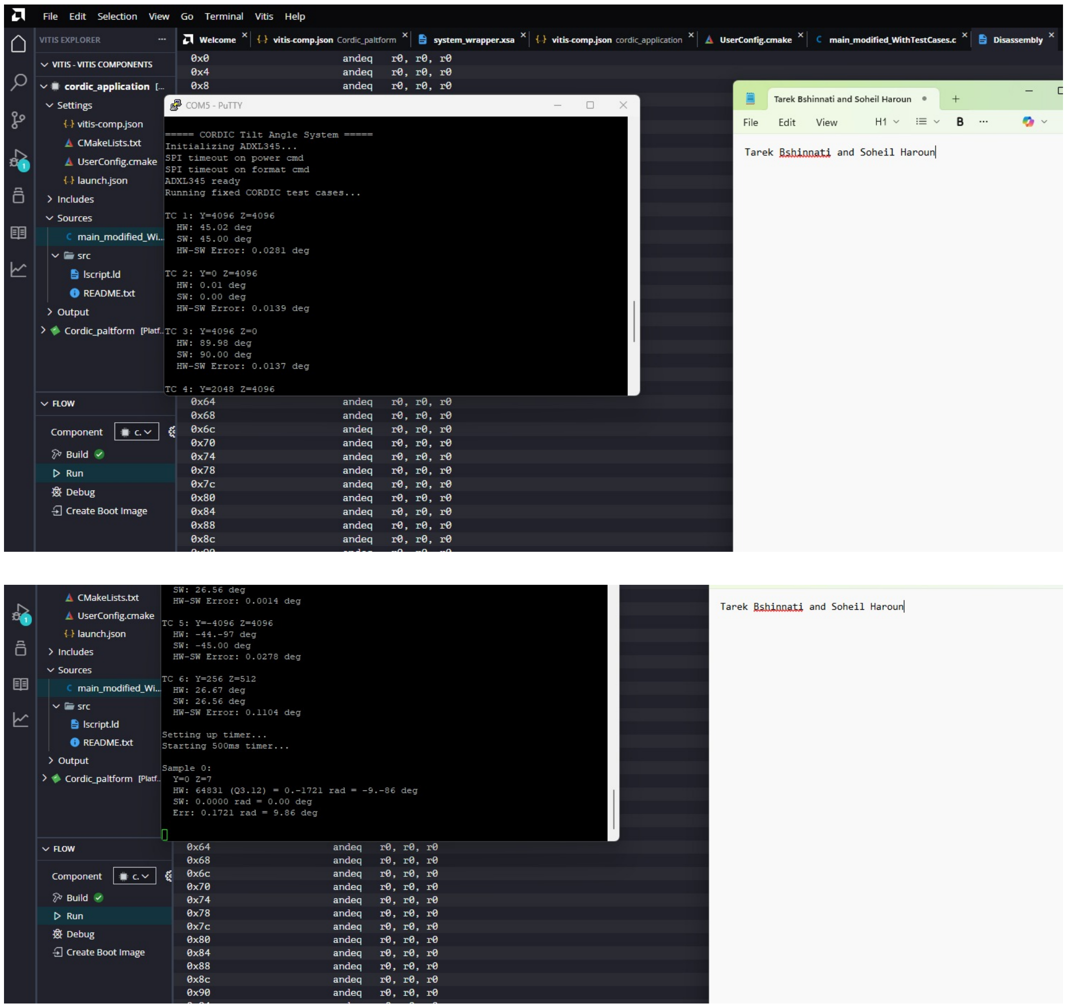
This approach provides cleaner validation than using accelerometer data, which can be affected by sensor noise, quantization, and mechanical vibrations.

5.2.2 Test Results

Table 2 compares hardware and software angle calculations (for both simulation values and recorded values).

Table 2: Three-way verification: Simulation, Hardware, and Software results

TC	Y	Z	SIM (deg)	HW (deg)	SW (deg)	HW-SW Error (deg)
1	4096	4096	45.03	45.02	45.00	0.0281
2	0	4096	0.01	0.01	0.00	0.0139
3	4096	0	89.99	89.98	90.00	0.0137
4	2048	4096	26.56	26.56	26.56	0.0014
5	-4096	4096	-44.97	-44.97	-45.00	0.0278
6	256	512	26.68	26.67	26.56	0.1104



```
0x0                  andeq   r0, r0, r0
0x4                  andeq   r0, r0, r0
0x8                  andeq   r0, r0, r0
```

```
===== CORDIC Tilt Angle System =====
Initializing ADXL345...
SPI timeout on power cmd
SPI timeout on format cmd
ADXL345 ready
Running fixed CORDIC test cases...

TC 1: Y=4096 Z=4096
  HW: 45.02 deg
  SW: 45.00 deg
  HW-SW Error: 0.0281 deg

TC 2: Y=0 Z=4096
  HW: 0.01 deg
  SW: 0.00 deg
  HW-SW Error: 0.0139 deg

TC 3: Y=4096 Z=0
  HW: 89.98 deg
  SW: 90.00 deg
  HW-SW Error: 0.0137 deg

TC 4: Y=2048 Z=4096
```

```
0x64                 andeq   r0, r0, r0
0x68                 andeq   r0, r0, r0
0x6c                 andeq   r0, r0, r0
0x70                 andeq   r0, r0, r0
0x74                 andeq   r0, r0, r0
0x78                 andeq   r0, r0, r0
0x7c                 andeq   r0, r0, r0
0x80                 andeq   r0, r0, r0
0x84                 andeq   r0, r0, r0
0x88                 andeq   r0, r0, r0
0x8c                 andeq   r0, r0, r0
```

Tarek Bshinnati and Soheil Haroun

```
  SW: 26.56 deg
  HW-SW Error: 0.0014 deg

TC 5: Y=-4096 Z=4096
  HW: -44.-97 deg
  SW: -45.00 deg
  HW-SW Error: 0.0278 deg

TC 6: Y=256 Z=512
  HW: 26.67 deg
  SW: 26.56 deg
  HW-SW Error: 0.1104 deg

Setting up timer...
Starting 500ms timer...

Sample 0:
  Y=0 Z=7
  HW: 64831 (Q3.12) = 0.-1721 rad = -9.-86 deg
  SW: 0.0000 rad = 0.00 deg
  Err: 0.1721 rad = 9.86 deg
```

```
0x64                 andeq   r0, r0, r0
0x68                 andeq   r0, r0, r0
0x6c                 andeq   r0, r0, r0
0x70                 andeq   r0, r0, r0
0x74                 andeq   r0, r0, r0
0x78                 andeq   r0, r0, r0
0x7c                 andeq   r0, r0, r0
0x80                 andeq   r0, r0, r0
0x84                 andeq   r0, r0, r0
0x88                 andeq   r0, r0, r0
0x8c                 andeq   r0, r0, r0
0x90                 andeq   r0, r0, r0
```

Tarek Bshinnati and Soheil Haroun

Figure 5: Hardware and software test cases results

The three methods showed very similar results, which shows that the 13-iteration CORDIC algorithm is accurate enough to measure tilt angle. The biggest difference was in TC6, where smaller input magnitudes amplified quantization effects (Absorbotion, loss of precision, etc... from math 251) in the Q3.12 fixed-point representation.

AI Tool Usage Acknowledgment

- **Code Development:** ChatGPT-5.1 and Google Gemini 3 pro were used for syntax clarification, debugging assistance, and optimization suggestions for both Verilog HDL and C implementations.
- **Code Documentation:** AI tools assisted in generating code comments and formatting suggestions to improve readability.
- **Report Writing:** contributed to structuring sections, refining technical explanations, and improving overall presentation clarity in this document.
- **Debugging Support:** AI tools provided insights during troubleshooting sessions, helping identify potential issues in both hardware and software components.

The AI assistance was used as a supplementary resource to enhance productivity, save us time, and help us in learning throughout the project process.