FPGA-Based Digital Compass Using the Nexys A7-100T and Pmod CMP32 Sensor

Christian Vanegas
College of Engineering
California State University, Long Beach
Long Beach, USA
Christian.Vanegas02@student.csulb.edu

Nathan Sarkozy
College of Engineering
California State University, Long Beach
Long Beach, USA
Nathan.Sarkozy@student.csulb.edu

Kaiya Hayashida
College of Engineering
California State University, Long Beach
Long Beach, USA
Kaiya.Hayashida@student.csulb.edu

Abstract – The purpose of this project is to design and implement a real-time digital compass on the Nexys A7-100T FPGA board. We will determine and display the precise heading angle in degrees relative to magnetic north using the Pmod CMP32 magnetometer. We will utilize the I²C communication protocol to process the raw X, Y, and Z-axis magnetic data and compute the correct directional heading. The result will be the angle in degrees delivered to the board's seven segment display for real-time feedback. This project demonstrates the use of digital logic design principles to create a hardware-based embedded system for sensor integration and real-time computation.

I. INTRODUCTION

This digital compass project applies the principles of combinational and sequential circuit design to create a fully functional hardware-based navigation system. The FPGA will acquire magnetic field data from the sensor, process it through custom logic to compute the real-time heading angle relative to magnetic north, and display the result on the board's seven-segment display. By integrating sensor interfacing, data processing, and real-time visualization, this project demonstrates how foundational digital logic concepts can be combined to implement a practical and sophisticated embedded system.

II. BACKGROUND & PRELIMINARIES

The concept of a digital compass relies on measuring the Earth's magnetic field to determine orientation relative to magnetic north. Modern electronic compasses typically rely on magnetometer sensors that output analog or digital readings corresponding to the magnetic field's strength along the X, Y, and Z axes. By processing these readings, the heading angle can be calculated using trigonometric relationships. The Pmod CMP32 magnetometer is a 3-axis magnetometer based on the LSM303DLHC chip, which communicates with external devices via the I^{2}C serial interface. This allows for precise and easily accessible magnetic field data.

In this project, the Nexys A7-100T FPGA serves as the central processing platform for real-time computation and display. Unlike microcontrollers, FPGA's allow for parallel digital logic design, making them ideal for custom hardware modules that handle sensor communication, signal processing, and display control simultaneously. Implementing the I^{2}C protocol in Verilog will enable direct communication between the FPGA board and the Pmod CMP32 sensor, while the computed angle will be displayed using the board's seven-segment display interface. An additional SPI driver will interface with the accelerometer for tilt measurement and compensation.

To achieve accurate directional readings regardless of device orientation, a tilt compensation circuit will process data from both the magnetometer and accelerometer to correct for roll and pitch angles. The heading calculation circuit will then use the compensated magnetic field data to compute the true heading relative to magnetic north. Finally, the seven-segment display driver (which will require a time-multiplexing scheme due to shared segment lines) will present the calculated heading angle in real time. Together, these modules form a complete FPGA-based embedded system capable of sensor integration, tilt correction, real-time angle computation, and visual output.

Currently, the obstacles we are encountering involve understanding how the I2C master will be programmed to effectively communicate with the CMP32 and the FPGA board. Additionally, we are struggling to decide what computational approach we will proceed with to compute the raw X/Y/Z data into a directional heading. Lastly, we are also doing more research into how to incorporate our tilt compensation circuit to work in tandem with our on-board accelerometer as it communicates with the FPGA via SPI.

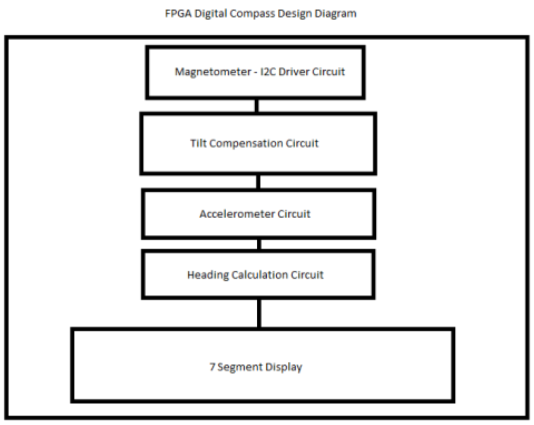

FPGA Digital Compass Design Diagram
Magnetometer - I2C Driver Circuit
Tilt Compensation Circuit
Accelerometer Circuit
Heading Calculation Circuit
7 Segment Display

Figure 1. Rough Design Diagram of Digital Compass

III. IMPLEMENTATION

The bcd_to_7seg module is responsible for converting a 4-bit binary-coded decimal (BCD) input into the corresponding seven-segment display pattern. Each BCD input value, ranging from 0 to 9, is mapped to a unique 7-bit output pattern that determines which segments of the display should be illuminated to form the correct decimal digit. The output follows the common-cathode configuration, where a logic '0' turns on a segment and a logic '1' turns it off. This mapping is implemented through a combinational case statement inside an always block that is sensitive to any change in the BCD input. If the input value falls outside the valid BCD range, the module outputs 7'b1111111, which turns off all segments. This module plays a key role in the display system by enabling numerical data (such as the computed heading angle) to be accurately visualized on the FPGA's seven-segment display.

The binary_to_bcd module converts a 10-bit binary number into its equivalent decimal representation using three separate 4-bit outputs for the hundredths, tenths, and ones digits. Since the input range is limited to 0–511 (the maximum value representable by 9 bits), each of the output digits is computed through basic arithmetic operations: integer division and modulus. Specifically, the hundredths digit is obtained by dividing the binary input by 100, the tenth digit is derived from the remainder after removing the hundreds place, and the ones digit represents the final remainder. To ensure safe operation, a conditional check limits the maximum displayable value to 511; if the input exceeds this range, the outputs default to "511." This module serves as an intermediate stage between the computed heading angle and the display system, formatting binary angle data into readable decimal digits that can be properly decoded by the bcd_to_7seg module and shown on the seven-segment display.

The top module serves as the main control unit that integrates all functional components required to display numerical data on the Nexys A7's seven-segment display. The module accepts a 9-bit binary input from the board's switches, representing values from 0 to 511, and uses the binary_to_bcd converter to separate this value into individual hundredths, tenths, and ones digits. Each of these digits is then passed through an instance of the bcd_to_7seg decoder, which generates the corresponding segment pattern for display. To manage the multiple seven-segment displays with limited I/O pins, the design employs a time-multiplexing technique using a 20-bit refresh counter. This counter cycles rapidly through the active digits, updating one display at a time while keeping the others off. Because this switching occurs faster than the human eye can perceive, the digits appear continuously illuminated. The result is a stable three-digit display that accurately shows the binary switch input as a readable decimal number, demonstrating real-time conversion, decoding, and multiplexed display control entirely in hardware.

The top_tb module functions as a testbench designed to verify the correct operation of the binary_to_bcd converter. It simulates the behavior of the FPGA in a controlled environment, allowing for step-by-step observation of the output values without requiring physical hardware. Within the testbench, a 9-bit input signal named switches represents the binary values that would normally come from the FPGA's physical switches. These inputs are sequentially assigned different test cases ranging from 0 to 359 to ensure accurate binary-to-decimal conversion across the expected range. The resulting BCD outputs (hundredths, tenths, and ones) can then be monitored to confirm proper functionality. Although a clock signal is generated to mimic FPGA timing, it primarily serves as a placeholder for future integration with time-dependent logic. The inclusion of a reset signal ensures consistent initialization before testing begins.

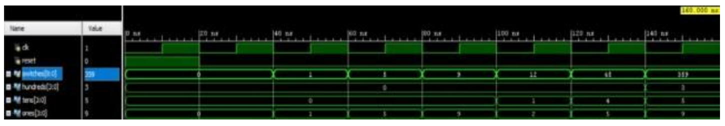


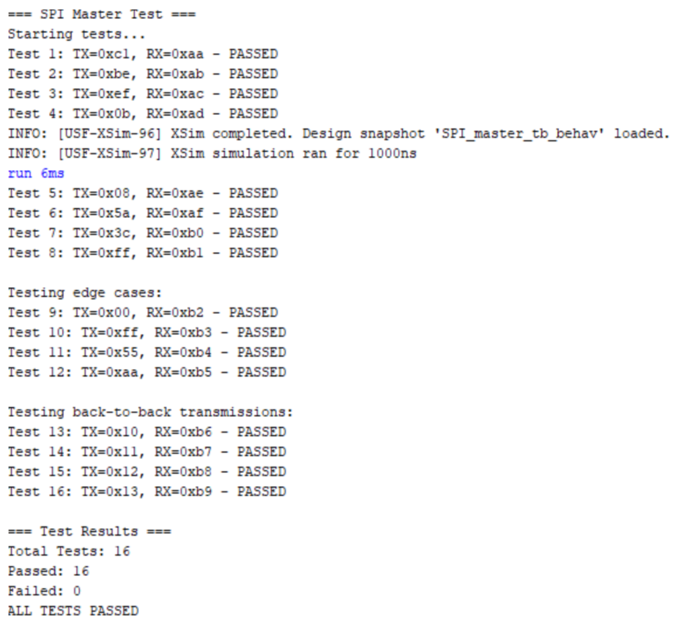
Figure 2. 7 Segment Decoder Simulation

The magnetometer_driver module is a Verilog design that interfaces with the MMC34160PJ magnetometer sensor over an I²C bus to perform magnetic field measurements along the X, Y, and Z axes. It coordinates communication with the sensor using an internal finite state machine (FSM) that handles initialization, data reading, and data collection. Upon reset, the module waits for a start_read signal, after which it either configures the magnetometer for continuous measurement mode or initiates a new reading if already initialized. During a read operation, it requests six bytes of data corresponding to the high and low bytes of each axis, combines them into signed 16-bit outputs (mag_x, mag_y, mag_z), and signals when valid data is available. The design also supports a testbench mode, allowing the I²C master interface signals to be exposed for simulation and debugging. This modular and flexible structure enables reliable sensor initialization, data acquisition, and I²C communication management for real-time magnetic field measurement applications.

The i2c_master module is a Verilog implementation of an I²C bus master controller designed to handle communication with I²C slave devices at up to 400 kHz (fast mode). It operates based on a finite state machine (FSM) that generates the proper timing and control signals for I²C operations such as START, STOP, read, and write sequences. The module takes inputs for the device address, register address, data to write, and number of bytes to read, while providing outputs to indicate when data is valid, when the bus is busy, when the transaction is complete, or when a NACK (no acknowledge) is detected. It uses open-drain control for the SDA and SCL lines, ensuring proper I²C bus behavior, and internally divides the system clock into timing phases for bit-level synchronization. This design supports both single-byte write operations and multi-byte read transactions, making it suitable for interfacing with sensors and other I²C-compatible peripherals such as the MMC34160PJ magnetometer.

The SPI_Master module is a Verilog implementation of a Serial Peripheral Interface (SPI) master controller, capable of operating in all four SPI modes (0–3) with a configurable clock speed. It manages the generation of the SPI clock (o_SPI_Clk), transmission of data on the MOSI line, and reception on the MISO line, synchronized according to the selected clock polarity (CPOL) and phase (CPHA). The module uses a precise counter-based mechanism to toggle the SPI clock and track leading and trailing edges for correct data timing. Transmission begins when a valid data pulse (i_TX_DV) is received, and the design shifts out bits MSB-first while simultaneously capturing incoming bits from the slave device. Once a full byte is transmitted and received, the o_RX_DV signal pulses high to indicate valid received data, and o_TX_Ready signals readiness for the next byte. This flexible and parameterized design enables reliable full-duplex SPI communication between an FPGA and external peripherals, with easy integration into a wide range of digital systems.

SPI_master_tb is a testbench designed to verify the functionality of the SPI_Master module by simulating various data transmission and reception scenarios. It generates a 100 MHz clock signal and applies a reset sequence before executing a series of tests. The testbench initializes predefined transmit (TX) and expected receive (RX) byte patterns, drives the SPI Master module through its transmit interface, and monitors the received data from the MISO line, which follows a predictable bit pattern for simulation. The SendAndVerify task automates the process of sending each byte, waiting for completion, and checking the received value against the expected result, displaying pass/fail messages accordingly. Additionally, the testbench performs edge case tests (such as all zeros, all ones, and alternating bit patterns) and back-to-back transmissions to evaluate robustness. At the end of simulation, it provides a summary of the total, passed, and failed tests, ultimately implementing a self-checking simulation testbench.



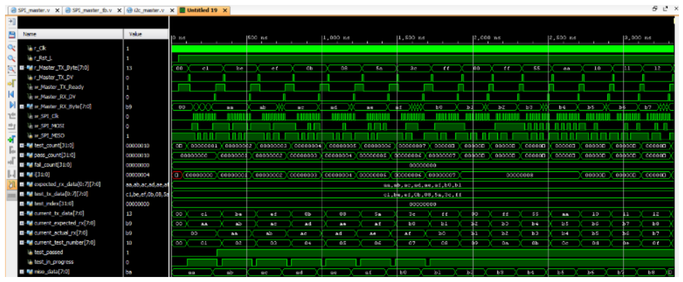
```
=== SPI Master Test ===
Starting tests...
Test 1: TX=0xc1, RX=0xaa - PASSED
Test 2: TX=0xbe, RX=0xab - PASSED
Test 3: TX=0xef, RX=0xac - PASSED
Test 4: TX=0x0b, RX=0xad - PASSED
INFO: [USF-XSim-96] XSim completed. Design snapshot 'SPI_master_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
run 6ms
Test 5: TX=0x08, RX=0xae - PASSED
Test 6: TX=0x5a, RX=0xaf - PASSED
Test 7: TX=0x3c, RX=0xb0 - PASSED
Test 8: TX=0xff, RX=0xb1 - PASSED

Testing edge cases:
Test 9: TX=0x00, RX=0xb2 - PASSED
Test 10: TX=0xff, RX=0xb3 - PASSED
Test 11: TX=0x55, RX=0xb4 - PASSED
Test 12: TX=0xaa, RX=0xb5 - PASSED

Testing back-to-back transmissions:
Test 13: TX=0x10, RX=0xb6 - PASSED
Test 14: TX=0x11, RX=0xb7 - PASSED
Test 15: TX=0x12, RX=0xb8 - PASSED
Test 16: TX=0x13, RX=0xb9 - PASSED

=== Test Results ===
Total Tests: 16
Passed: 16
Failed: 0
ALL TESTS PASSED
```

Figure 3. SPI Master Testbench Console Output



Name	Value
r_Clk	1
r_Rst_L	1
r_Master_TX_Byte[7:0]	13
r_Master_TX_DV	0
w_Master_TX_Ready	1
w_Master_RX_DV	0
w_Master_RX_Byte[7:0]	b9
w_SPI_Clk	0
w_SPI_MOSI	0
w_SPI_MISO	1
test_count[31:0]	00000010
pass_count[31:0]	00000010
fail_count[31:0]	00000000
i[31:0]	00000004
expected_rx_data[0:7][7:0]	aa,ab,ac,ad,ae,af
test_tx_data[0:7][7:0]	c1,be,ef,0b,08,5a
test_index[31:0]	00000000
current_tx_data[7:0]	13
current_expected_rx[7:0]	b9
current_actual_rx[7:0]	b9
current_test_number[7:0]	10
test_passed	1
test_in_progress	0
miso_data[7:0]	ba

Figure 4. SPI Master Testbench Simulation Waveform

IV. PROGRESS

Since the first progress report, we have significantly expanded our project by developing and integrating multiple key modules to improve sensor communication and data acquisition reliability. We implemented the magnetometer_driver module to interface with the MMC34160PJ magnetometer via I²C, managing sensor initialization, configuration, and magnetic field data collection through a finite state machine. Alongside it, we designed an i2c_master module capable of handling standard and fast-mode I²C communication, supporting both single-byte writes and multi-byte reads to ensure accurate data transfer between the FPGA and sensor. To complement these, we developed an SPI_Master module with full support for all four SPI modes and configurable clock speeds, enabling robust full-duplex communication for future sensor or peripheral integration. Finally, we created the SPI_master_tb, a comprehensive self-checking testbench that automates the verification of SPI transactions, testing normal and edge cases for reliability.

One of the main challenges we encountered was debugging synchronization and timing issues between modules, particularly ensuring proper coordination between the I²C FSM states and the data-valid signals during sensor reads. Additionally, designing flexible communication modules that adhere to protocol timing specifications while remaining simulation-friendly required careful parameterization and testing. Despite these challenges, these updates have improved our system's scalability, testability, and real-time data handling capabilities.

V. TEAM CONTRIBUTION

The project distribution percentage at this stage can be described as follows: 33% Nathan Sarkozy, 33% Christian Vanegas, and 33% Kaiya Hayashida.

- Nathan Sarkozy contributed by designing and implementing the SPI_Master module, which supports all four SPI modes and handles full-duplex data transmission with configurable clock speeds. He also developed the SPI_master_tb, a comprehensive self-checking testbench used to verify SPI functionality through automated transmit and receive tests, including edge case and back-to-back transmission scenarios. His work ensured reliable SPI communication and robust module verification.
- Christian Vanegas contributed by developing the i2c_master module and the magnetometer_driver module. The i2c_master module handles I²C protocol communication with accurate timing, acknowledgment detection, and multi-byte read and write capabilities. The magnetometer_driver integrates with the MMC34160PJ sensor, managing initialization, configuration, and magnetic field data collection through a finite state machine. His work established the foundation for accurate sensor interfacing and data acquisition.
- Kaiya Hayashida contributed by completing a significant portion of the progress report. In addition to documentation, she assisted with the integration and organization of the modules developed by Christian and Nathan, helping connect the I²C and SPI components within the overall system design and ensuring smooth interaction between the magnetometer driver, I²C master, and SPI master modules.