Marwadi University	Marwadi University Faculty of Engineering and Technology Department of Information and Communication Technology	
Subject: Capstone Project (01CT1718)	Implementation	
	Date: 19/09/2025	Enrolment No: 92200133031

Implementation

Introduction:

The objective of the Capstone project is to build reliable communication protocols like UART/I2C over FPGA board on Altera Cyclone II or Nexys4 DDR to easily interface sensors and different module to interface the Input and Output via FPGA boards using the unique features of the FPGA board. It will help the people to understand the FPGA and communication protocol structure and use of it while testing.

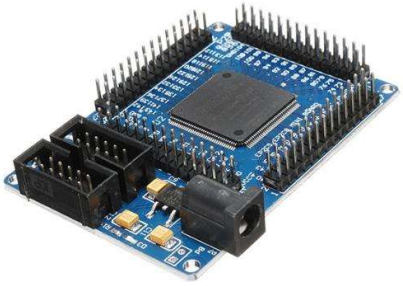
Project Proposal:

A Field-Programmable Gate Array (FPGA) is a semiconductor device based on a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs are reconfigurable, meaning that the logic functions within the FPGA can be reprogrammed to implement different designs or algorithms after manufacturing. This reconfigurability contrasts with fixed-function Application-Specific Integrated Circuits (ASICs), making FPGAs highly versatile for prototyping and deployment in a wide range of applications.

The FPGA boards are quite faster than usual microcontroller because of its capability to adapt the advantage of different parameter like high speed parallel processing, custom GPIO interfaces, frequency trading, While FPGAs offer superior hardware-level performance and flexibility, microcontrollers are generally better for cost-sensitive or low-power applications with sequential tasks, thanks to their easier development process and lower unit cost.

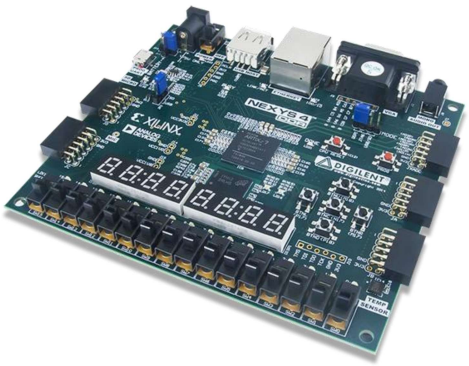
Here we can use FPGA board of Altera Cyclone II or Xilinx Nexys 4 DDR, which are very well known FPGA Board for implementing Prototype of HDL design on it and test its compatibility for moving forward the Chip manufacturing or for making a design for an system in less time to build application with low cost.

Altera Cyclone II with EP2C5T144 chip



- EPROM chip configuration using EPCS4, the size of 4Mbit
- Onboard 50M active patch crystal (Crystal Slaughter halfback)
- Power supply with a largemouth outlet, single 5V power supply
- Board has power indicator and resets switch
- Onboard 3 SMD LED, the LED test can be done an experiment, more experiments using lead to complete

Nexys 4 DDR with XC7A100TCS0324A



Nexys 4 DDR

Artix-7 FPGA

Features	
• Programmable over JTAG and Quad-SPI Flash • On-chip analog-to-digital converter	
Key Specifications	
FPGA Part #	XC7A100T-1CSG324C
Logic Slices	15,850 (4 6-input LUTs & 8 flip-flops each)
Block RAM	4,860 Kbits
Clock Tiles	6 (each with PLL)
DSP Slices	240
Internal clock	450 MHz+
DDR2	128 MiB
Cellular RAM	16MB
Ethernet	10/100 PHY
Connectivity and Onboard I/O	
SD	microSD card connector
Pmod Connectors	4 Pmod ports
VGA	12-bit VGA port
Audio	PWM audio output
Microphone	PDM mic
Temp sensor	One temperature sensor
Display	2 4-digit seven segment displays
Switches	16
Buttons	4
LEDs	16
Tri-color LEDs	2

Hardware Platform

The core hardware is the **Xilinx Nexys 4 DDR FPGA board** (featuring the XC7A100TCS0324A chip). This board was chosen for its widespread use in academia and its onboard peripherals that simplify integration testing.

- **UART:** Implemented for basic data transmission and receiving, typically using a loopback or USB-to-UART interface for monitoring.

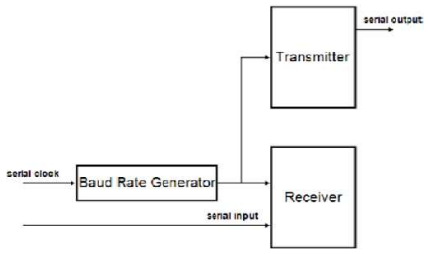


Fig. 1 UART Module

- **SPI Sensor:** The onboard ADXL362 3-axis accelerometer is used to demonstrate SPI communication. The FPGA acts as the Master, reading 12-bit acceleration data from the sensor's registers.

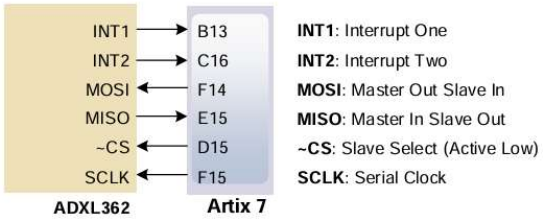
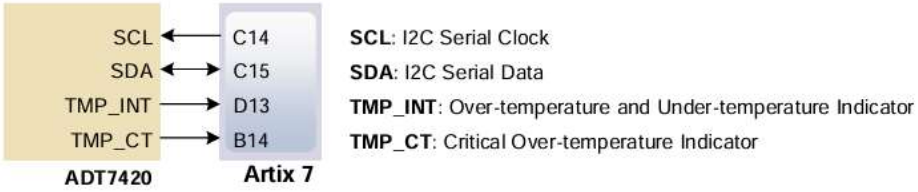


Figure 23. Accelerometer interface.

- **I2C Sensor:** The onboard ADT7420 temperature sensor is used to validate I2C communication, with the FPGA serving as the Controller/Master. The I2C bus address 0x4B was selected for interfacing.

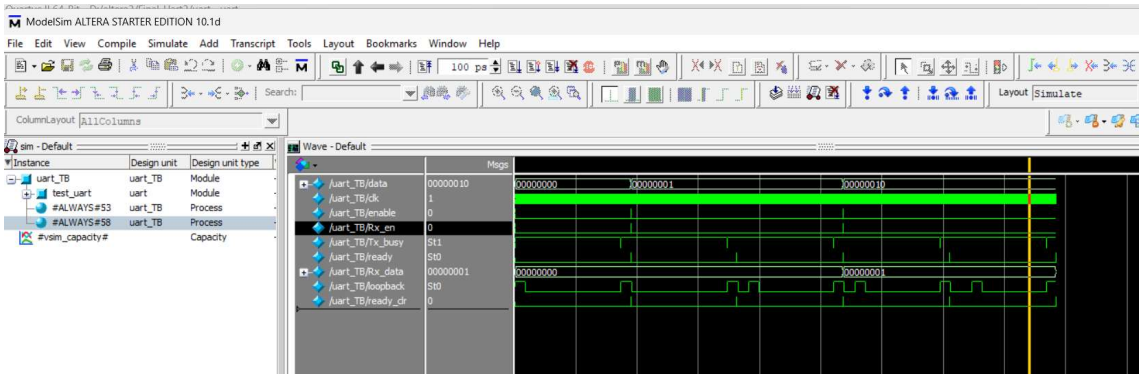


Implemented Simulation :

The project utilized Verilog HDL for all hardware designs, and Vivado (for Xilinx) and Quartus II (for Altera) for synthesis and implementation.

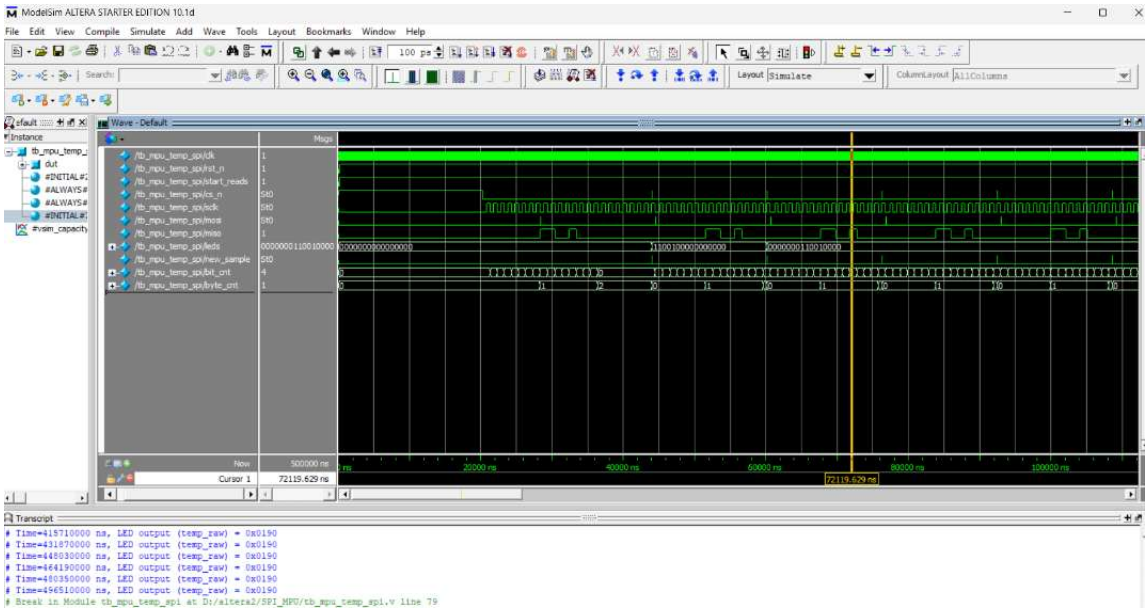
ModelSim was used extensively for simulation and verification via testbenches before deployment to the physical hardware.

UART simulation testing :

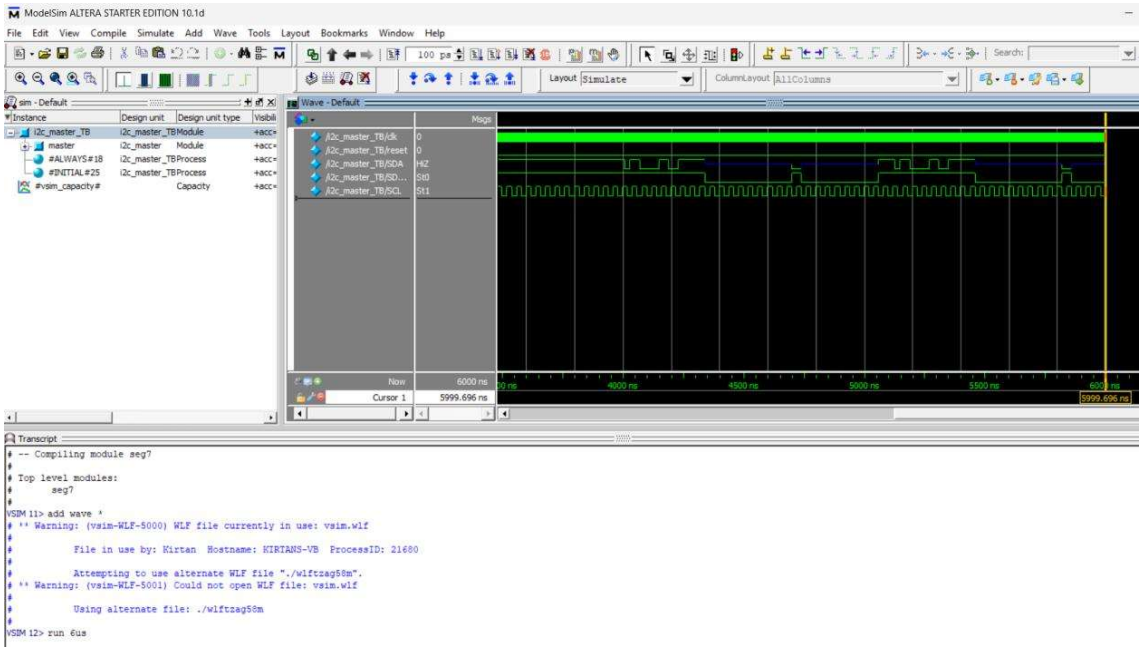


SPI simulation testing :

Marwadi University	Marwadi University Faculty of Engineering and Technology Department of Information and Communication Technology
Subject: Capstone Project (01CT1718)	Implementation
	Date: 19/09/2025 | Enrolment No: 92200133031



I2C simulation testing :



Implemented Code Snippets:

UART: The below code is snippent for baud rate generation

```
module baud_rate_generator #(
    parameter CLOCK_FREQ = 100_000_000, // 100 MHz default for Nexys4 DDR
    parameter BAUD_RATE  = 9600         // UART baud rate
)(
    input  wire clk,
    input  wire rst_n,
    output reg  baud_clk_en // One-cycle pulse at baud rate
);

    localparam COUNTER_LIMIT = (CLOCK_FREQ + BAUD_RATE/2) / BAUD_RATE - 1;

    reg [31:0] counter;

    always @(posedge clk or negedge rst_n) begin
        if (!rst_n) begin
            counter <= 0;
            baud_clk_en <= 1'b0;
        end else begin
            if (counter == COUNTER_LIMIT) begin
                counter <= 0;
                baud_clk_en <= 1'b1;
            end else begin
                counter <= counter + 1;
                baud_clk_en <= 1'b0;
            end
        end
    end

endmodule
```

SPI: The below code is of top module frame for SPI protocol between ADXL362 sensor and FPGA board

```
    iclk_gen clock_generation(
        .CLK100MHZ(CLK100MHZ),
        .clk_4MHz(w_4MHz)
    );

    spi_master master(
        .iclk(w_4MHz),
```

```
    .miso(ACL_MISO),
    .sclk(ACL_SCLK),
    .mosi(ACL_MOSI),
    .cs(ACL_CSN),
    .acl_data(acl_data)
);

seg7_control display_control(
    .CLK100MHZ(CLK100MHZ),
    .acl_data(acl_data),
    .seg(SEG),
    .dp(DP),
    .an(AN)
);
```

FSM of master module:

```
// SPI WRITE
                                    BEGIN_SPIW  = 7'h01,
            // Send write instruction 0x0A
                                    SEND_WCMD7   = 7'h02,
                                    SEND_WCMD6   = 7'h03,
                                    SEND_WCMD5   = 7'h04,
                                    SEND_WCMD4   = 7'h05,
                                    SEND_WCMD3   = 7'h06,
                                    SEND_WCMD2   = 7'h07,
                                    SEND_WCMD1   = 7'h08,
                                    SEND_WCMD0   = 7'h09,
            // Send register address to write to 0x2D
                                    SEND_WADDR7  = 7'h0A,
                                    SEND_WADDR6  = 7'h0B,
                                    SEND_WADDR5  = 7'h0C,
                                    SEND_WADDR4  = 7'h0D,
                                    SEND_WADDR3  = 7'h0E,
                                    SEND_WADDR2  = 7'h0F,
                                    SEND_WADDR1  = 7'h10,
                                    SEND_WADDR0  = 7'h11,
            // Send byte to put into measurement mode 0x02
                                    SEND_BYTE7  = 7'h12,
                                    SEND_BYTE6  = 7'h13,
                                    SEND_BYTE5  = 7'h14,
                                    SEND_BYTE4  = 7'h15,
                                    SEND_BYTE3  = 7'h16,
                                    SEND_BYTE2  = 7'h17,
```

```
                    SEND_BYTE1  = 7'h18,
                    SEND_BYTE0  = 7'h19,
// Wait for first valid data after init measurement mode = 40ms
                    WAIT        = 7'h1A,
                    // SPI READ
                    BEGIN_SPIR  = 7'h1B,
                    // Send read instruction 0x0B
                    SEND_RCMD7  = 7'h1C,
                    SEND_RCMD6  = 7'h1D,
                    SEND_RCMD5  = 7'h1E,
                    SEND_RCMD4  = 7'h1F,
                    SEND_RCMD3  = 7'h20,
                    SEND_RCMD2  = 7'h21,
SEND_RCMD1  = 7'h22,
                    SEND_RCMD0  = 7'h23,
                    // Send X data LSB register address 0x0E
                    SEND_RADDR7 = 7'h24,
                    SEND_RADDR6 = 7'h25,
                    SEND_RADDR5 = 7'h26,
                    SEND_RADDR4 = 7'h27,
                    SEND_RADDR3 = 7'h28,
                    SEND_RADDR2 = 7'h29,
SEND_RADDR1 = 7'h2A,
SEND_RADDR0 = 7'h2B,
                    // Receive X data LSB from 0x0E
                    REC_XLSB7   = 7'h2C,
                    REC_XLSB6   = 7'h2D,
                    REC_XLSB5   = 7'h2E,
                    REC_XLSB4   = 7'h2F,
                    REC_XLSB3   = 7'h30,
                    REC_XLSB2   = 7'h31,
                    REC_XLSB1   = 7'h32,
                    REC_XLSB0   = 7'h33,
                    // Receive X data MSB from 0x0F
                    REC_XMSB7   = 7'h34,
                    REC_XMSB6   = 7'h35,
                    REC_XMSB5   = 7'h36,
                    REC_XMSB4   = 7'h37,
                    REC_XMSB3   = 7'h38,
                    REC_XMSB2   = 7'h39,
                    REC_XMSB1   = 7'h3A,
                    REC_XMSB0   = 7'h3B,
                    // Receive Y data LSB from 0x10
```

```
REC_YLSB7    = 7'h3C,
REC_YLSB6    = 7'h3D,
REC_YLSB5    = 7'h3E,
REC_YLSB4    = 7'h3F,
REC_YLSB3    = 7'h40,
REC_YLSB2    = 7'h41,
REC_YLSB1    = 7'h42,
REC_YLSB0    = 7'h43,
// Receive Y data MSB from 0x11
REC_YMSB7    = 7'h44,
REC_YMSB6    = 7'h45,
REC_YMSB5    = 7'h46,
REC_YMSB4    = 7'h47,
REC_YMSB3    = 7'h48,
REC_YMSB2    = 7'h49,
REC_YMSB1    = 7'h4A,
REC_YMSB0    = 7'h4B,
// Receive Z data LSB from 0x12
REC_ZLSB7    = 7'h4C,
REC_ZLSB6    = 7'h4D,
REC_ZLSB5    = 7'h4E,
REC_ZLSB4    = 7'h4F,
REC_ZLSB3    = 7'h50,
REC_ZLSB2    = 7'h51,
REC_ZLSB1    = 7'h52,
REC_ZLSB0    = 7'h53,
// Receive Z data MSB from 0x13
REC_ZMSB7    = 7'h54,
REC_ZMSB6    = 7'h55,
REC_ZMSB5    = 7'h56,
REC_ZMSB4    = 7'h57,
REC_ZMSB3    = 7'h58,
REC_ZMSB2    = 7'h59,
REC_ZMSB1    = 7'h5A,
REC_ZMSB0    = 7'h5B,
// End SPI communications
END_SPI      = 7'h5C;
```

I2C: The below code is of top module frame for SPI protocol between ADT7420 sensor and FPGA

Top module structure:

```
i2c_master master(
      .clk_200kHz(w_200kHz),
      .reset(reset),
      .temp_data(w_data),
      .SDA(TMP_SDA),
      .SDA_dir(sda_dir),
      .SCL(TMP_SCL)
   );

   // Instantiate 200kHz clock generator
   clkgen_200kHz cgen(
      .clk_100MHz(CLK100MHZ),
      .clk_200kHz(w_200kHz)
   );

   // Instantiate 7 segment control
   seg7 seg(
      .clk_100MHz(CLK100MHZ),
      .temp_data(w_data),
      .SEG(SEG),
      .NAN(NAN),
      .AN(AN)
   );
```

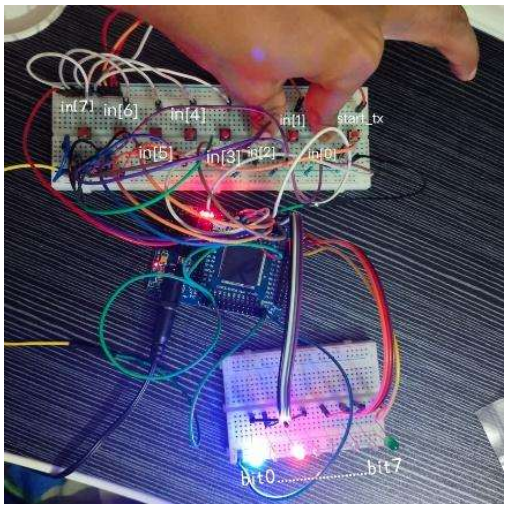
FSM for master module for I2C:

```
// State Declarations - need 28 states
   localparam [4:0] POWER_UP   = 5'h00,
                    START      = 5'h01,
                                          SEND_ADDR6 = 5'h02,
                                          SEND_ADDR5 = 5'h03,
                                          SEND_ADDR4 = 5'h04,
                                          SEND_ADDR3 = 5'h05,
                                          SEND_ADDR2 = 5'h06,
                                          SEND_ADDR1 = 5'h07,
                                          SEND_ADDR0 = 5'h08,
                                          SEND_RW    = 5'h09,
                    REC_ACK    = 5'h0A,
                    REC_MSB7   = 5'h0B,
                                          REC_MSB6     = 5'h0C,
                                          REC_MSB5     = 5'h0D,
                                          REC_MSB4     = 5'h0E,
```

```
                                REC_MSB3    = 5'h0F,
                                REC_MSB2    = 5'h10,
                                REC_MSB1    = 5'h11,
                                REC_MSB0    = 5'h12,
SEND_ACK   = 5'h13,
                               REC_LSB7   = 5'h14,
                                REC_LSB6    = 5'h15,
                                REC_LSB5    = 5'h16,
                                REC_LSB4    = 5'h17,
                                REC_LSB3    = 5'h18,
                                REC_LSB2    = 5'h19,
                                REC_LSB1    = 5'h1A,
                                REC_LSB0    = 5'h1B,
NACK       = 5'h1C;
```

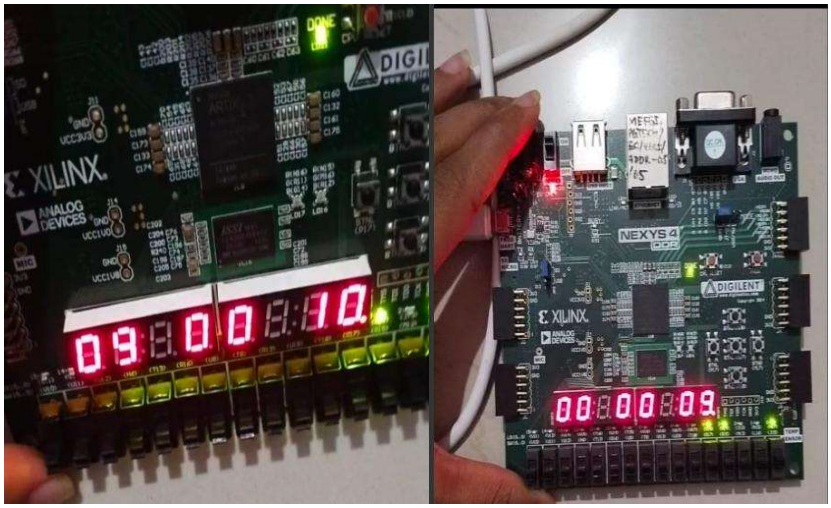
Implemented Hardware:

UART with RX TX: The below photo is test of UART on FPGA board of Altera Cyclone 2 whre while pressing start TX button we ca send desired data and see the same on LED by connecting RX and TX together.



SPI with 3 axis Acceleration reading: The SPI protocol was implemented to communicate with the ADXL362 accelerometer. The result, representing X, Y, and Z axis acceleration data, was successfully displayed on the Nexys 4 DDR's seven-segment display.

Marwadi University	Marwadi University Faculty of Engineering and Technology Department of Information and Communication Technology	
Subject: Capstone Project (01CT1718)	Implementation	
	Date: 19/09/2025	Enrolment No: 92200133031



I2C with Temperature reading: The I2C Master module was designed to interact with the ADT7420 temperature sensor. The captured temperature data was successfully displayed on the seven-segment display.