Description of Module and Testbench

Sensor Module

The sensor module is made up of three inputs, sclk, cs_n, and expected_value and an output sdo. The sclk input directly controls the conversion and readout process by updating sdo on the negative edge of sclk. The cs_n input determines when the conversion process begins which is on its falling edge. The expected_value input is an 8-bit input port that gets the next value to be sent from the testbench. Output sdo samples are clocked out of the sdo pin on falling edges of the sclk pin as determined by the normal mode of operation, which is entered when cs_n is pulled low. Sixteen sclk cycles are required to read all of a conversion word from the device, after which cs_n may be idled high or low until the next conversion. Three leading zeros and three trailing zeros surround the data bits in a given conversion. Also, there is a 40 ns delay before the sdo output is set from the falling edge of sclk to model tacc, data access time, from the specification.

Testbench

The testbench generates the sensor modules input and output signals including sclk, cs_n, expected_value and sdo. The sclk signal is set to 2MHz, meeting the frequency requirements, fsclk, in the ALS specification. The cs_n signal is set to be low for 16 sclk clock cycles and high for 4 sclk clock cycles, meeting the duty cycle requirements defined by the low and high pulse width, tcl and tch, in the ALS specification. There is a 20 ns delay inside of the sclk always block to model the data access time described previously. Also, an actual_value signal is generated by sampling the sdo output from the sensor module at the corresponding indices, taking the leading and trailing zeros into account. Then, the actual output is compared to the expected_value input which reports pass if they are the same, and fails otherwise.

Test Sequence Generation

The test sequence was generated by starting with an expected_value input of 0xf5 and then shifting left 1 bit after each transaction. This allowed for a different expected_value input to be tested over 4 transactions: 0xf5, 0xea, 0xd4, and 0xa8. The testbench generates the expected_value, feeds it into the sensor module, samples the sdo output, and then compares the actual and expected values. The sensor module generates the sdo output for use as the actual_value in the testbench.

Simulation Results and Analysis

First of all, I checked that the actual_value sampled by the testbench matches the expected_value sensor module input. As shown in the pass/fail and time report below, the results of these comparisons were recorded in the messages window for each of the 4 read operations. Also, I checked the sdo bit sequence to make sure that it also matches the actual and expected values. As shown in the single transaction waveform below, I can see that an expected_value input of 0xf5 generated the following sdo sequence updated on the falling edge of sclk: 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0. This waveform includes the 3 leading zeros, 4 trailing zeros, and correct 8 data bits expected for this transaction. Similarly, I verified that the sdo output sequence matched the other expected_value inputs for the 4 consecutive transactions.

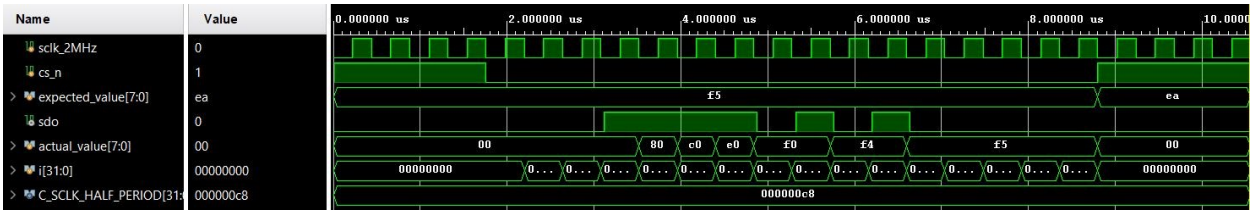

Figure 1: Waveform with single transaction

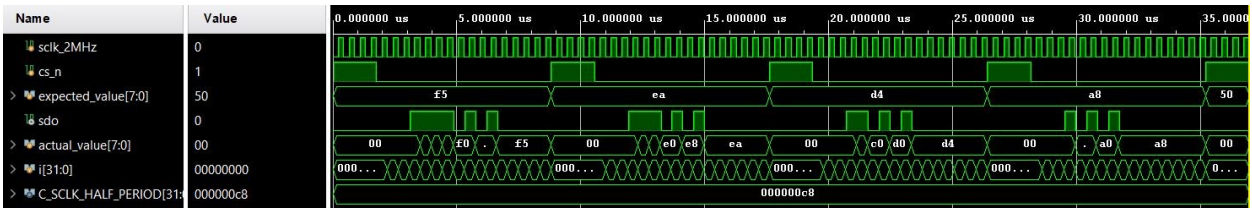

Figure 2: Waveform with 4 consecutive transactions

```
PASS: Expected: 11110101, Actual: 11110101
Time: 804.000 ns
PASS: Expected: 11101010, Actual: 11101010
Time: 1608.000 ns
PASS: Expected: 11010100, Actual: 11010100
Time: 2412.000 ns
PASS: Expected: 10101000, Actual: 10101000
Time: 3216.000 ns
```

Figure 3: Pass/Fail and time report for 4 transactions

Appendix with Verilog Code

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: Worcester Polytechnic Institute
// Engineer:
//  Adam Grabowski
// Create Date: 09/21/2022 01:52:04 AM
// Design Name: ECE 3829 Lab 3
// Module Name: als_pmod_bfm
// Project Name: ECE 3829 Lab 3
// Target Devices: Basys 3 Artix-7 Development Board
// Tool Versions: Verilog 2021.1
// Description:
//  Specifies the inputs and outputs for the light sensor bus function module
// Dependencies:
//  None
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//  None
//////////////////////////////////////////////////////////////////////////////////


module als_pmod_bfm(
    input sclk, // clock input
    input cs_n, // chip select
    input [7:0] val, // value sent
    output reg sdo = 1'b0 // data output
    );

    // Parameters
    parameter C_TERM_COUNT = 4'd15; // terminal count

    // Internal signals
    reg [3:0] count = 4'd0; // transaction position

    // Update on negative edge of sclk
    always @ (negedge sclk) begin
        if (cs_n == 1'b0) begin // transaction in progress
            if (count == C_TERM_COUNT) begin // reset transaction position
                count <= 1'b0;
            end else begin
                if (count >= 4'd3 && count <= 4'd10) begin // corresponding indicies
                    #40 sdo <= val[4'd7 - (count - 4'd3)]; // data access time delay,
                                                                then set sdo
                end else begin
                    #40 sdo <= 1'b0; // data access time delay, then set sdo
                end
                count <= count + 4'd1;
            end
        end else begin // transaction stopped
            count <= 1'b0;
            #40 sdo <= 1'b0; // data access time delay, then set sdo
        end
    end

endmodule
```

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: Worcester Polytechnic Institute
// Engineer:
//  Adam Grabowski
// Create Date: 09/23/2022 02:28:43 AM
// Design Name: ECE 3829 Lab 3
// Module Name: test_als_pmod_bfm
// Project Name: ECE 3829 Lab 3
// Target Devices: Basys 3 Artix-7 Development Board
// Tool Versions: Verilog 2021.1
// Description:
//  Specifies the testbench conditions to stimulate the als_pmod_bfm module
// Dependencies:
//  None
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//  None
//////////////////////////////////////////////////////////////////////////////////


module test_als_pmod_bfm(

    );

    // Parameters
    parameter C_SCLK_HALF_PERIOD = 200; // 400 ns period, 2.5 MHz frequency

    // UUT inputs and outputs
    reg sclk_2MHz; // clock input
    reg cs_n; // chip select
    reg [7:0] expected_value; // next value sent
    wire sdo; // data output

    // Internal wires
    reg [7:0] actual_value; // sampled from sdo
    integer i; // keeps track of transaction

    // Generate sclk
    always begin
        #20; // data access time delay
        #C_SCLK_HALF_PERIOD sclk_2MHz = ~sclk_2MHz; // toggle sclk
    end

    // Start simulation
    initial begin
        // Set time format and initial signals
        $timeformat(-9, 3, " ns", 3);
        sclk_2MHz = 1'b0;
        cs_n = 1'b1;
        expected_value = 8'hf5;
        actual_value = 8'd0;
        i = 0;

        // Repeat for 4 consecutive transactions
        repeat (4) begin
            // Keep cs_n high for 4 sclk cycles
            repeat(4) begin
                @ (negedge sclk_2MHz);
            end
            cs_n = 1'b0;
```

```
            // Collect sdo data over 16 sclk cycles
            for (i = 0; i < 16; i = i + 1) begin
                @ (negedge sclk_2MHz); // wait for negative edge
                if (i >= 3 && i <= 10) begin
                    actual_value[7 - (i - 3)] = sdo; // record sdo
                end
            end
            i = 0;

            // Compare actual and expected values
            if (expected_value == actual_value) begin
                $display("PASS: Expected: %b, Actual: %b", expected_value,
                    actual_value);
            end else begin
                $display("FAIL: Expected: %b, Actual: %b", expected_value,
                    actual_value);
            end

            // display time and reset signals
            $display("TIME: %t", $realtime);
            cs_n = 1'b1;
            expected_value = expected_value << 1;
            actual_value = 8'd0;
        end

        // Keep cs_n high for 4 sclk cycles
        repeat(4) begin
            @ (negedge sclk_2MHz);
        end
        $stop;
    end

    // Instantiate sensor module UUT
    als_pmod_bfm uuti (.sclk(sclk_2MHz), .cs_n(cs_n), .val(expected_value),
.sdo(sdo));

endmodule
```