Spring 2021

California State University, Northridge Department of Electrical & Computer Engineering

Experiment 10
Serial Protocol

May 13, 2021

ECE 526L

Written By: Jose Luis Martinez

Introduction

In this lab we are to design a serial protocol module shown in **Fig. 1** that is able to convert serialized data into parallel and parallel data into serialized data. We can think of this module as two separate modules doing one of the tasks mentioned beforehand each. The serial to parallel portion of the module will use the inputs i_clk, i_data, i_strobe, and i_rst to gather the serialized data. The outputs would be D_out and valid to output the now parallel data. The parallel portion of the module will use sys_clk, D_in, and send to gather the parallel data. The outputs will be o_clk, o_data, and o_strobe to output the now serialized data.

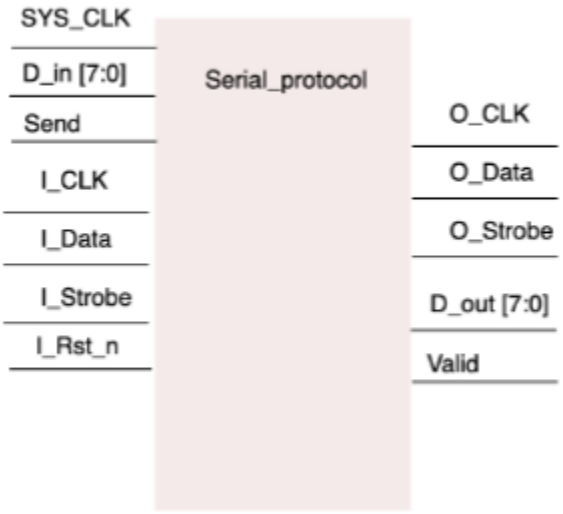

Fig. 1 Serial Protocol Model

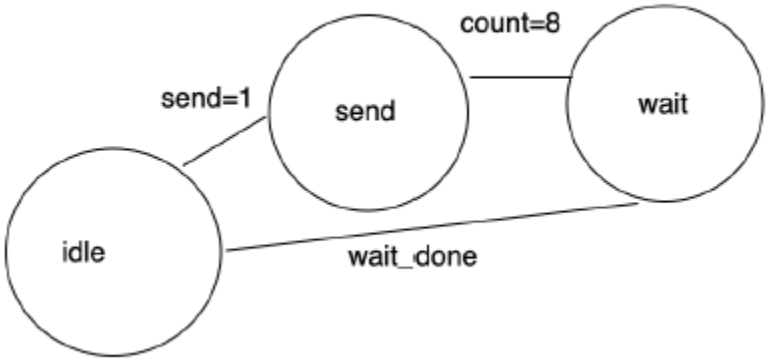

Fig. 2 Parallel to Serial FSM Diagram

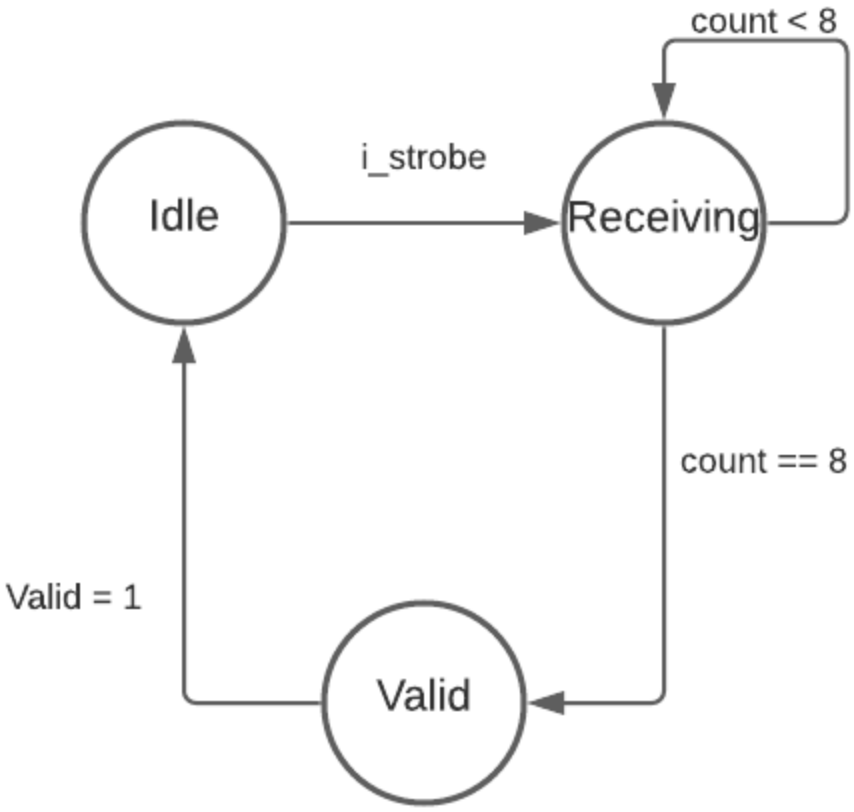

Fig. 3 Serial to Parallel Diagram

Methodology

I followed these steps in order to complete the lab:

1. The serial_to_parallel.v verilog file shown in **Fig. 4** was created with the fsm in **Fig. 2** used as the model.
2. The parallel_to_serial.v verilog file shown in **Fig. 5** was created with the fsm in **Fig. 3** used as the model.
3. The serial_protocol.v verilog file was created to combine both modules into one as shown in **Fig. 6**.
4. The lab10_tb was created as shown in **Fig. 7** to test out the serial_protocol.v to ensure that the module is working as intended.
    a. For the tb file I connected the outputs of the parallel_to_serial to the serial_to_parallel inputs.
5. The simv command was used to simulate the project and the output can be seen in **Fig. 8.**
6. The dve -full64 & command was run and we can see the waveforms generated in **Fig. 9.**

Results/Verilog Files

```
/**********************************************************************************
***                                                                             ***
*** ECE 526 L Experiment #10            Jose Luis Martinez, Spring, 2021      ***
***                                                                             ***
*** Experiment 10 - Serial Protocol                                            ***
***                                                                             ***
**********************************************************************************
*** Filename: serial_to_parallel.v Created by Jose Luis Martinez, May 2, 2021 ***
***                                                                             ***
**********************************************************************************/

`include "definitions.v"

module serial_to_parallel(sys_clk, i_clk, i_strobe, i_rst, i_data, D_out, valid);
	input sys_clk, i_clk, i_strobe, i_rst, i_data;
	output reg [7:0] D_out;
	output reg valid;

	reg [1:0] state, nextstate;
	reg [3:0] count;

	always@(state, count, i_strobe) begin //next state logic
		case (state)
			2'b00  : // idle state
				if(i_strobe) begin nextstate <= 2'b01; end
				else nextstate <= state;
			2'b01  : // receiving state
				if(count > 7) begin nextstate <= 2'b10; end
				else nextstate <= state;
			2'b10  : // valid state
				nextstate <= 2'b0;
			default: nextstate <= 2'b00;
		endcase
	end

	always @(posedge sys_clk, negedge i_rst) begin //current state logic
		if (!i_rst) begin
			state <= 2'b00;
		end else state <= nextstate;
	end

	// Output logic
	always@(state) begin
		case (state)
			2'b00  :
				begin
					valid <= 0;
			    end
			2'b01  :
				begin
					valid <= 0;
```

```
                        end
                2'b10  :
                        begin
                                valid <= 1;
                        end
        endcase
    end

    always@(negedge i_clk) begin
        if (state == 2'b01) begin
                D_out[count] <= i_data;
                count <= count + 1;
        end else begin
                count <= 3'b000;
        end
    end

endmodule
```

Fig. 4 serial_to_parallel.v

```
/**********************************************************************************
***                                                                             ***
*** ECE 526 L Experiment #10            Jose Luis Martinez, Spring, 2021      ***
***                                                                             ***
*** Experiment 10 - Serial Protocol                                            ***
***                                                                             ***
**********************************************************************************
*** Filename: serial_to_parallel.v Created by Jose Luis Martinez, May 2, 2021 ***
***                                                                             ***
**********************************************************************************/

`include "definitions.v"

module parallel_to_serial(sys_clk, D_in, send, o_clk, o_data, o_strobe, i_rst);
    input sys_clk, send, i_rst;
    input [7:0] D_in;
    output reg o_clk, o_strobe, o_data;

    reg [1:0] state, nextstate;
    reg [3:0] count;
    reg [7:0] D_in_reg;

    initial o_clk <= 1'b0;

    always@(state, count, send) begin //next state logic
        case (state)
                2'b00  : // idle state
                if(send) begin nextstate <= 2'b01; end
                else nextstate <= state;
                2'b01  : // send state
                nextstate <= 2'b10;
```

```
                2'b10  : // wait state
                       if(count > 7) begin nextstate <= 2'b00; end
                       else nextstate <= state;

                default: nextstate <= 2'b00;
        endcase
    end

    always @(posedge sys_clk or i_rst) begin //current state logic
        o_clk <= ~o_clk;
        if (!i_rst) begin
                state <= 2'b00;
        end else state <= nextstate;
    end

    // Output logic
    always@(posedge o_clk, state) begin
        case (state)
                2'b00  :
                       begin
                              o_data <= 8'b0;
                              o_strobe <= 1'b0;
                      end
                2'b01  :
                       begin
                              o_strobe <= 1'b0;
                              D_in_reg <= D_in;
                              o_data <= 8'b0;
                      end
                2'b10  :
                       begin
                              o_strobe <= 1'b1;
                       end
        endcase
    end

    always@(posedge o_clk) begin
        if(state == 2'b10) begin
                o_data <= D_in_reg[count];
                count <= count + 1;
        end else count <= 0;
    end

endmodule
```

Fig. 5 parallel_to_serial.v

```
/*****************************************************************************
***                                                                        ***
*** ECE 526 L Experiment #10            Jose Luis Martinez, Spring, 2021    ***
***                                                                        ***
*** Experiment 10 - Serial Protocol                                        ***
```

```
***                                                                          ***
*******************************************************************************
*** Filename: serial_protocol.v  Created by Jose Luis Martinez, May 2, 2021 ***
***                                                                          ***
*******************************************************************************/

`include "definitions.v"

module serial_protocol(sys_clk, D_in, send, i_clk, i_data, i_strobe,
                                    i_rst, o_clk, o_data, o_strobe, D_out, valid);

      input sys_clk, send, i_clk, i_data, i_strobe, i_rst;
      input [7:0] D_in;
      output o_strobe, valid;
      output o_clk, o_data;
      output [7:0] D_out;

      serial_to_parallel stb1( .sys_clk(sys_clk), .i_clk(i_clk), .i_strobe(i_strobe),
.i_rst(i_rst),
                              .i_data(i_data), .D_out(D_out), .valid(valid));
      parallel_to_serial bts1(.sys_clk(sys_clk), .i_rst(i_rst), .D_in(D_in), .send(send),
                             .o_clk(o_clk), .o_data(o_data), .o_strobe(o_strobe));
endmodule
```

Fig. 6 serial_protocol.v

```
/*******************************************************************************
***                                                                          ***
*** ECE 526 L Experiment #10            Jose Luis Martinez, Spring, 2021    ***
***                                                                          ***
*** Experiment 10 - Serial Protocol                                         ***
***                                                                          ***
*******************************************************************************
*** Filename: lab10_tb.v         Created by Jose Luis Martinez, May 2, 2021 ***
***                                                                          ***
*******************************************************************************/

module lab10_tb();
      reg sys_clk, send, i_rst;
      reg [7:0] D_in;

      wire o_strobe, valid;
      wire o_clk, o_data;
      wire [7:0] D_out;

      serial_protocol sp1(.sys_clk(sys_clk), .D_in(D_in), .send(send), .i_clk(o_clk),
.i_data(o_data),
                             .i_strobe(o_strobe), .i_rst(i_rst), .o_clk(o_clk), .o_data(o_data),
.o_strobe(o_strobe),
                             .D_out(D_out), .valid(valid));

      initial begin
```

```
		sys_clk <= 0;
		forever begin
			#(`CLK_PER/2) sys_clk <= ~sys_clk;
		end
	end

	initial begin
		$vcdpluson;
		send <= 0;
		i_rst <= 1;
		D_in <= 8'hfa;
		#(`CLK_PER)
		send <= 1'b1;
		$strobe(`STRING, $time, sys_clk, send, i_rst,
		                   D_in, o_strobe, valid, o_clk, o_data, D_out);
		#(`CLK_PER)
		send <= 1'b0;
		$strobe(`STRING, $time, sys_clk, send, i_rst,
		                   D_in, o_strobe, valid, o_clk, o_data, D_out);
		#(`CLK_PER)

		$strobe(`STRING, $time, sys_clk, send, i_rst,
		                   D_in, o_strobe, valid, o_clk, o_data, D_out);
		#(`CLK_PER)

		$strobe(`STRING, $time, sys_clk, send, i_rst,
		                   D_in, o_strobe, valid, o_clk, o_data, D_out);
		#(`CLK_PER)

		$strobe(`STRING, $time, sys_clk, send, i_rst,
		                   D_in, o_strobe, valid, o_clk, o_data, D_out);
		#(`CLK_PER)

		$strobe(`STRING, $time, sys_clk, send, i_rst,
		                   D_in, o_strobe, valid, o_clk, o_data, D_out);
		#(`CLK_PER)

		$strobe(`STRING, $time, sys_clk, send, i_rst,
		                   D_in, o_strobe, valid, o_clk, o_data, D_out);
		#(`CLK_PER)

		$strobe(`STRING, $time, sys_clk, send, i_rst,
		                   D_in, o_strobe, valid, o_clk, o_data, D_out);
		#(`CLK_PER)

		$strobe(`STRING, $time, sys_clk, send, i_rst,
		                   D_in, o_strobe, valid, o_clk, o_data, D_out);

		#(`CLK_PER)
		$strobe(`STRING, $time, sys_clk, send, i_rst,
		                   D_in, o_strobe, valid, o_clk, o_data, D_out);
		#(`CLK_PER)
```

```
$strobe(`STRING, $time, sys_clk, send, i_rst,
                   D_in, o_strobe, valid, o_clk, o_data, D_out);
#(`CLK_PER)
$strobe(`STRING, $time, sys_clk, send, i_rst,
                   D_in, o_strobe, valid, o_clk, o_data, D_out);
#(`CLK_PER)
$strobe(`STRING, $time, sys_clk, send, i_rst,
                   D_in, o_strobe, valid, o_clk, o_data, D_out);
#(`CLK_PER)
$strobe(`STRING, $time, sys_clk, send, i_rst,
                   D_in, o_strobe, valid, o_clk, o_data, D_out);
#(`CLK_PER)
$strobe(`STRING, $time, sys_clk, send, i_rst,
                   D_in, o_strobe, valid, o_clk, o_data, D_out);
#(`CLK_PER)
$strobe(`STRING, $time, sys_clk, send, i_rst,
                   D_in, o_strobe, valid, o_clk, o_data, D_out);
#(`CLK_PER)
$strobe(`STRING, $time, sys_clk, send, i_rst,
                   D_in, o_strobe, valid, o_clk, o_data, D_out);
#(`CLK_PER)
$strobe(`STRING, $time, sys_clk, send, i_rst,
                   D_in, o_strobe, valid, o_clk, o_data, D_out);
#(`CLK_PER)
$strobe(`STRING, $time, sys_clk, send, i_rst,
                   D_in, o_strobe, valid, o_clk, o_data, D_out);
#(`CLK_PER)
$strobe(`STRING, $time, sys_clk, send, i_rst,
                   D_in, o_strobe, valid, o_clk, o_data, D_out);
#(`CLK_PER)
$strobe(`STRING, $time, sys_clk, send, i_rst,
                   D_in, o_strobe, valid, o_clk, o_data, D_out);
#(`CLK_PER)
$strobe(`STRING, $time, sys_clk, send, i_rst,
                   D_in, o_strobe, valid, o_clk, o_data, D_out);
D_in <= 8'hd4;
#(`CLK_PER)
send <= 1'b1;

$strobe(`STRING, $time, sys_clk, send, i_rst,
                   D_in, o_strobe, valid, o_clk, o_data, D_out);
#(`CLK_PER)
send <= 1'b0;
#(`CLK_PER)
$strobe(`STRING, $time, sys_clk, send, i_rst,
                   D_in, o_strobe, valid, o_clk, o_data, D_out);
#(`CLK_PER)
$strobe(`STRING, $time, sys_clk, send, i_rst,
                   D_in, o_strobe, valid, o_clk, o_data, D_out);
      #(`CLK_PER)
$strobe(`STRING, $time, sys_clk, send, i_rst,
                   D_in, o_strobe, valid, o_clk, o_data, D_out);
```

```
        #(`CLK_PER)
        $strobe(`STRING, $time, sys_clk, send, i_rst,
                           D_in, o_strobe, valid, o_clk, o_data, D_out);
        #(`CLK_PER)
        $strobe(`STRING, $time, sys_clk, send, i_rst,
                           D_in, o_strobe, valid, o_clk, o_data, D_out);
        #(`CLK_PER)
        $strobe(`STRING, $time, sys_clk, send, i_rst,
                           D_in, o_strobe, valid, o_clk, o_data, D_out);
        #(`CLK_PER)
        $strobe(`STRING, $time, sys_clk, send, i_rst,
                           D_in, o_strobe, valid, o_clk, o_data, D_out);
        #(`CLK_PER)
        $strobe(`STRING, $time, sys_clk, send, i_rst,
                           D_in, o_strobe, valid, o_clk, o_data, D_out);
        #(`CLK_PER)
        $strobe(`STRING, $time, sys_clk, send, i_rst,
                           D_in, o_strobe, valid, o_clk, o_data, D_out);
        #(`CLK_PER)
        $strobe(`STRING, $time, sys_clk, send, i_rst,
                           D_in, o_strobe, valid, o_clk, o_data, D_out);
        #(`CLK_PER)
        $strobe(`STRING, $time, sys_clk, send, i_rst,
                           D_in, o_strobe, valid, o_clk, o_data, D_out);
        #(`CLK_PER)
        $strobe(`STRING, $time, sys_clk, send, i_rst,
                           D_in, o_strobe, valid, o_clk, o_data, D_out);
        #(`CLK_PER)
        $strobe(`STRING, $time, sys_clk, send, i_rst,
                           D_in, o_strobe, valid, o_clk, o_data, D_out);
        #(`CLK_PER)
        $strobe(`STRING, $time, sys_clk, send, i_rst,
                           D_in, o_strobe, valid, o_clk, o_data, D_out);
        #(`CLK_PER)

        $strobe(`STRING, $time, sys_clk, send, i_rst,
                           D_in, o_strobe, valid, o_clk, o_data, D_out);
        #(`CLK_PER)

        $strobe(`STRING, $time, sys_clk, send, i_rst,
                           D_in, o_strobe, valid, o_clk, o_data, D_out);
        #(`CLK_PER)

        $strobe(`STRING, $time, sys_clk, send, i_rst,
                           D_in, o_strobe, valid, o_clk, o_data, D_out);
        #(`CLK_PER)

        $strobe(`STRING, $time, sys_clk, send, i_rst,
                           D_in, o_strobe, valid, o_clk, o_data, D_out);
        #(`CLK_PER)
        $finish;
    end
```

```
endmodule
```

Fig. 7 lab10_tb.v

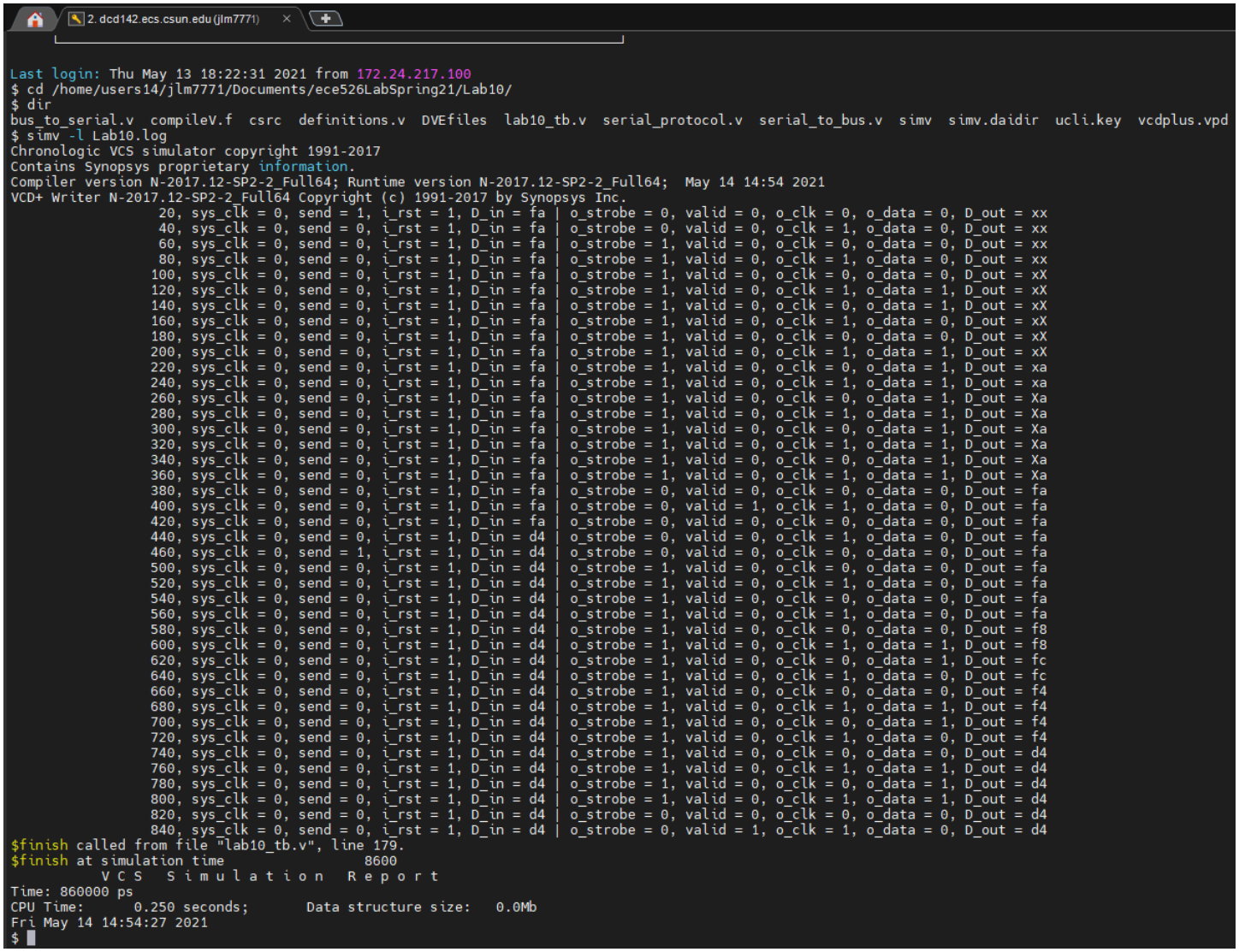

Fig. 8 simv output

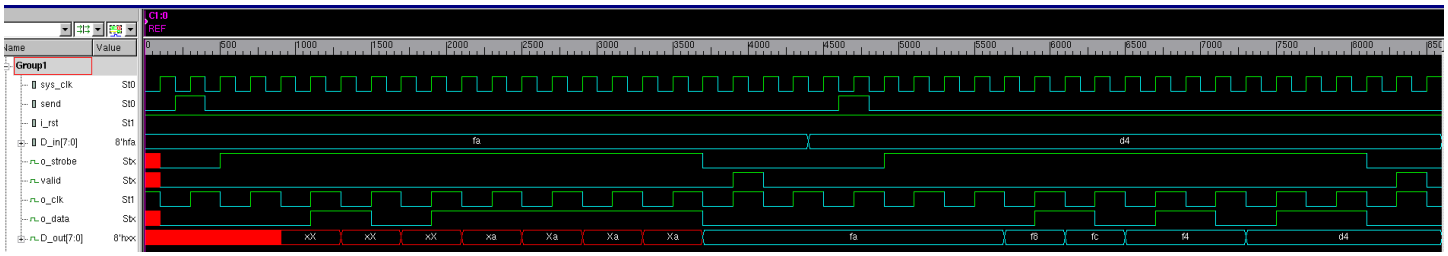

Fig. 9 Waveforms

Analysis

For the test bench I sent two values to the parallel_to_serial portion of the serial_protocol module and if everything is working properly then we should be able to see the value again at the output of the serial_to_parallel portion. So the first value I sent was 8'hfa. As you can see from **Fig. 8 & 9,** we are able to see 8'fa on D_out. The next value is 8'hd4, and similarly to the previous attempt we are able to see in **Fig. 8 & 9,** that 8'hd4 on D_out as well.

Conclusion

In conclusion I was able to implement the serial_protocol by separating each component into their own modules. I used an FSM for each component and then combined them in the top level module. In the test bench file I connected the output of parallel_to_serial to the inputs of the serial_to_parallel to ensure that whatever i sent to the parallel input i got out in the parallel output. As we can see from the results I was able to correctly complete the task. I learned how to implement a serial protocol using FSM.

Academic Dishonesty

Submitting any report that is not entirely your own work is a form of academic dishonesty and will not be tolerated. Each and every lab report must include the following statement, signed and dated by the student. Lab reports without the statement will be summarily rejected.

I hereby attest that this lab report is entirely my own work. I have not copied either code or text from anyone, nor have I allowed or will I allow anyone to copy my work.

Name (printed) Jose L Martinez

Name (signed) Jose Martinez Date 5/14/2021