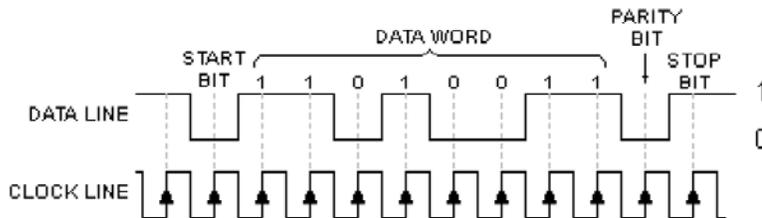


## CSE 2021 Computer Organization Quiz #2 (Fall 2011)

1. The RS232 interface is an older serial data transfer method. Data to be sent to a receiving circuit is provided in the following format:



The parity bit may have 5 different settings including even, odd, mark, space or none. The above is an example of odd parity. That is the parity bit is set (asserted or set to '1') so that the total number of '1's (not including the start and stop bits) to be transmitted is an odd number (in the above example the data word contains 5 set bits or '1's so the parity bit is a zero so that total number of '1's remains an odd number).

The user may also set the number of data bit (the example is 8) and the number of stop bit (in the above there is 1 stop bit). So the RS232 settings are 8O1 - 8 data bits, Odd parity and 1 stop bit.

Write a Verilog module (next sheet) that takes in a data word, and outputs the RS232 which includes the additional start, stop and parity bit corresponding the 8E1 (8 data bits, Even parity and 1 stop bit). The following is a testbench module to test your code:

```

module testbench;
    reg [7:0] test_word;
    reg [10:0] out_word

    rs2328e1 uut_rs2328e1(test_word, out_word);

    initial
    begin
        #200
        test_word = 8'b10100100;
        #1 $display($time, " in = %b, 8e1_out = %b ",test_word, out_word);
        #200
        test_a = 8'b00111100;
        #2 $display($time, " in = %b, 8e1_out = %b ",test_word, out_word);
    end
endmodule

```

Name: \_\_\_\_\_

Student #: \_\_\_\_\_



(a) Insert your code here:

```
module
```

```
end module
```

(b) What is the expected output from `testbench` when it and your module from part (a) are compiled and run using `iverilog` and `vvp` commands?

Name: \_\_\_\_\_

Student #: \_\_\_\_\_



2. Consider the instruction `sll` (shift left logical) with the following format:

| opcode |    | rs |    | rt |    | rd |    | shamt |   | funct |   |
|--------|----|----|----|----|----|----|----|-------|---|-------|---|
| 31     | 26 | 25 | 21 | 20 | 16 | 15 | 11 | 10    | 6 | 5     | 0 |

that executes  $RF[rd] = RF[rs] \ll \text{shamt}$  (note this is slightly different than the implementation shown on the MIPS card).

Assume that the ALU design has been modified so that it is capable of shifting the operand applied at the upper input *A* by the number of bits specified at the lower input *B* when  $ALUOp[1..0]$  is 11. A 4-input multiplexer has been added and is shown for the ALU input *B* - the control for this multiplexer is  $ALUSrcB[1..0]$

- a. Add the necessary circuitry to the single cycle datapath of Fig. 2 (Next sheet) to implement the `sll` instruction. Highlight the datapath for the instruction.
- b. Fill in Fig. 1 to add the necessary control signals needed for the `sll` instruction.

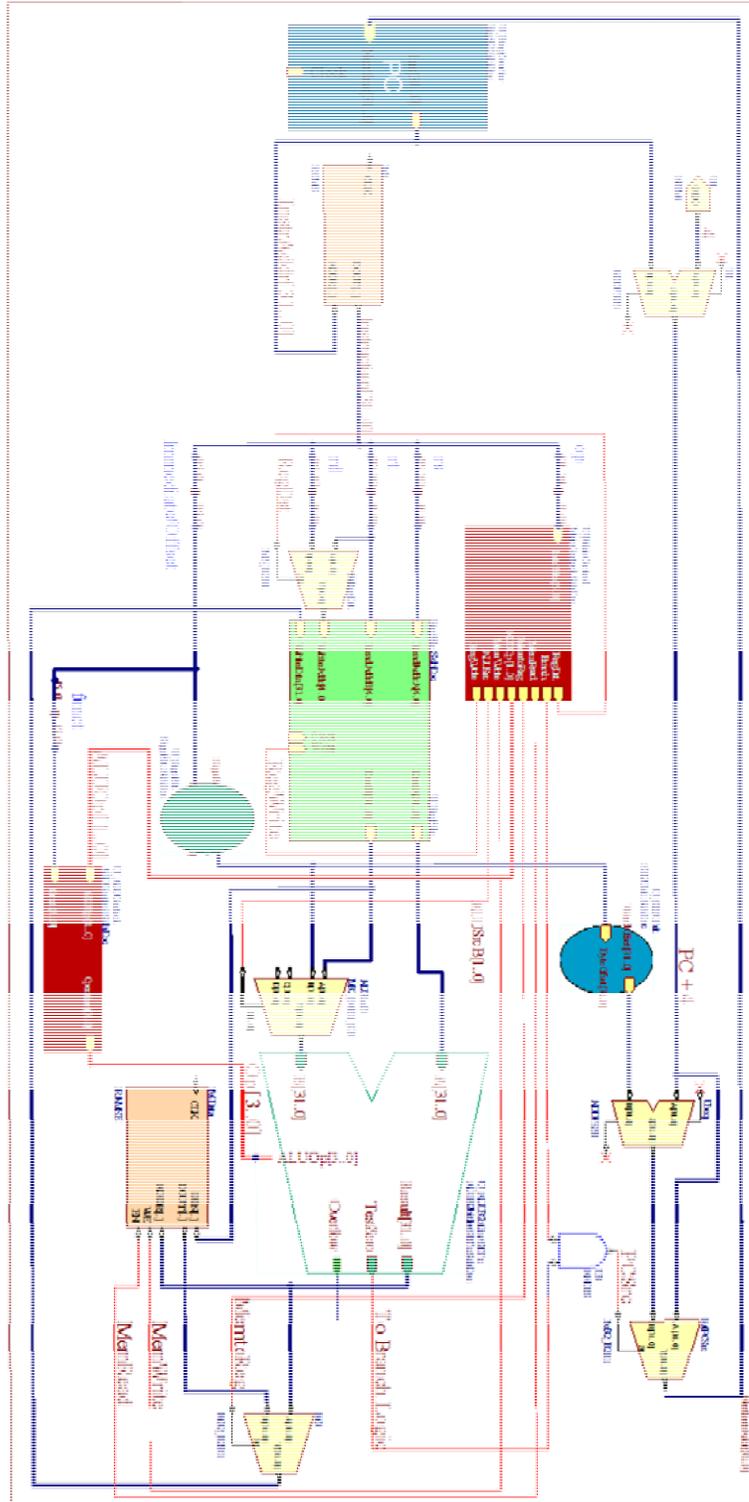
Fig. 1: Control Signals for `sll` in a single cycle implementation.

| Instruction      | RegDst | ALUSrcB[1..0] | MemtoReg | RegWrite | MemRead | MemWrite | Branch | ALUOp[1..0] |
|------------------|--------|---------------|----------|----------|---------|----------|--------|-------------|
| R-format         | 1      | 00            | 0        | 1        | 0       | 0        | 0      | 10          |
| <code>sll</code> |        |               |          |          |         |          |        |             |

Fig .2: Datapath for R-type, `lw`, `sw`, and `beq` instructions. Add additional hardware, if required, to implement the datapath for the instruction specified in the problem statement. Also, highlight the datapath for the instruction. (Next Sheet)

Name: \_\_\_\_\_

Student #: \_\_\_\_\_



**CSE 2021 Computer Organization Quiz #2 (Fall 2011)****Answer Section****PROBLEM**

1. ANS:

(a) A verilog routine would look like:

```
module rs2328e1(data, out_data);
    input wire[7:0] data;
    output wire [10:0] out_data;

    // internal signal declaration
    wire p;

    //parity bit
    assign p = data[7]^data[6]^data[5]^data[4]^data[3]^data[2]^data[1]^data[0];
    // product terms
    assign out_data[10] = 1'b0;
    assign out_data[9:2] = data[7:0];
    assign out_data[1] = p;
    assign out_data[0]=1'b1;

endmodule
```

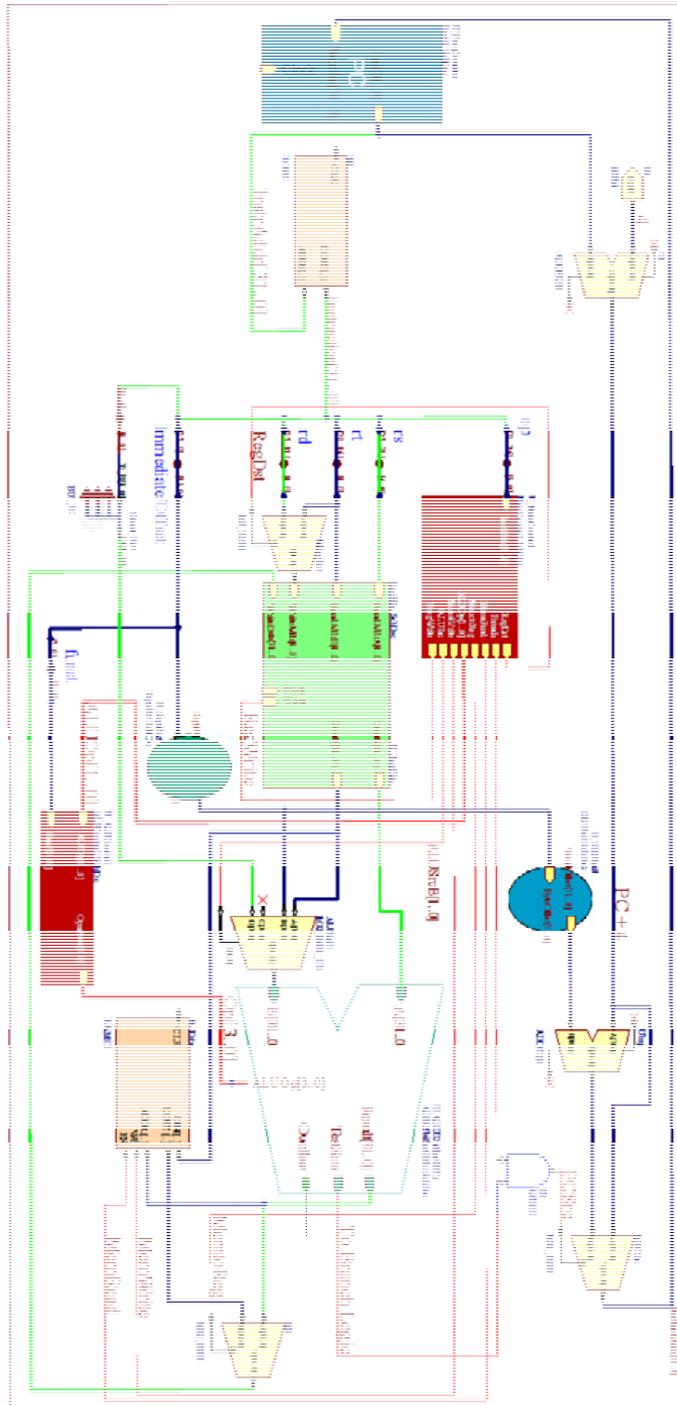
(b) The output would look like:

```
C:\iverilog>vvp a.out
      201 in = 10100100, 8e1_out = 01010010011
      403 in = 00111100, 8e1_out = 00011110001
```

PTS: 1

2. ANS:

(a) The updated single cycle datapath should look something like as shown below (green line) - “shamt” field (instruction [10:6]) from the instruction has been added (with a bus joiner adding zeroes to the left to make it 32 bits wide as an input to the 4-input multiplexer.



(b) The instruction is in R-format and is controlled according to the first line in the table below. No changes in the table are needed.

| Instruction | RegDst | ALUSrc B[1..0] | MemtoReg | RegWrite | MemRead | MemWrite | Branch | ALUOp[1..0] |
|-------------|--------|----------------|----------|----------|---------|----------|--------|-------------|
|             |        |                |          |          |         |          |        |             |

|     |   |    |   |   |   |   |   |    |
|-----|---|----|---|---|---|---|---|----|
| sll | 1 | 11 | 0 | 1 | 0 | 0 | 0 | 10 |
|-----|---|----|---|---|---|---|---|----|

PTS: 1