

University of California at Berkeley
College of Engineering
Department of Electrical Engineering and Computer Sciences

EECS151/251A
Spring 2024

J. Wawrzynek
3/12/24

Exam 1

Name: _____

Student ID number: _____

Class (EECS151 or EECS251A): _____

Before solving the problems, write your student ID number on all pages.

You have 170 minutes to take the exam.

For each problem if you find yourself taking excessive time to work out a solution consider skipping the problem or a fresh approach. Also, start by answering the easier questions and then move on to the more difficult ones.

No calculators, phones, or other devices allowed.

Neatness counts. Make your answers neat and clear—particularly circuit diagrams. *Also, remember to use "solder dots" on three-way connections.* We will deduct points if we need to work hard to understand your answer.

Work out your answers on scrap paper then neatly copy them into the space provided for them.

1 Pareto Optimality [5 pts]

John is exploring the space of circuits implementations for his design and discovers that Pareto optimal designs all lie on a curve described by $AT^2 = k$, where A is the area, T is the minimal clock period, and k is a constant. He has a particular Pareto optimal design with an area of $100\mu m^2$ and runs at 1 GHz. He finds a way to increase the speed to 2 GHz, but the area increases to $500\mu m^2$. Is this new design also Pareto optimal? Show your work.

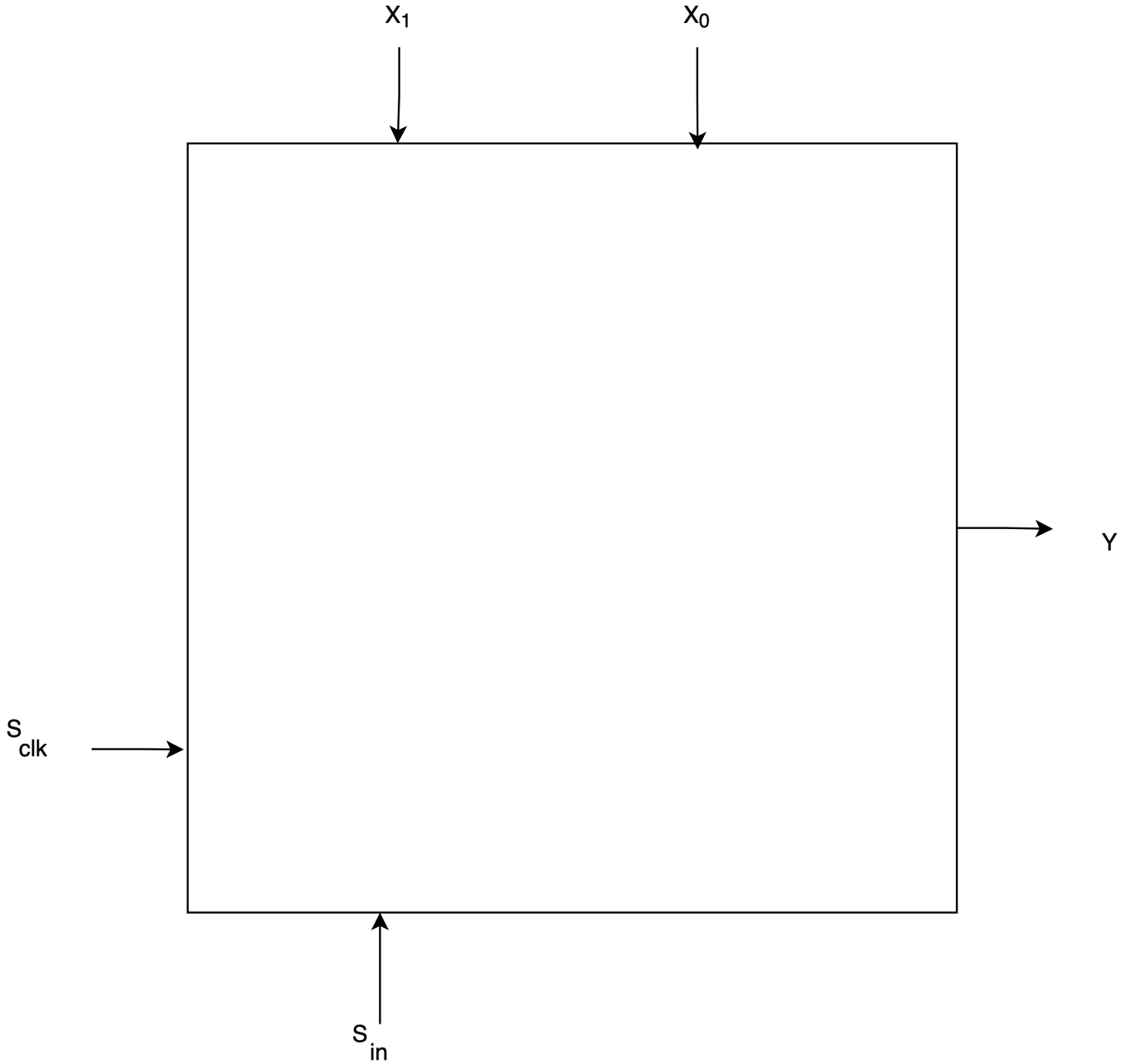
2 Chips Costs [5pts]

You are charged with designing an ASIC for your company. You spend \$2M on NRE costs and then the foundry charges you \$10 per die. You buy 100,000 chips worth of wafers and package each chip for \$1 each and then test the packaged parts for \$1 each. The results of the testing indicates that the final yield is only 40%. What is the resulting cost to your company of each of your working chips?

Given your knowledge of ASIC design, fabrication, and testing, list two ways you could have saved money.

3 FPGA LUT Circuit [12pts]

1. In the box below and using the indicated port signals, complete the drawing of the internal circuitry of a 2-LUT, using FFs, 2-to-1 multiplexors, and simple logic gates as needed. S_{in} is a bit-serial data input port and S_{clk} is a clock signal to be used for configuring the LUT function. x_1 , x_0 are the LUT data inputs and y is the LUT output.
2. How many distinct logic functions can this LUT implement?

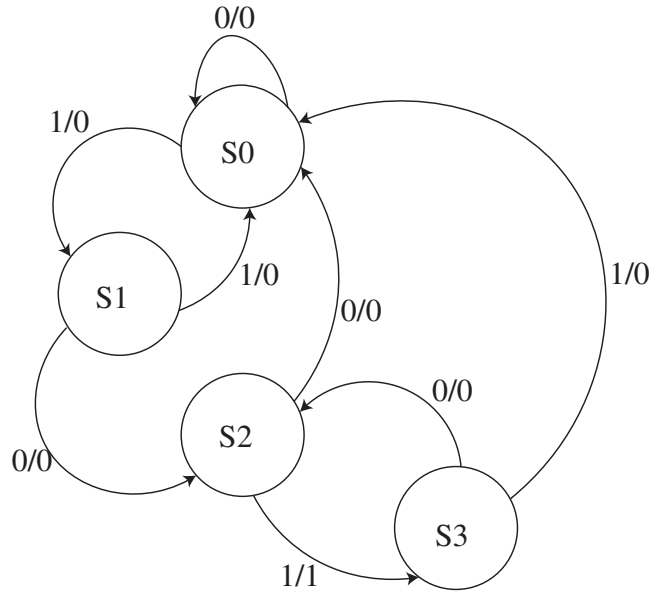


5 Combinational Logic Design [10pts]

Recall that Binary Coded Decimal (BCD) is a number representation that uses the binary encodings 0000–1001 to represent the 10 decimal digits. Derive two combinational logic functions that each accept a BCD digit and outputs a 1 iff that digit is evenly divisible by 3. (Yes, 0 is divisible by 3.) One expression should be in SOP form and the other is POS, and both should be minimized.

6 Finite State Machine [25pts]

Below is the STD for an FSM with a single input, in, and a single output, out. The reset (initial) state is S0.



Based on the STD:

1. In words, briefly describe the function of the FSM:

2. Write the *behavioral* Verilog description using a case statement. We have provided the initial module definition. Note that our code includes the EECS151 library, and interfaces for the register generators are shown at the end of the exam.

```
`include "EECS151.v";

module FSM(in, out, clk, rst);
    input in, clk, rst;
    output out;

    reg [1:0] next_state
    wire [1:0] state;
    reg output;

    REGISTER state_register #(2) (.q(state), .d(next_state),
    .clk(clk), .rst(rst));

    localparam S0 = 2'b00,
    localparam S1 = 2'b01,
    localparam S2 = 2'b10,
    localparam S3 = 2'b11;

    always @(*) begin
        next_state = S0;
        output = 1'b0;
        case (current_state)
            S0: begin
                if(in) next_state = S1
                else next_state = S0
            end
            S1: begin
                if(in) next_state = S0
                else next_state = S2
            end
            S2: begin
                output = in;
                if(in) next_state = S3
                else next_state = S0
            end
            S3: begin
                if(in) next_state = S0
                else next_state = S2
            end
        endcase
    end
    assign out = output;
endmodule
```


Student ID number:

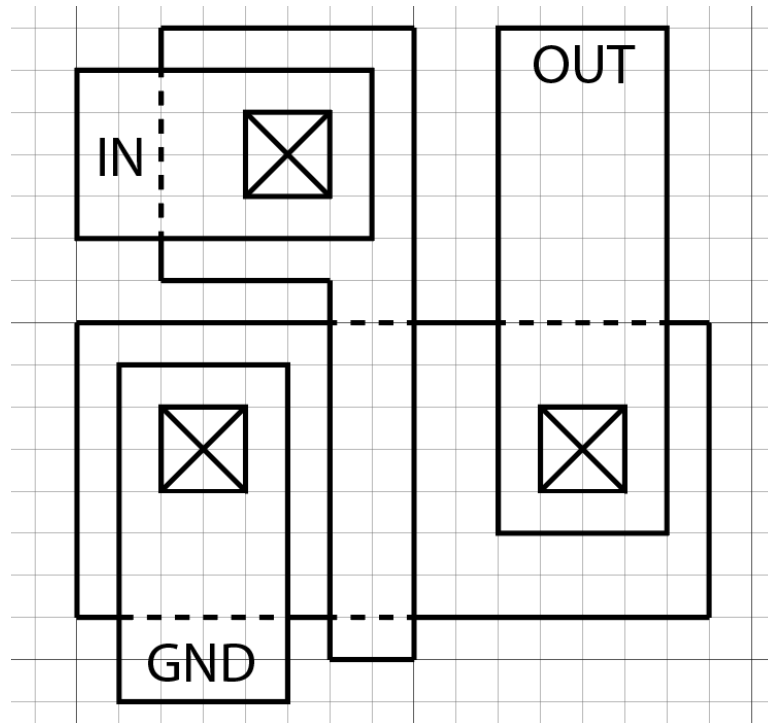
3. Draw the circuit diagram for its implementation using 1-hot encoding. Use only FFs, ANDs, ORs (of any number of inputs), and bubbles for inversion. You don't need to connect the clk, and reset signal to the FFs, but do need to indicate the reset values. Draw your circuit in this box:

4. Derive the next state and output logic for an *binary encoded* version with the following state assignments: S0=00, S1=01, S2=11 S3=10. Leave your answer in *unoptimized* SOP form.

7 Transistor Layout [5pts]

The figure below shows the layout of a transistor used as a pulldown in an inverter. Count and report the total number of unit squares for each of the following:

1. The transistor gate region:
2. The source diffusion-region:
3. The drain diffusion-region:



8 Static CMOS gate [5pts]

Draw the transistor level circuit diagram for a static CMOS gate that implements $y = c + \bar{a}b$ using the least number of transistors possible. You may assume you have inputs available in both complemented and uncomplemented form.

9 Tri-state Buffers [10pts]

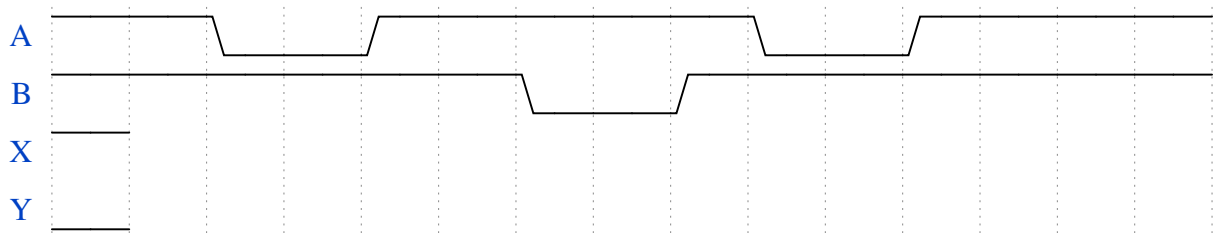
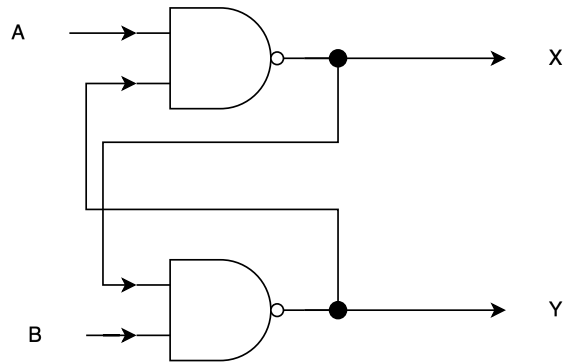
1. For some application you need a tri-state buffer, but you would like to combine it with a NAND function. Draw a transistor level circuit that would achieve both functions with the minimal number of transistors. Label the inputs and outputs.

Student ID number:

2. Supposed you are asked to design a 4-to-1 multiplexor using inverting tri-state buffers, inverters, and no other logic elements. Your multiplexor must have a non-inverting output. Neatly draw the circuit diagram. Label the data inputs, a, b, c, and d, the output as y, and the select controls as s0 and s1.

10 Logic Gates [5pts]

For the circuit shown below, complete the waveforms for the signal nodes, X and Y.

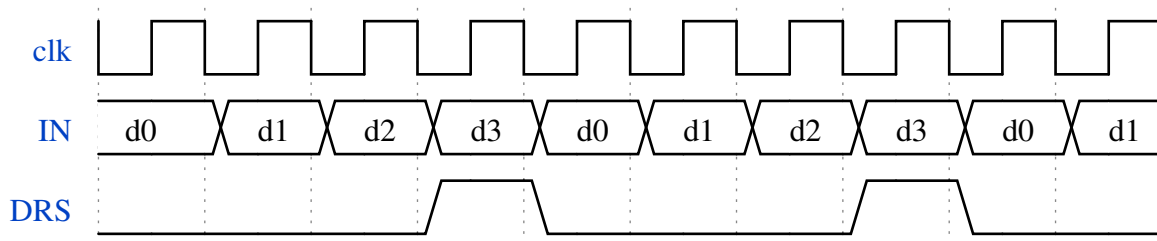


11 Circuit Design [12pts]

In lecture we presented a parallel-to-serial converter. It could be used, for instance, for sending words over a wire or a wireless link, one bit at a time. Your task here is to design a circuit for receiving the bits, a *serial-to-parallel converter*. You are tasked to design a circuit that adheres to the following specifications, using FFs, multiplexors, and simple logic gates as needed.

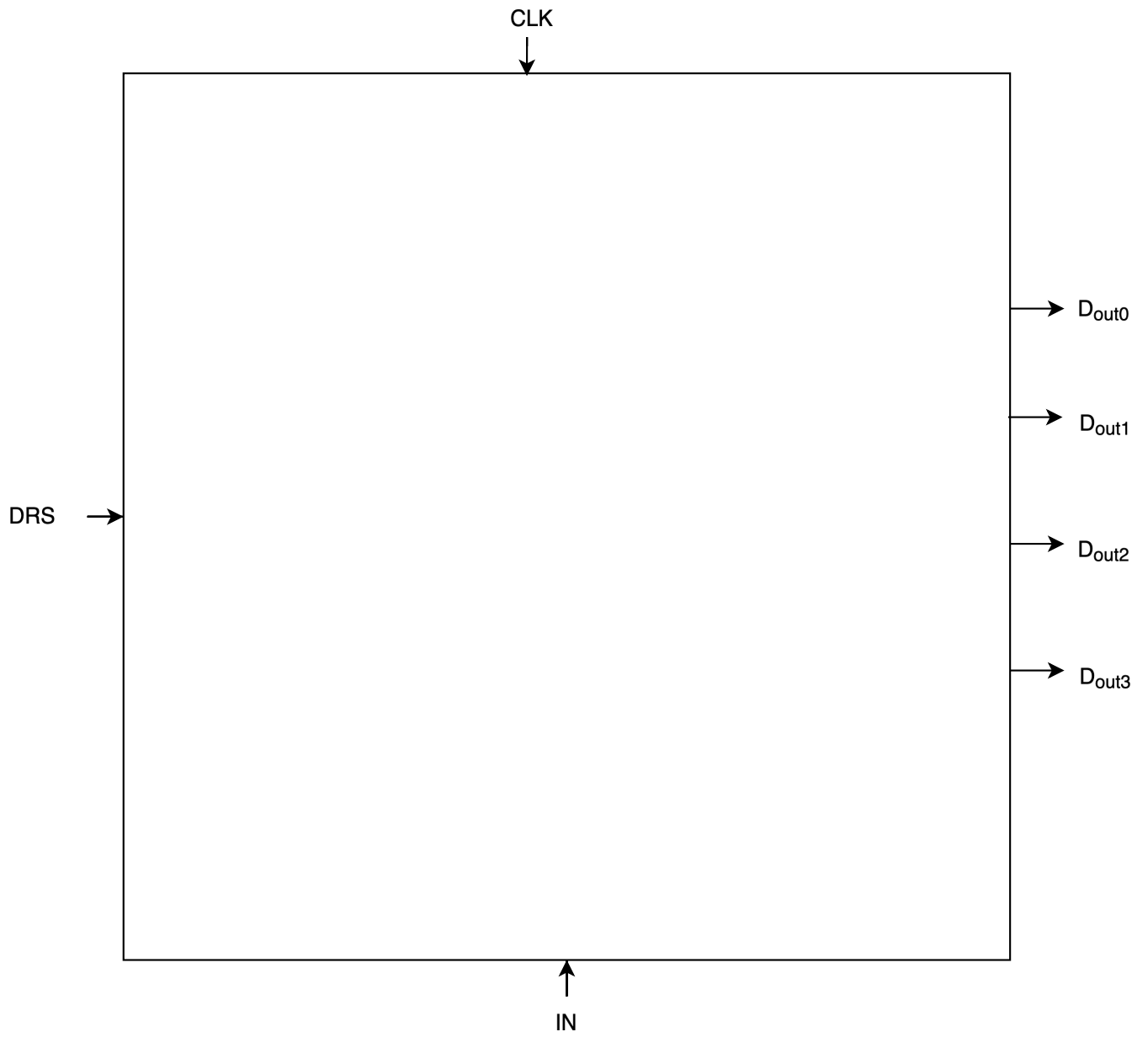
1. Your circuit will receive the bits of each word (4-bit words in this case), LSB first, one per clock cycle as shown below and must collect up the bits and present them to the external interface in word form.
2. The external interface supplies a “data request signal (DRS)” every four clock cycles that your circuit should use to provide the received bits to the interface. The output needs to remain stable until the next occurrence of the DRS signal.

The input waveform will look like the following:



Neatly draw your circuit in the box provided on the next page:

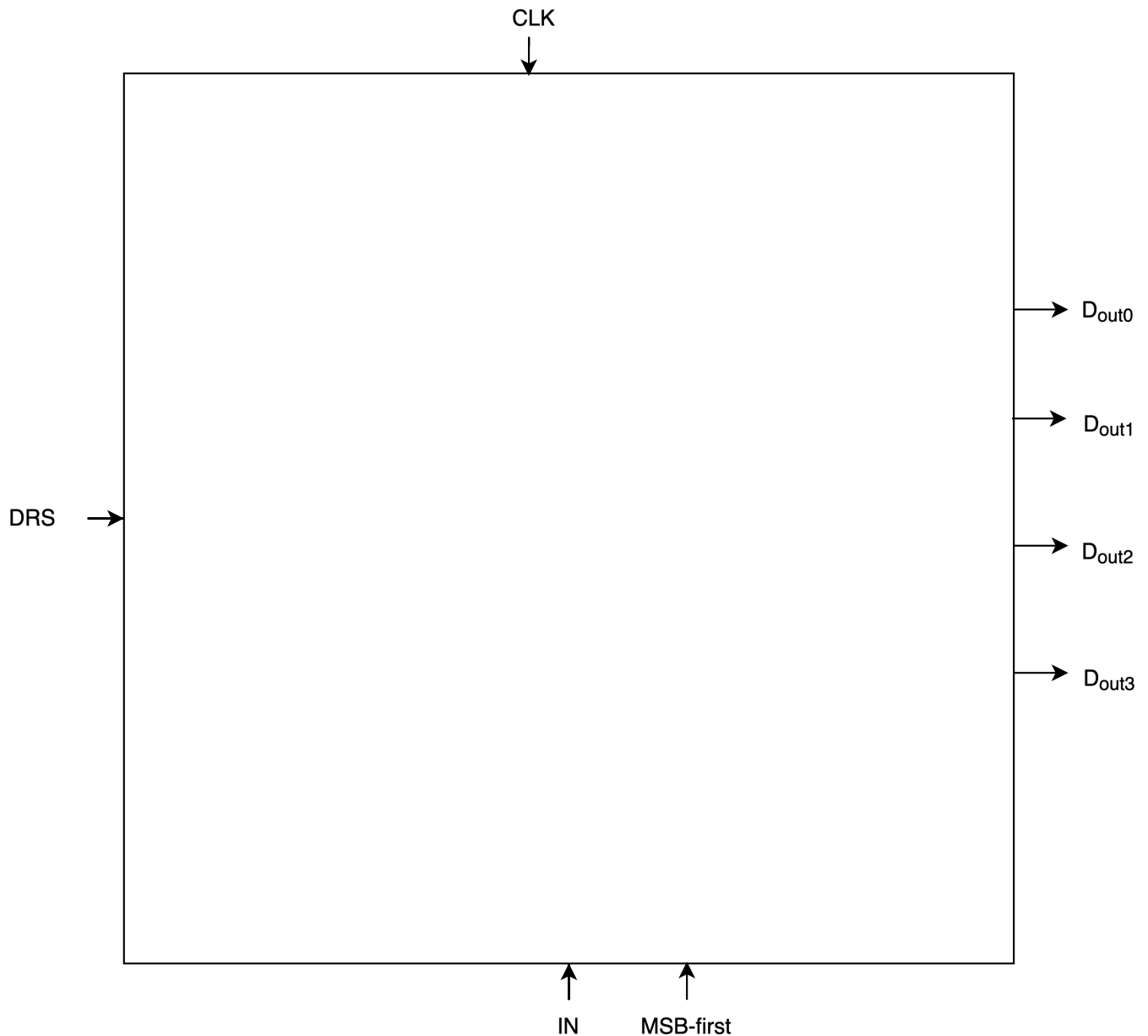
Student ID number:

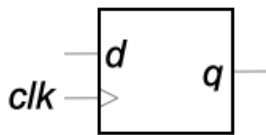


Student ID number: _____

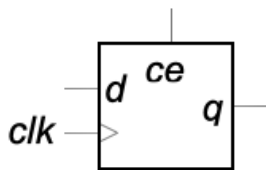
Now suppose, the external interface wants to be able to get the received word either MSB-first or LSB-first (big vs. little endian). The external circuit will additionally send an "MSB-first" signal every four clock cycles. If the MSB-first signal is high, then the first bit received in the stream (d0) should be considered the MSB, and the last bit (d3) the LSB. If the MSB-first signal is low, then the last bit received is the MSB and the first bit is the LSB.

Draw the updated circuit below. You may choose to abstract your answer from above and add additional circuitry, or to redraw it with modifications as needed.



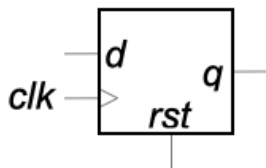


```
module REGISTER(q, d, clk);
parameter N = 1;
```



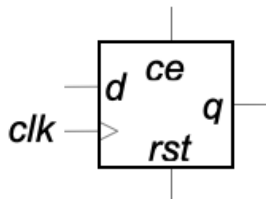
```
module REGISTER_CE(q, d, ce, clk);
parameter N = 1;
```

On the rising clock edge if clock enable (ce) is 0 then the register is disabled (it's state will not be changed).



```
module REGISTER_R(q, d, rst, clk);
parameter N = 1;
parameter INIT = {N{1'b0}};
```

On the rising clock edge if reset (rst) is 1 then the state is set to the value of INIT. Default INIT value is all 0's.



```
module REGISTER_R_CE(q, d, rst, ce, clk);
parameter N = 1;
parameter INIT = {N{1b'0}};
```

Reset (rst) has priority over clock enable (ce).

Student ID number:

Wednesday 27th March, 2024 03:42

Student ID number:

blank

Student ID number:

blank

Student ID number:

blank