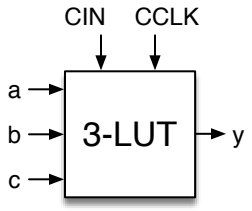
EECS 151/251A Homework 3

Due Monday, Feb 18th, 2019

Problem 1: LUT Implementation

Consider the design of a 3-input FPGA lookup table (LUT). An abstracted view of the 3-LUT is shown below. The ports, a, b, c, and y, are the data inputs and output, respectively. The input CIN stands for "configuration input", and CCLK stands for "configuration clock". The configuration clock is used at FPGA initialization time to shift in the LUT contents, one bit per clock cycle, over the CIN port.



(a) Draw a circuit diagram for the internal implementation of the 3-LUT, including configuration loading. Use only the following circuit components: Inverter, 2-input and, 2-input or, Flip-flop. Remember to label all inputs and outputs.

(b) Imagine now that the 3-LUT is programmed to implement the following function: `~a & b & c`. Label the appropriate nodes in your circuit diagram with the correct configuration values.

Problem 2: The "Uber" Flip-Flop

Write a Verilog module which implements a d-Flip-Flop with clock enable, synchronous reset, synchronous set, asynchronous preset, and asynchronous clear. Give priority to reset & clear. Give asynchronous priority over synchronous.

Synchronous Inputs:

- reset: q=0
- set: q=1

Asynchronous Inputs:

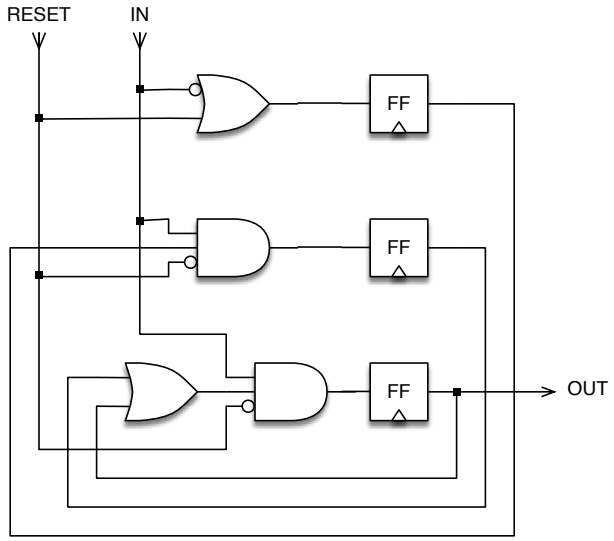
- preset: q=1
- clear: q=0

Note: The clock enable only effects synchronous inputs.

Problem 3: Verilog Circuit Implementation

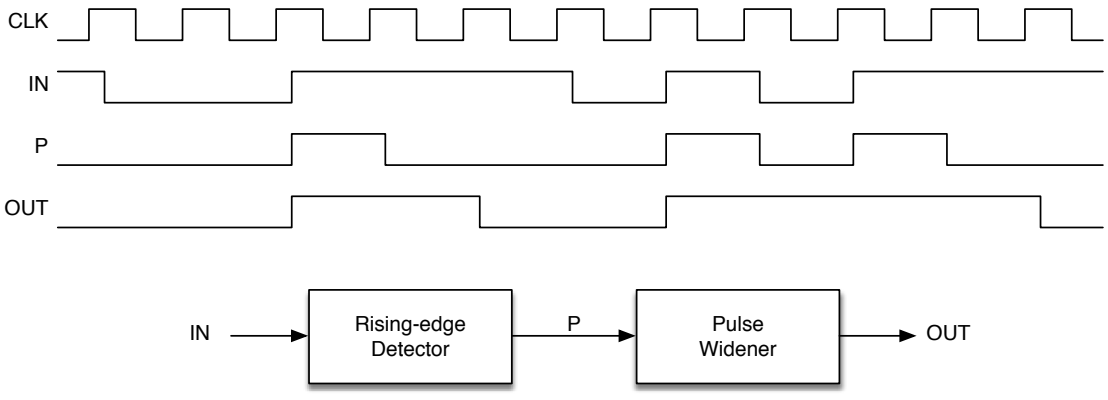
Write a Verilog description of the circuit shown below.

Note: A small circle drawn at an input port indicates that the input signal is inverted before passing to the gate.



Problem 4: Clocked Circuits

In this problem, you will design a simple synchronous circuit that outputs a pulse of width 2 clock cycles on each rising edge of the input signal. An example input and output signal is shown below. To make things easier, we will split the design into two parts. The "Rising-edge Detector" outputs a pulse in response to a rising edge on the input, as shown below. The "Pulse Widener" stretches out the pulse for 2 cycles.

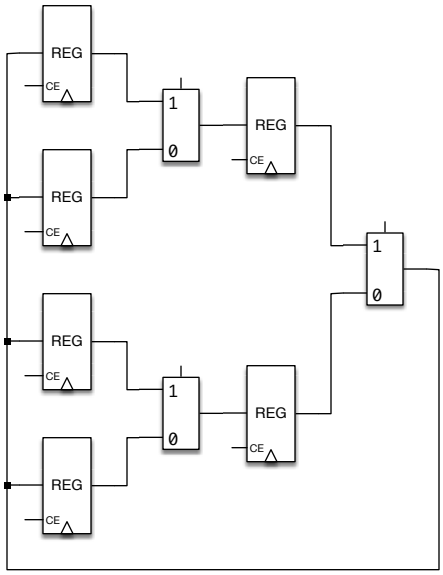


(a) Draw your circuit for the Rising-edge Detector. Use only the following circuit components: Inverter, 2-input and-gate, 2-input or-gate, flip-flop. Simpler designs are worth more points.

(b) Draw your circuit for the Pulse Widener.

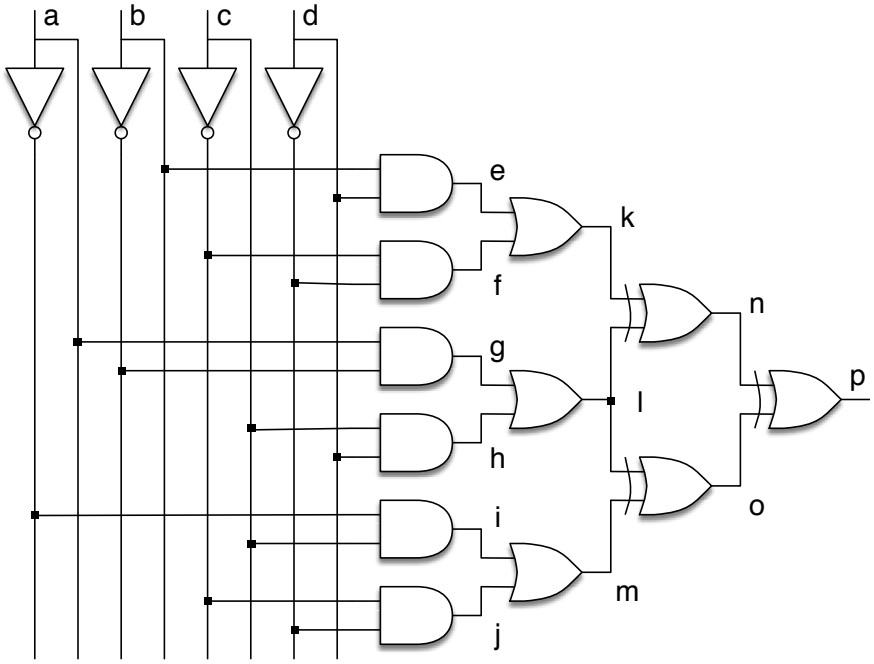
Problem 5: Register Transfers

Given the datapath below, what is the minimum number of clock cycles needed to exchange the top-left-most register value with the bottom-left-most register value?



Problem 6: FPGA Mapping

Using only 4-input lookup tables (LUTs), partition the circuit shown below into as few LUTs as possible. *Do not attempt to simplify the gate-level circuit before mapping it to LUTs.* Indicate your answer by drawing boundaries around your partitions.



Problem 7: 251A only — *Optional* Challenge Question for 151

Using only simple logic gates and flip-flops, derive a circuit whose output is a *square-wave* with 1/8 the frequency of the input clock frequency. Try to use as few flip-flops and gates as possible.

Hint: Start out by first designing a circuit that outputs a square-wave with 1/2 the clock frequency, then one with 1/4, etc.

Problem 8: LUT Mapping

Imagine you are given a "mystery" FPGA programmed to perform a particular set of functions. We know the following about the FPGA:

1. The device contains a collection of N-LUTs (the value of N is part of the mystery). The LUTs are numbered 0, 1, 2, ... Each LUT in the FPGA has the same number of inputs (same N).
2. Each LUT has an output labeled y_i, where i is the LUT number, and inputs labeled x_{i_j}, where i is the LUT number and j is the input number.
3. For programming, the LUTs are connected in a shift register. They are programmed with a configuration bit-stream shifted in from one LUT to the next.
4. LUT 0 is programmed as an inverter, therefore $y_0 = NOT(x_{0\_0})$. Futhermore, LUT 0 is the only LUT programmed as an inverter.
5. We were able to recover a fragment of the bit stream, shown here: 0xE9AC96017F88FF55.

(a) How many LUTs would the above bit stream program?

(b) For each of the LUTs, draw a circuit using simple logic gates to represent the function it is programmed to implement. Label all inputs and outputs. Show your work.

Problem 9: Interpreting Verilog

Neatly sketch the circuit that corresponds to the following Verilog code:

```
module foo (CE, X, CLK, RST, OUT);
   input CE, CLK, X, RST;
   output OUT;
   reg [3:0] Q1;
   reg [3:0] Q2;
   assign OUT = ^Q1;
   always @ (posedge CLK)
     if (RST) begin
        Q1 <= 4'h5;
        Q2 <= 0;
```

```
      end
      else
        if (CE) begin
           if (X) Q2 <= Q1<<1;
           else Q2 <= Q1;
           Q1 <= Q2;
        end
endmodule // foo
```