P116B Homework 4

Due 2/28/2020

1. In class, we discussed the concept of "data sparsification". Imagine we have a system with 50,000 channels, each with 8 bits of data. We have two options for storing this data

   - Store all the data we read out, regardless of the signal size.
   - Only store channels that exceed a predefined threshold, along with their address.

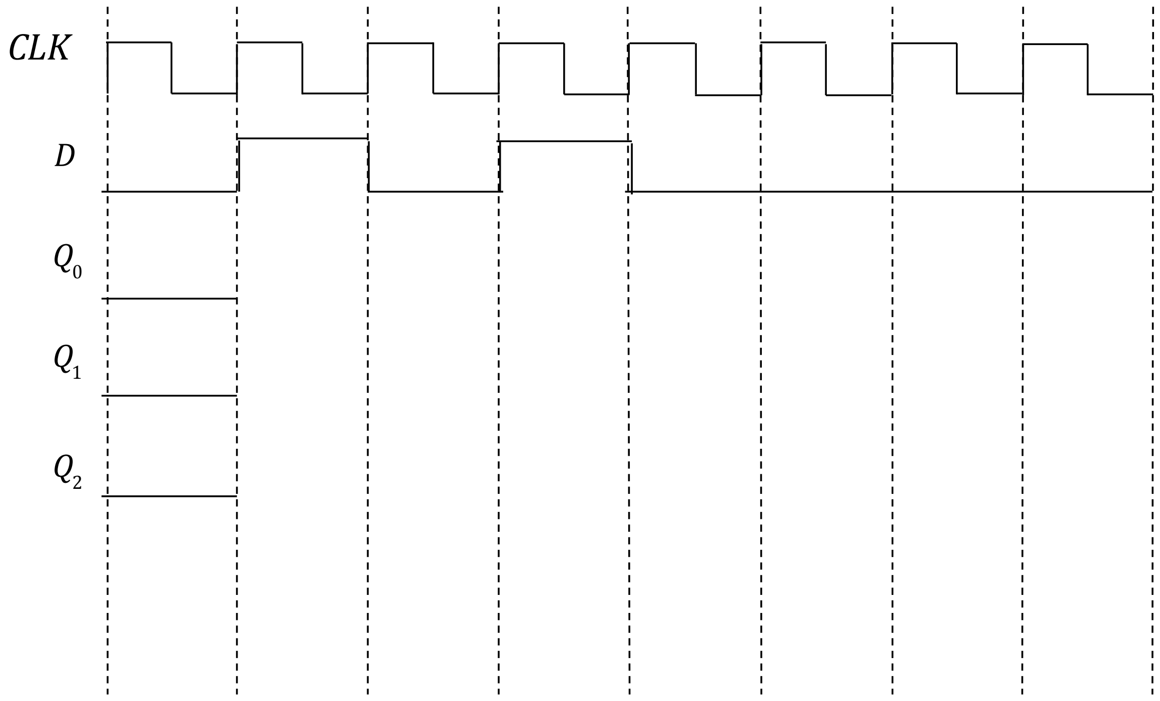
   (a) How many bits do I need to allocate for the address?

   (b) At what "occupancy" (percentage of channels exceeding threshold) does it take less space to store all the data (with no addresses) rather than the "sparsified" version?

2. For the following module

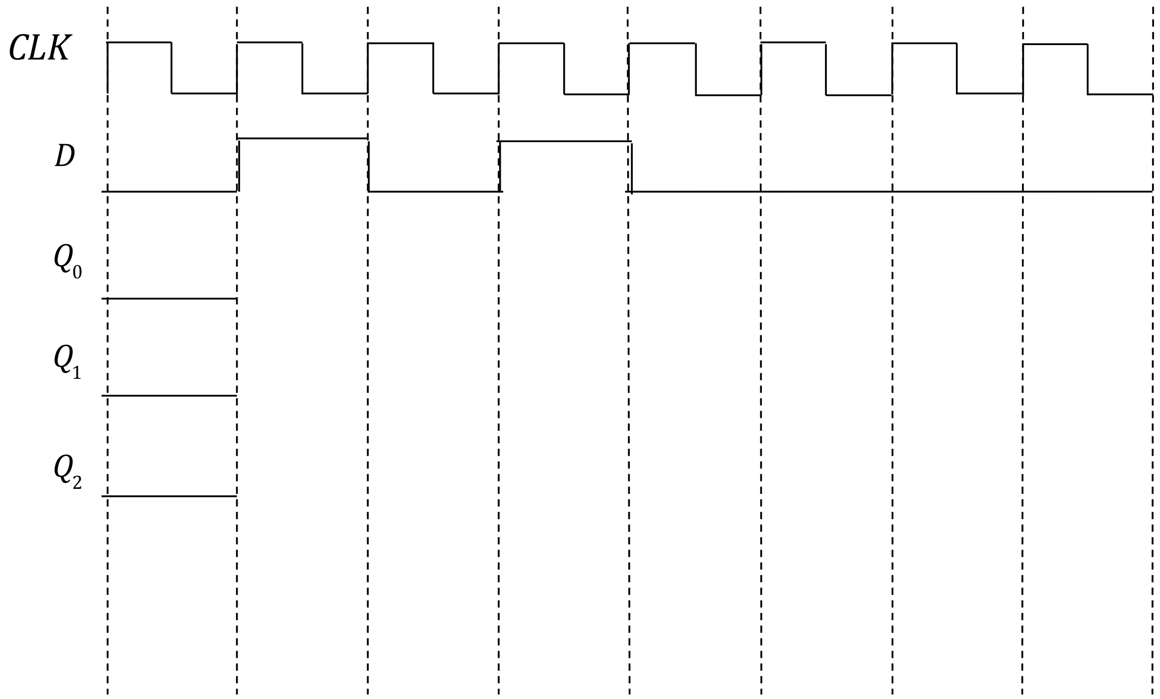
```
module hw4 (
  input clk,          // system clock
  input D,
  output reg Q0=0,
  output reg Q1=0,
  output reg Q2=0);    // Serial data bit

  always @(posedge clk) begin
    Q0 <= D;          // non-bloacking statements
    Q1 <= Q0|D;
    Q2 <= Q0&Q1;
  end

endmodule
```

   Complete the following time line, assuming $Q1$, $Q2$ and $Q3$ all start at 0, and the D input is synchronous.



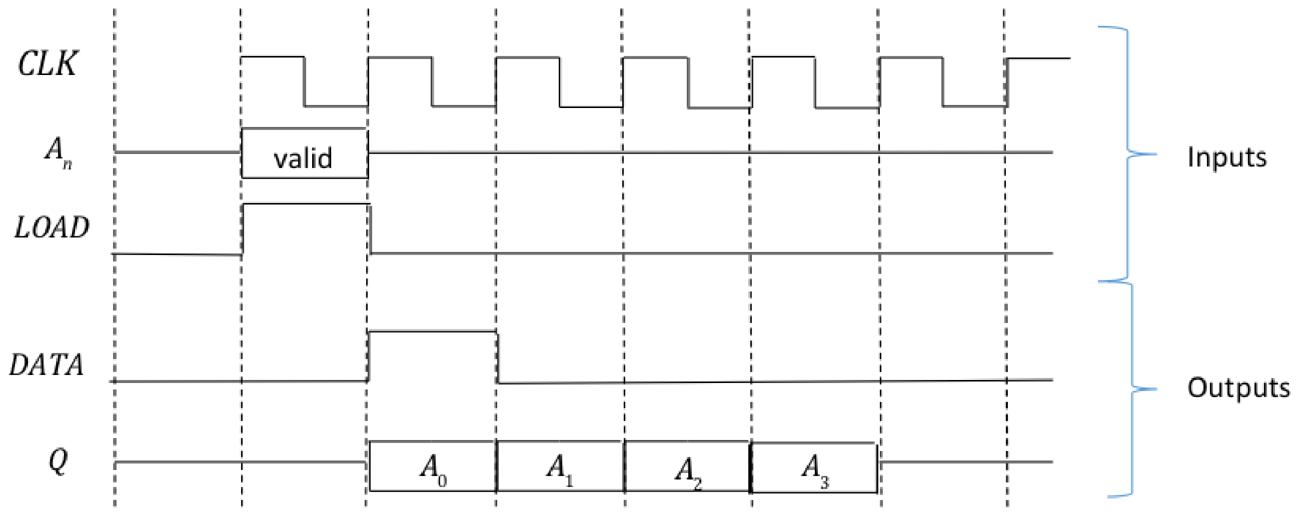
3. Repeat the previous problem if the non-blocking statements are replaced with blocking statements.

```
module hw4 (
  input clk,           // system clock
  input D,
  output reg Q0=0,
  output reg Q1=0,
  output reg Q2=0);    // Serial data bit

  always @(posedge clk) begin
    Q0 = D;          // bloacking statements
    Q1 = Q0|D;
    Q2 = Q0&Q1;
  end

endmodule
```



4. Redo the "parallel to serial converter" problem from a previous homework set, using Verilog code rather than discrete components.

   As a reminder, this is a circuit that will load a four bit word ($A[3:0]$) in response to a "LOAD" signal, and then clock the bits out serially to Q over the next four clock cycles. The circuit should also issue a DATA output concurrent with the first serial data bit, to signal the start of serial data, as illustrated in the timeline below



(traces between 0 and 1 indicate "don't care")

   As before, you may assume that the A_n bits are valid for at least one cycle, and the LOAD bit is synchronously asserted for exactly one clock cycle. Define your module as

```
module p2s (
  input clk,          // system clock
  input [3:0] A,      // Input parallel word
  input LOAD,         // synchronous load command
  output reg DATA=0,  // Serial data coming out
  output reg Q=0);    // Serial data bit
```

You will simulate your design in the next problem.

5. Use EDA Playground to simulate the module you wrote in the previous problem. Once you create a profile account in EDA Playground

   - Under "Tools & Simulations", select "Icarus Verilog 0.9.7".
   - Select the "Open **EPWave** after run" radio button.
   - Give your project a unique name in the title bar of the window at the bottom.

   Load your p2s module in the design.sv window, and dowload the file "ps2_testbench.v" from the "Files/FPGA Files" directory at the Canvas site and copy it into the testbench.sv window.

   Save your project and click "Run". Once you have things working, submit a screen shot of the output waveforms.