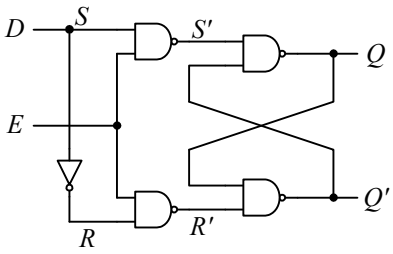
Lab 4: Sequential Circuits – Latch, Flip-Flip and Register

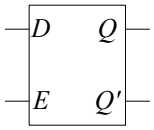
Purpose

In this lab you will learn about sequential circuits. Sequential circuits are memory circuits for storing binary values. You will design and implement the latch, the flip-flop and the register. These are standard memory circuits used in microprocessors for storing values similar to variables in a program.

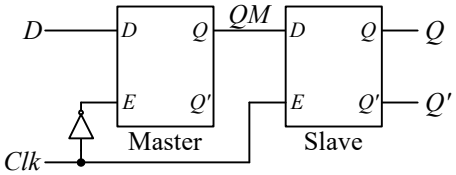
Procedure

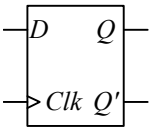
1. The **D latch with enable** is a simple storage element for storing one bit of data. When *E* (*Enable*) is a 1, the data input at *D* is stored at *Q*. In other words, the D latch stores data into *Q* as long as *E* is a 1, i.e. *Q* changes as long as *E* is a 1. The circuit and symbol are shown below. Draw the schematic circuit for a D latch with enable. Connect the inputs and outputs to appropriate switches and LEDs. Implement the circuit onto the FPGA board to test and verify that it works correctly.





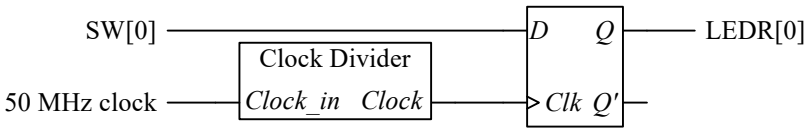
2. The **D flip-flop** stores data at every rising edge of the clock signal. In other words, it is synchronize to the clock. The circuit and symbol are shown below. Draw the schematic circuit for a D flip flop. Connect the inputs and outputs to appropriate switches and LEDs. Implement the circuit onto the FPGA board to test and verify that it works correctly. In order to test the circuit, you need to slow down the 50 MHz clock. The Verilog code for a clock divider to slow down a 50 MHz clock to 1 Hz is shown below. Connect the 50 MHz clock source (PIN_L1) to the input of the clock_divider circuit. Connect the output of the clock_divider to the *Clk* signal of the flip flop.



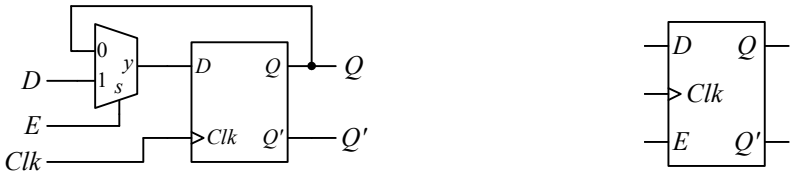



```
module clock_divider
#(parameter [24:0] half = 25'd25000000)  // 50M/2 = 1 Hz output clock
(
  input clock_in,          // 50MHz input clock
  output reg clock
  );

  reg [24:0] count;

  always @(posedge clock_in) begin
    if (count == half) begin
      count <= 25'd0;
      clock <= ~clock;
      end
```

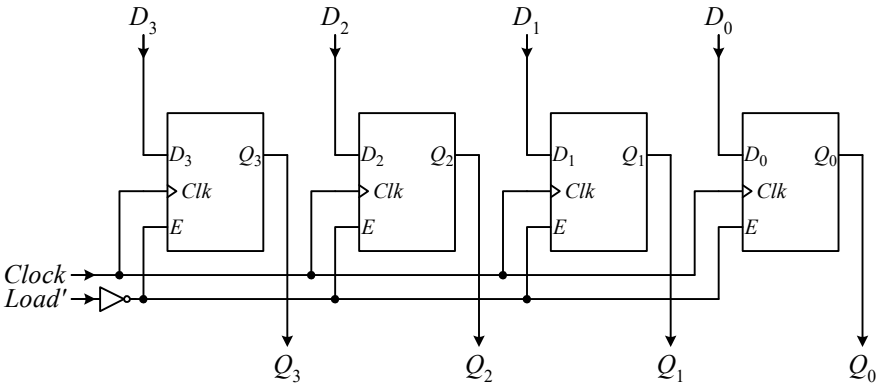
```
    else begin
      count <= count+1;
      end
  end
endmodule
```



3. Instead of using the D flip-flop where it will always store a new value at every rising clock edge, we want a **D flip-flop with enable** so that we can control when we want to store in a value. The circuit and symbol for the D flip-flop with enable is shown next. The other component in the circuit is a 1-bit 2-to-1 mux which you should already have from lab 1.



4. A **4-bit register** can store four bits of data together as a unit using four D flip-flop with enable circuits. The circuit and symbol for the 4-bit register are shown below.



Connect the inputs and outputs to appropriate switches and LEDs as follows:

Load′ to KEY[0]

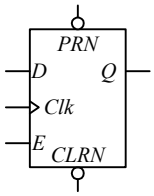
Clock to the 50 MHz clock

$D_3 - D_0$ to SW[3] – SW[0]

$Q_3 - Q_0$ to LEDR[3] – LEDR[0]

Implement the circuit onto the FPGA board to test and verify that it works correctly.

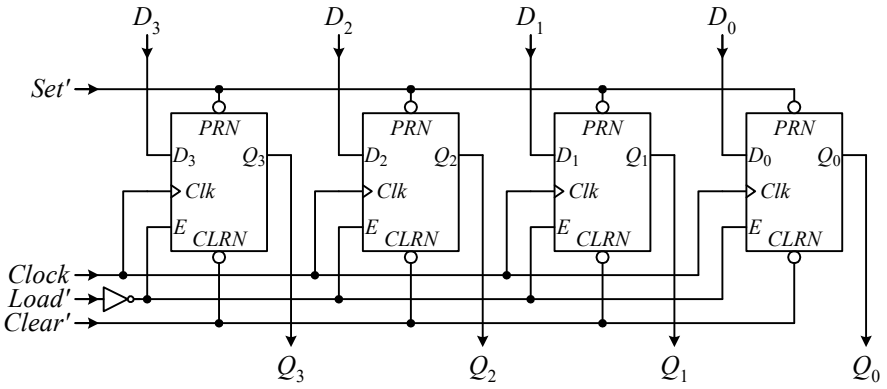
Alternatively, if instead of using the D flip-flop with enable that you created from #3 above, you can use the symbol from the Quartus library. It is called **dffe** and is under the storage folder. The dffe symbol from the library is



PRN stands for *preset'* which means a 0 signal will set *Q* to a 1.

CLRN stands for *clear'* which means a 0 signal will set *Q* to a 0.

If you don't want to use the *PRN* and *CLRN* signals, you need to connect them to VCC.



Connect and pin map the following:

Load' to KEY[0]

Set' to KEY[1]

Clear' to KEY[2]

Clock to the 50 MHz clock

$D_3 - D_0$ to SW[3] – SW[0]

$Q_3 - Q_0$ to LEDR[3] – LEDR[0]