ECSE 4770 LAB 1 REPORT

Yuchen Wang, Rin: 661960546, section 2

Introduction

- The primary object of this lab is to help students get familiar with the hardware design language, Verilog, and the method to use the FPGA board and its software, Quartus 15.
- In this lab, students need to build an 8-bit D-Flipflop, 4x4 multiplier, and a 16-8 MUX with Verilog. After testing the components' function by iVerilog, students will build components in Quartus and simulate them in Quartus.
- The top level of the lab contains the three components. The logic circuit should be able to get two 4-bit binary inputs and then multiply them together. The result of the multiplier should be saved in the 8-bit D-Flipflop. A push-button on the FPGA board should control the D-Flipflopd to save the result from the multiplier. The MUX should be able to choose between the multiplier result and the D-Flipflop saving data. Another push-button on the FPGA board will control the MUX. Output data will show in heximal form on the FPGA board by the given segments in the lab template.

Schematic

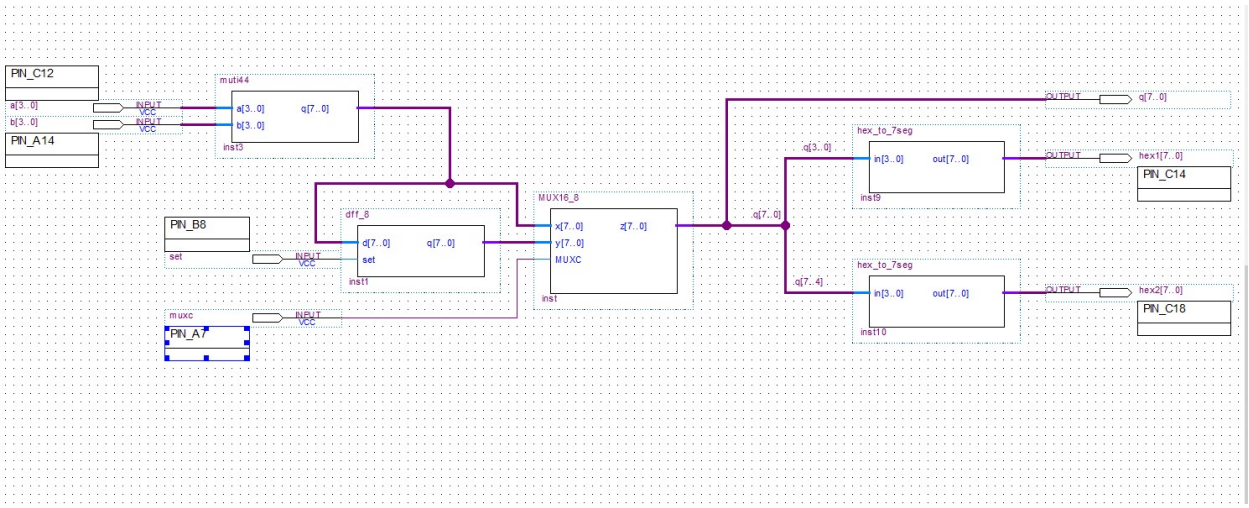

Figure 1: Top Level Circuit Schematic

Verilog Code

- **8-bit D-Flipflop**

```
module dff8(input [7:0]d, input set, output reg [7:0]q );
always @(posedge set) begin
    q <= d;
end
endmodule
```

This is the 8-bit D-Flipflop code. Input set is a 1-bit binary input that controls the flipflop. When the set has a positive edge, input d will be saved to output q. In Quartus, the input, set, connects to the push-button on the FPGA board.

- **4*4 Multiplier**

```
module muti44 (input [3:0]a, input [3:0]b, output [7:0]q);
    assign q = a*b;
endmodule
```

Verilog provides a straightforward way to fulfill the multiplier. Using the symbol of time can multiply the two inputs together.

- **16-8 MUX**

```
module MUX16_8 (input [7:0] x, input [7:0] y, input MUXC, output [7:0] z);
  assign z = MUXC?x:y;
endmodule
```

In the 16-8 MUX, conditional statement is used to fulfill the MUX function. For high input, MUXC, the output will be x. Otherwise, the output will be y.

Simulation Result

- **8-bit D-Flipflop**

```
wangy62@DESKTOP-OBTD29O:/mnt/c/Users/wangy62/LAB1$ iverilog -o myDFF DFF8_tb.v DFF8.v
wangy62@DESKTOP-OBTD29O:/mnt/c/Users/wangy62/LAB1$ vvp myDFF
VCD info: dumpfile dump.vcd opened for output.
Reset flop.
d:11010010, set:0, z:xxxxxxxx
trigger a rising edge.
d:11010010, set:1, z:11010010
change only the input
d:00000000, set:0, z:11010010
```

Figure 2: 8-bit D-Flipflop iVerilog Test Branch Result

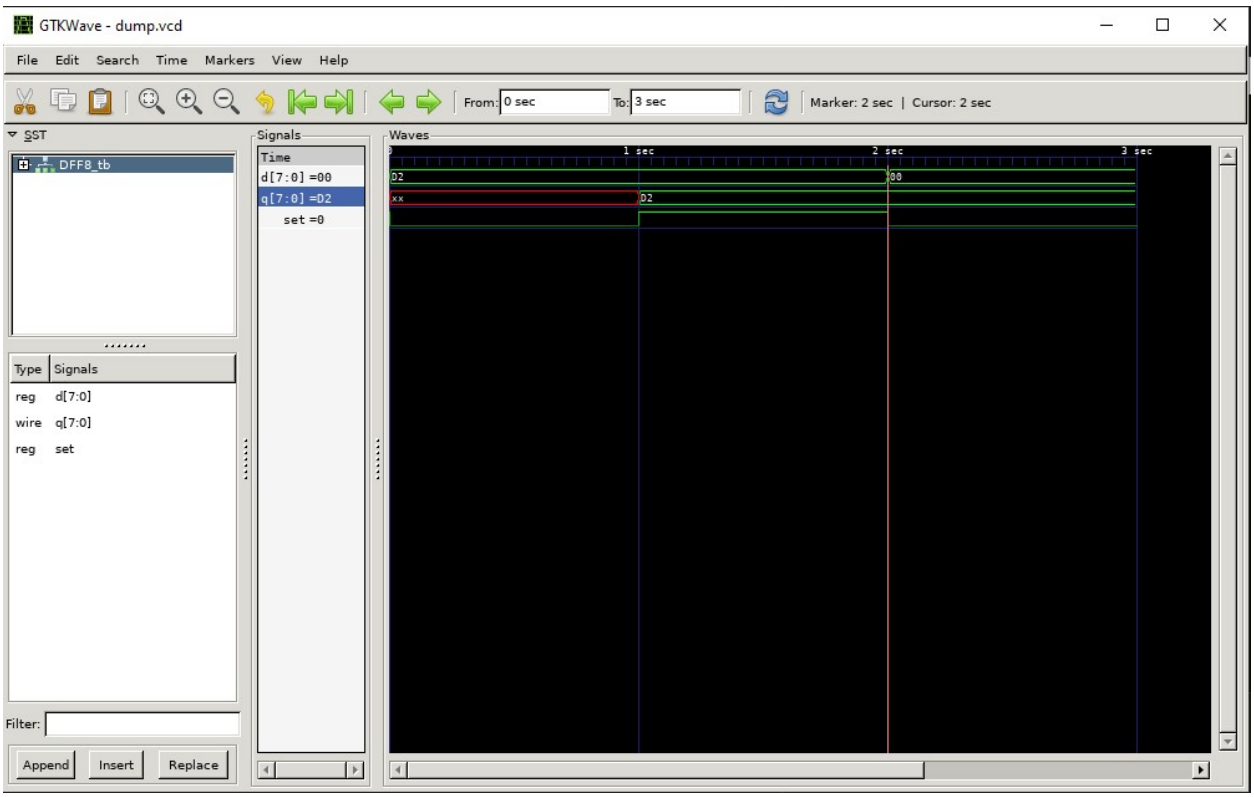

Figure 3: 8-bit D-Flipflop Wave Form Output

The flipflop can save data when the input, set, has a positive edge. And it stores the data correctly when the input changes.

- **4*4 Multiplier**

```
wangy62@DESKTOP-OBTD29O:/mnt/c/Users/wangy62/LAB1$ iverilog -o myDFF muti_tb.v muti44.v
wangy62@DESKTOP-OBTD29O:/mnt/c/Users/wangy62/LAB1$ vvp myDFF
VCD info: dumpfile dump.vcd opened for output.
test with x=12, y=5.
x:1100, y:0101, z:00111100
test with x=0, y=15.
x:0000, y:1111, z:00000000
test with x=6 y=3
x:0110, y:0011, z:00010010
test with x=15 y=15
x:1111, y:1111, z:11100001
```

Figure 4: 4*4 multiplier iVerilog Test Branch Result

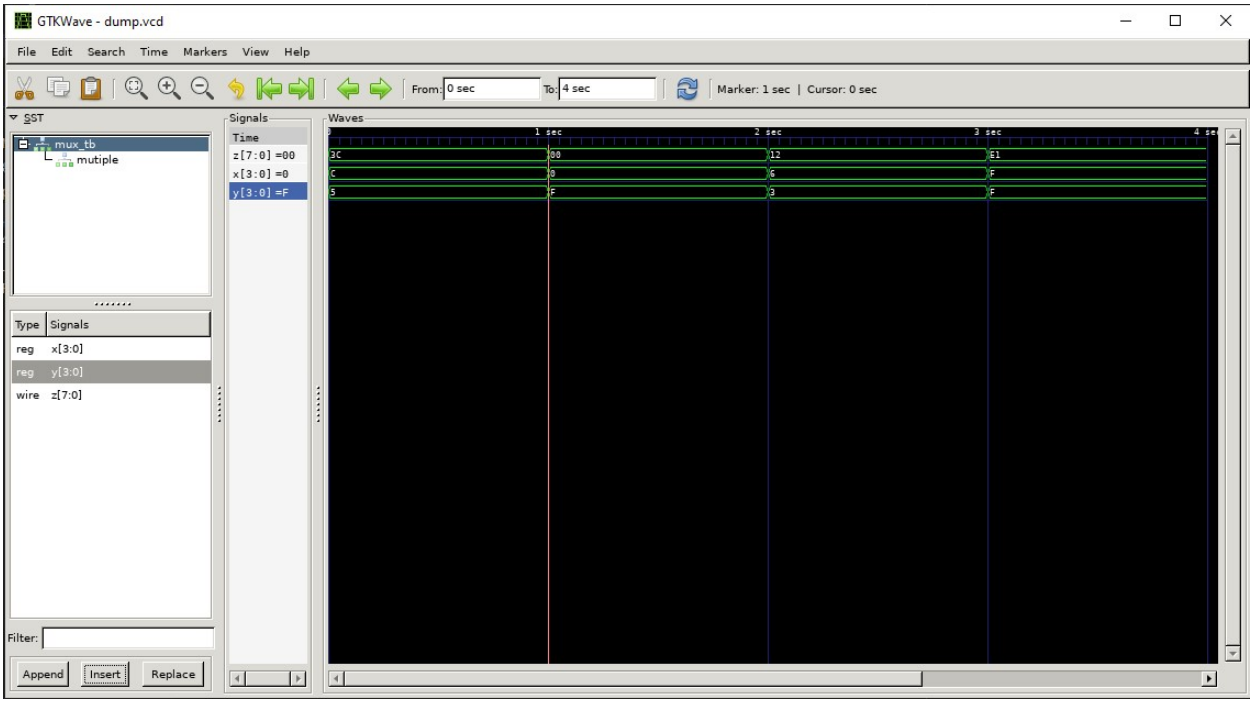

Figure 5: 4*4 multiplier Wave Form Output

The outputs of the multiplier are as expected—both the zero operands and none zero operands.

- **16-8 MUX**

```
wangy62@DESKTOP-OBTD29O:/mnt/c/Users/wangy62/LAB1$ iverilog -o myDFF mux_tb.v 16-8MUX.v
wangy62@DESKTOP-OBTD29O:/mnt/c/Users/wangy62/LAB1$ vvp myDFF
VCD info: dumpfile dump.vcd opened for output.
test with MUXC = 1.
x:00000000, y:11111111, z:00000000
test with MUXC = 0.
x:00000000, y:11111111, z:11111111
test with MUXC = 1 again.
x:00000000, y:11111111, z:00000000
```

Figure 6: 16-8 MUX iVerilog Test Branch Result

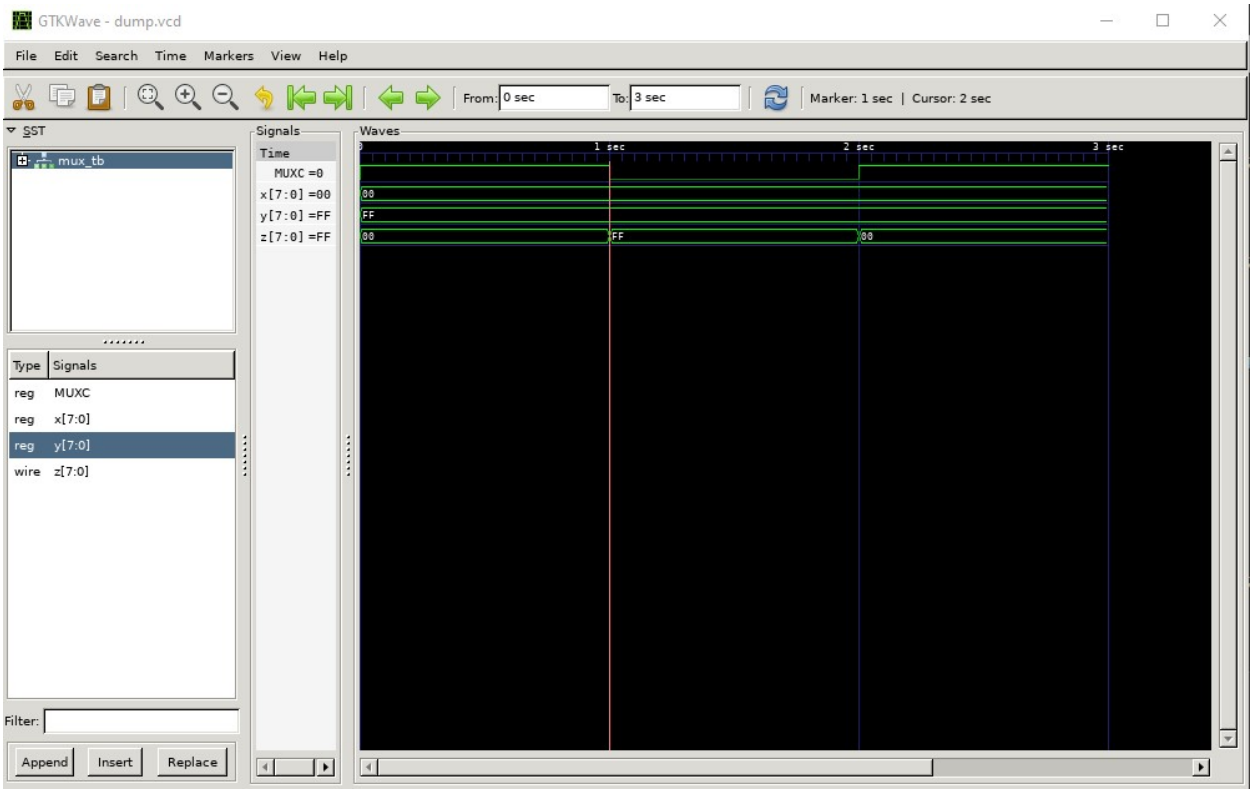

Figure 7: 16-8 MUX Wave Form Output

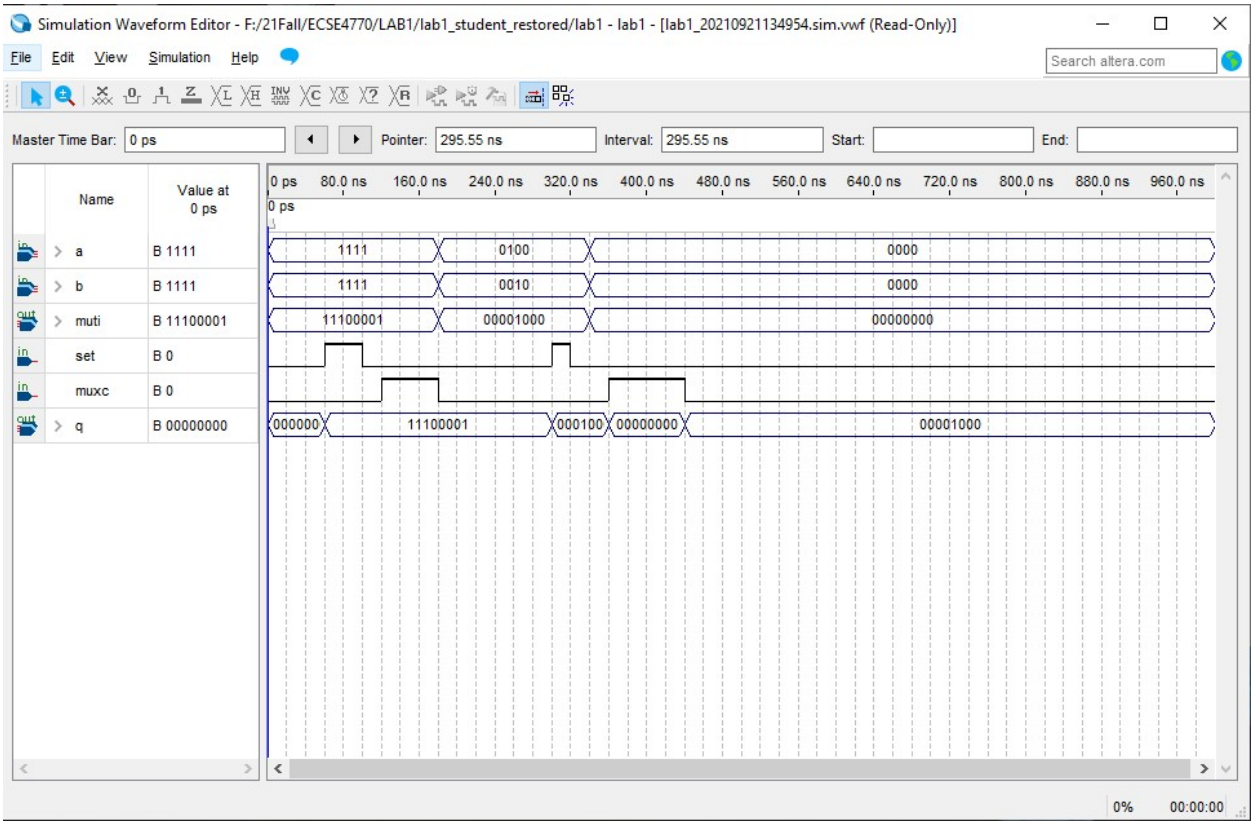

Figure 8: 16-8 MUX Quartus Simulation Result

In Figure 8, the muti pin is the output of the multiplier, set is for D-Flipflpo, and muxc is the input of 16-8 MUX. After 320ns, the output 0b0000 1000 is saved in the D-Flipflop, the output of the multiplier is 0. The muxc can determine which data the output q is.

FPGA Result

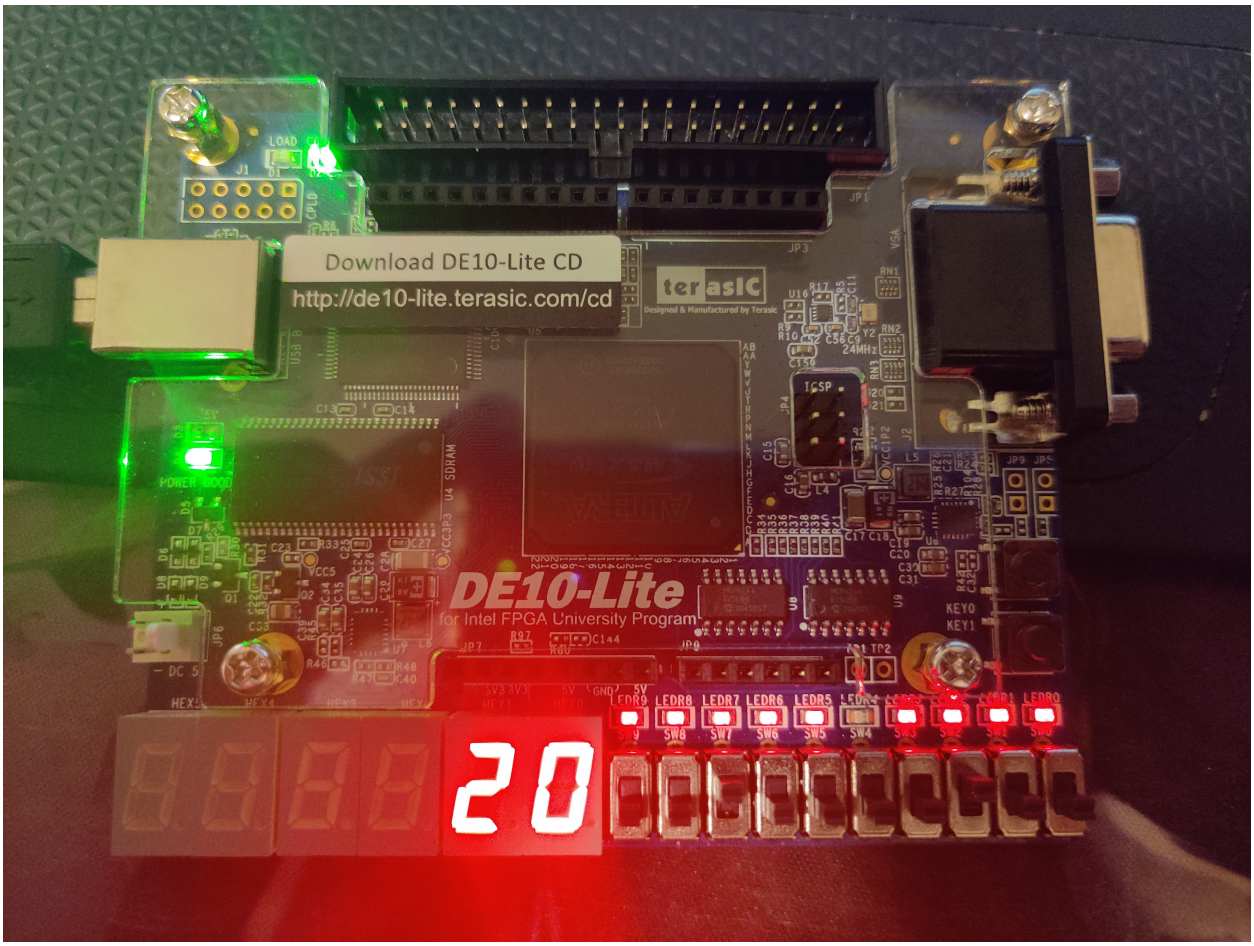

Figure 9: multiplier output with non-zero operands

Here the two inputs are 8 and 4, the result is 0x20.

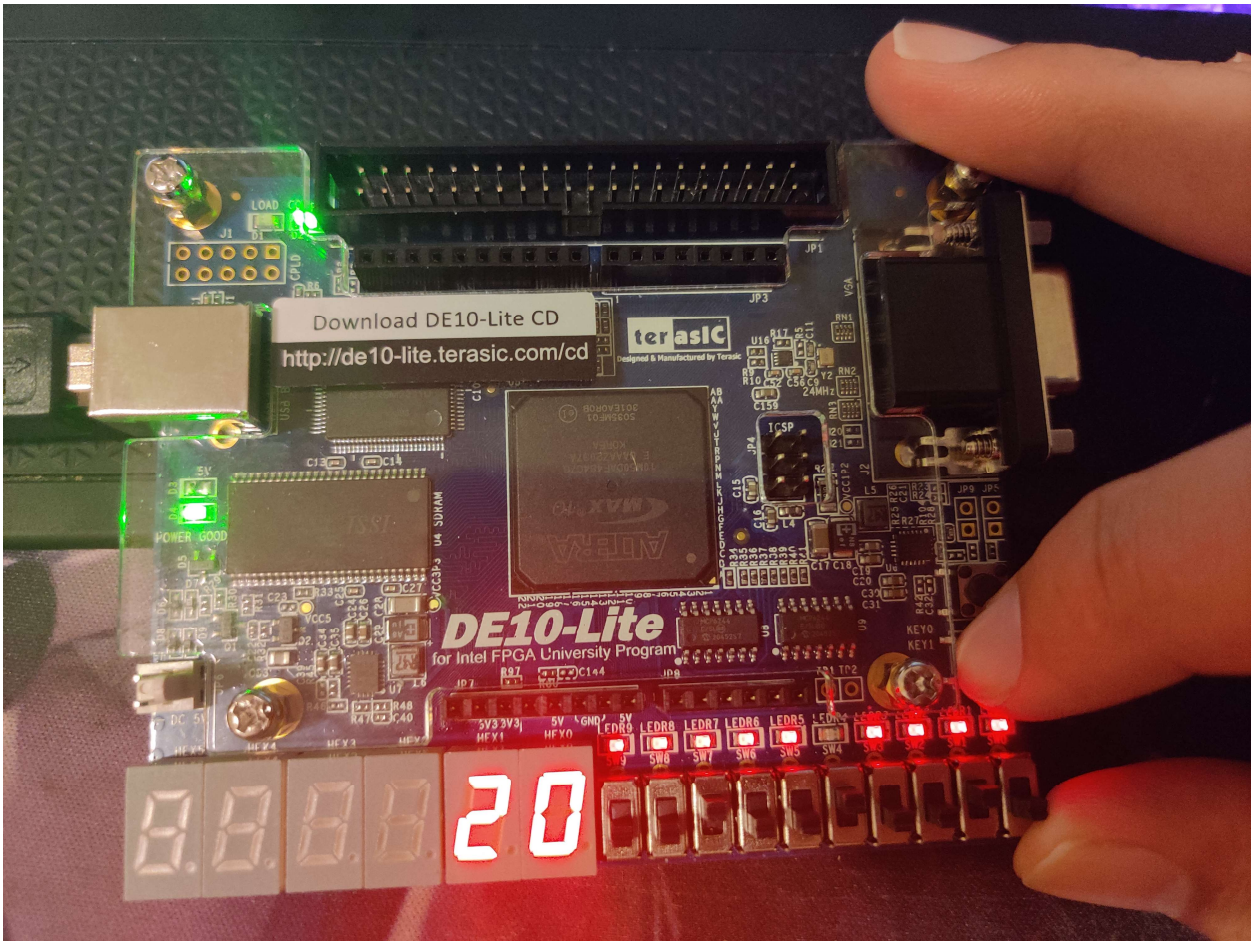

Figure 9: DFF output with non-zero operands

The inputs here are 9 and 2, the multiplier output is 0x12, but here with the push-button of MUX pushed, the output is the previous result 0x20.

Conclusion

The lab result turns out as expected. The student learned how to write Verilog code, simulate on Quaturs, and use the FPGA board to test the design.