LAB 4

Ryan Mansfield
Daniel Komac
Garrett Wood
(EEE 64 – CpE64)
Thursday

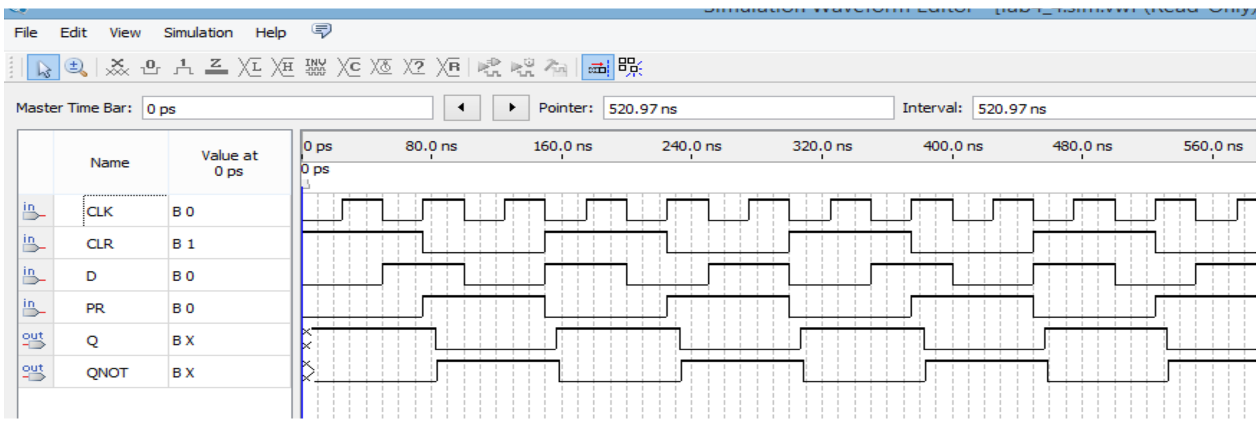
LAB 4

LAB Purpose: Introduction to latches and the D type flip-flop. To use actual flip-flops to help with understand sequential logic. Help with becoming more familiar with simulation. Help understand the function of a "clock" and the simulation of the clock. The functions of the flip-flop clock inputs using rising edge or falling edge

4.4 The behavioral model is coded as such:

```
module lab4_4(PR,CLR,D,CLK,Q,QNOT);
        input CLR,PR,D,CLK;
        output Q,QNOT;
    wire S,NAND1,NAND2,NAND3;

    assign NAND1 = ~(PR&NAND3&S);
    assign NAND2 = ~(CLK&NAND3&S);
  assign NAND3 = ~(D&CLR&NAND2);


    assign S = ~(CLR&NAND1&CLK);

      assign Q = ~(S&PR&QNOT);
    assign QNOT = ~(NAND2&CLR&Q);
           endmodule
```

And the waveforms for sample inputs look like this:



Name	Value at 0 ps
CLK	B 0
CLR	B 1
D	B 0
PR	B 0
Q	B X
QNOT	B X

4.4 The rising edge (positive 1) of the clock is the edge that changes the Q.
4.5 The logic levels of Q is 0 while the logic levels of QN is 1.
4.6 The difference of the flip flop is D will be inverted with the NAND. and will not be inverted with the NOR.

4 part 5
Code 4 bit :

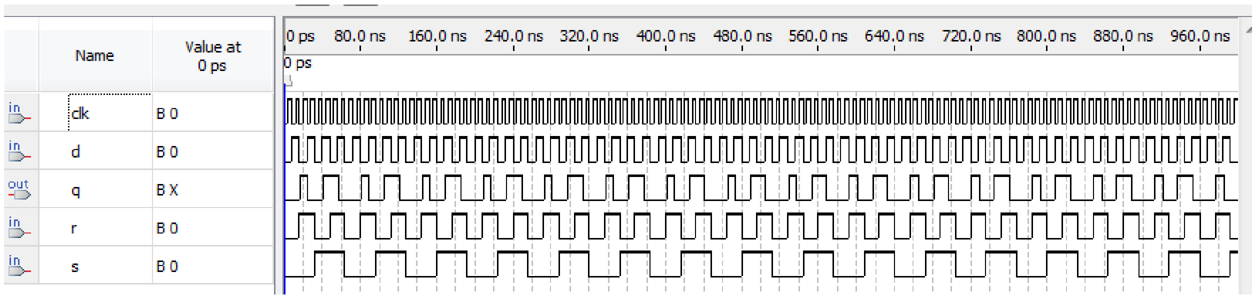
```
code 4 bit
module lab4pt5 (clk, s, r, d, q);
input clk, s, r, d;
output q;
reg q;
always @(posedge clk)
```

```
if (r) q = 4'b0;
else if (s) q = 4'b1;
else q = d;
endmodule
```

code 16 bit

```
module lab4pt5 (clk, s, r, d, q);
input clk, s, r, d;
output q;
reg q;
always @(posedge clk)
if (r) q = 16'b0;
else if (s) q = 16'b1;
else q = d;
endmodule
```

16 bit waveform:

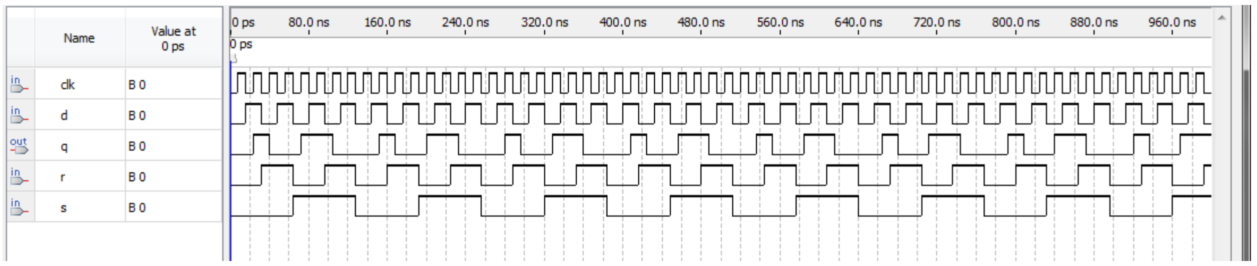


code 32 bit

```
module lab4pt5 (clk, s, r, d, q);
input clk, s, r, d;
output q;
reg q;
always @(posedge clk)
if (r) q = 32'b0;
```

```
else if (s) q = 32'b1;
else q = d;
endmodule
```

32



Name	Value at 0 ps
clk	B 0
d	B 0
q	B 0
r	B 0
s	B 0

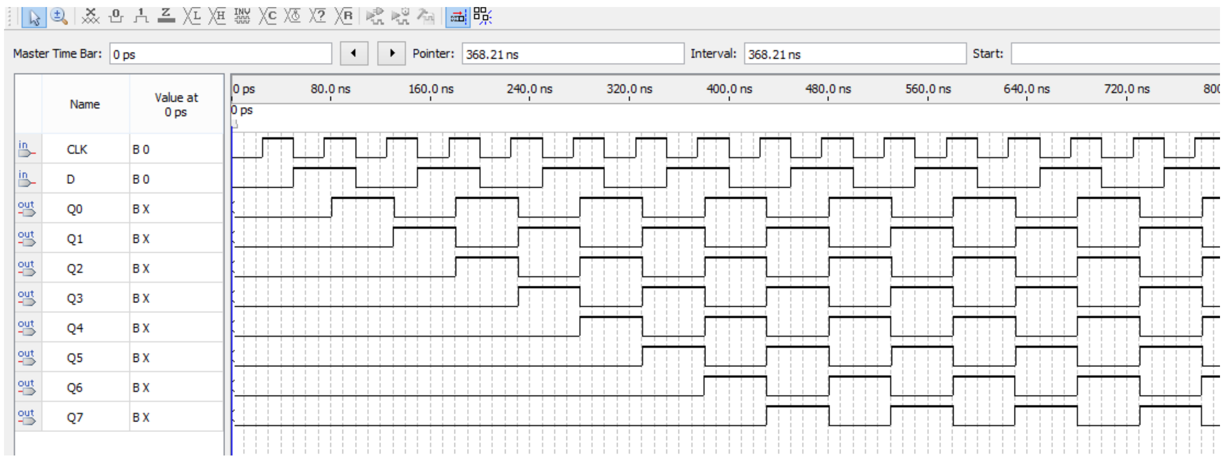
5.1 The wave function behaves with a pattern depending on the clock cycle.

Pt6

Code

```
module part6 (D,clock,Q7,Q6,Q5,Q4,Q3,Q2,Q1,Q0);

input D,clock;

output reg Q7,Q6,Q5,Q4,Q3,Q2,Q1,Q0;


always@(posedge clock) // will build a complete D-type flip-flop

begin
Q0 <= D;
Q1 <= Q0;
Q2 <= Q1;
Q3 <= Q2;
Q4 <= Q3;
Q5 <= Q4;
Q6 <= Q5;
Q7 <= Q6;
end

endmodule
```

Waveform for part 4_6:



Master Time Bar: 0 ps   Pointer: 368.21 ns   Interval: 368.21 ns   Start:

Name	Value at 0 ps
CLK	B 0
D	B 0
Q0	B X
Q1	B X
Q2	B X
Q3	B X
Q4	B X
Q5	B X
Q6	B X
Q7	B X

Conclusion. This lab helped with the understanding of: simulating clocks, the use of wave form, the use of NAND and NOR gates, and Quartus wave form simulator. The lab took about 3 hours total to complete with trying to figure out how to access and run the wave form simulator. The only improvement to the lab would be to include instructions on how to access the wave form simulator and a quick tutorial on how to use it.