

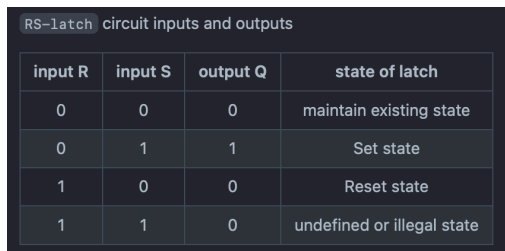
EECS 140: Lab 12
Latches FlipFlops
Morgan Bergen
KUID: 3073682
Date submitted: 12/3/2022

1. Introduction and Background

The purpose of this lab was to implement latches, flip-flops, and investigate the relationship these had to registers. The purpose of Latches are to store a single bit of information, and to change that information when a signal is applied to the latch. Latches in short are circuits that store single bits, one basic type of latch that we looked at were RS-latches which had two inputs, namely Set and Reset. These inputs provide a means for changing state Q of the circuit. The following are the configurations and state changes of such configs.

- When $R = 0$ and $S = 0$, the latch maintains its existing state.
- When $R = 0$ and $S = 1$ the latch is said to be in the Set state. In this case, the circuit output is 1.
- When $R = 1$ and $S = 0$ the latch is said to be in the Reset state. In this case, the circuit output is 0.
- When $R = 1$ and $S = 1$ the circuit is said to be in an undefined or illegal state for the RS-latch. In this case the circuit output is 0.

Flip flops were another basic sequential circuit element that had the capability of storing one bit. Flip flops changes its output state at the edge of a controlling clock signal. When a set of n flip-flops are used in order to store n bits of information we refer those as registers. A common clock is used to each flip-flop in a register. n flip-flops are required to implement a n -bit register, thus 16 flip-flops are required to implement a 16-bit register. We were able to test the functionality of the gated D latch with the test bench simulation.

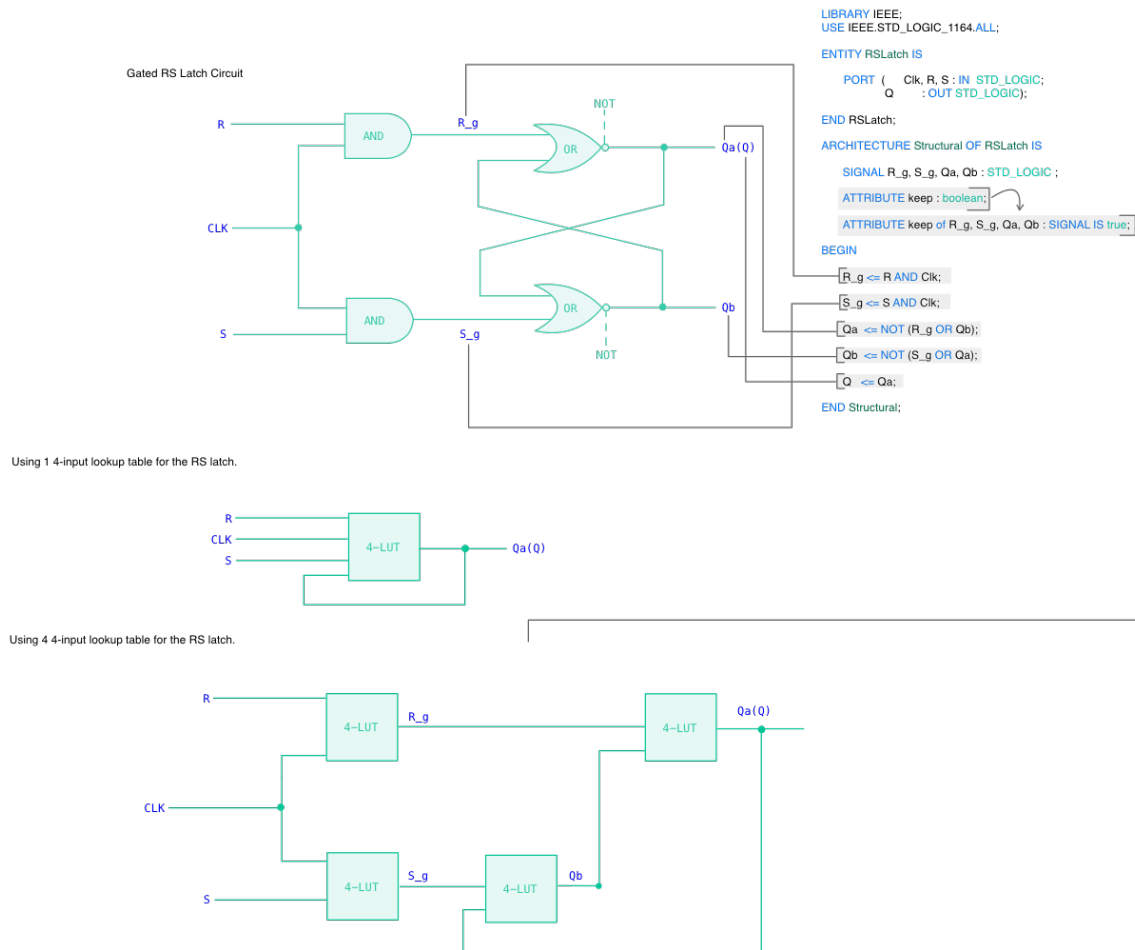


The image shows a screenshot of a simulation window titled "RS-latch circuit inputs and outputs". It contains a table with four columns: "input R", "input S", "output Q", and "state of latch". The table lists four input combinations and their corresponding outputs and states.

input R	input S	output Q	state of latch
0	0	0	maintain existing state
0	1	1	Set state
1	0	0	Reset state
1	1	0	undefined or illegal state

2. Implementation Process

The implementation process contained the following steps, understand the preconditions, generate a truth table for the R, S, and Q, implement the RSLatch, gated D Latch, and Master Slave D Flip-Flop. Also Given a 100-MHz signal, we can generate a 50-MHz and 25-MHz clock signal by dividing the 100-MHz signal by 2 and 4, respectively this is what allows us to manipulate the signal frequency for the clock. Storage elements can be created in FPGA without using flip-flops, if the latch is implemented in FPGA that has a 4-input look up table, then only one table is necessary. We needed to allow for internal signals to be observed, however this could not be done so we needed to preserve the signals via a compiler directive in the code which comprised of a VHDL attribute statement; which instructs the compiler to use separate logic elements for each of the signals, R_g , S_g , Qa , and Qb . The following diagram shows a gated RS-latch and 4-LUTs that I made along side a mapping with the code of the RSLatch.



Secondly we then created a new project with the VHDL file with the code shown above. Then there after I added a testbench source file to the project, I ran the simulation, and viewed the results. After I mapped and viewed the circuit diagram for the gated D latch and implemented this latch on the Basys3 board by performing a downloaded correction file in .tcl format, went and created a new project from scratches, ensured that all logic elements were separated, and I then thereafter ensured that separate logic elements were used to implement the signals. I went to the bitstream and uploaded the Correction.tcl script file onto tcl.pre to avoid any conflicts, and I verified that all the latches worked properly for all input conditions. I implemented the gated D latch on the board, used two slide switches to provide for the inputs D and CLK. And wrote my own xdc file for implementing it on the board.

Lastly I made another project that instantiated the two copies of the gated D latch entity and to implement the master-slave flip-flop. I included the appropriate ports on the board and connected the Q output to an LED, compiled, and tested the functionality by toggling the D and Clock switches. The following will show all modules and simulations.

```

Project Summary  x master_slave_d_latch_tb.vhd  x master_slave_d_latch.vhd  x
H:/Documents/EECS140-Lab/lab12-d/srcs/sources_1/new/master_slave_d_latch.vhd

1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity master_slave_d_latch is
5   Port ( Clk, D : in STD_LOGIC;
6         Q : out STD_LOGIC);
7 end master_slave_d_latch;
8
9 architecture Structural of master_slave_d_latch is
10
11   COMPONENT RSLatch IS
12     Port ( Clk, D : in STD_LOGIC;
13           Q : out STD_LOGIC);
14   END COMPONENT RSLatch;
15
16   SIGNAL Qm, Qs, nClk : STD_LOGIC;
17   ATTRIBUTE keep : boolean;
18   ATTRIBUTE keep of Qm, Qs, nClk : SIGNAL is TRUE;
19
20   begin
21
22     nClk <= not(Clk);
23
24     master : RSLatch
25       PORT MAP (Clk => Clk, D => D, Q => Qm);
26
27     slave : RSLatch
28       PORT MAP (Clk => nClk, D => Qm, Q => Qs);
29
30     Q <= Qs;
31
32   end architecture Structural;
33

```

```

RSLatch.vhd
H:/Documents/EECS140-Lab/lab12-d/srcs/sources_1/new/RSLatch.vhd

1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity RSLatch is
5   Port ( Clk, R, S : in STD_LOGIC;
6         Q : out STD_LOGIC);
7 end RSLatch;
8
9 architecture Structural of RSLatch is
10
11   SIGNAL R_g, S_g, Qa, Qb : STD_LOGIC;
12   ATTRIBUTE keep : boolean;
13   ATTRIBUTE keep of R_g, S_g, Qa, Qb : SIGNAL is true;
14
15   begin
16
17     R_g <= R AND Clk;
18     S_g <= S AND Clk;
19     Qa <= NOT (R_g OR Qb);
20     Qb <= NOT (S_g OR Qa);
21     Q <= Qa;
22
23   end Structural;
24
25

```

```

RSLatch_tb.vhd
H:/Documents/EECS140-Lab/lab12-d/srcs/sources_1/new/RSLatch_tb.vhd

1 LIBRARY ieee;
2 USE ieee.std_logic_1164.ALL;
3
4 ENTITY RSLatch_tb IS
5 END RSLatch_tb;
6
7 ARCHITECTURE behavior OF RSLatch_tb IS
8
9   COMPONENT RSLatch
10     PORT (Clk,R,S : IN STD_LOGIC;
11           Q : OUT STD_LOGIC);
12   END COMPONENT RSLatch;
13
14   signal S : std_logic := '0';
15   signal R : std_logic := '0';
16   signal Clk : std_logic := '0';
17   signal Q : std_logic;
18
19   constant clk_period : time := 10 ns;
20
21   BEGIN
22
23     uut: RSLatch PORT MAP (Clk => Clk,
24                           R=>R,
25                           S=>S,
26                           Q=>Q);
27
28     -- Clock process definitions( clock with 50% duty cycle is generated here.
29     clk_process :process
30     begin
31       clk <= '0';
32       wait for clk_period/2;
33       clk <= '1';
34       wait for clk_period/2;
35     end process;
36
37     rslatch_proc :process
38     begin
39       s <= '0';
40       r <= '1';
41       wait for 20ns;
42       s <= '1';
43       r <= '0';
44       wait for 20ns;
45       s <= '0';
46       r <= '0';
47       wait for 20ns;
48       s <= '1';
49       r <= '1';
50       wait for 20ns;
51     end process;
52   END;

```

```

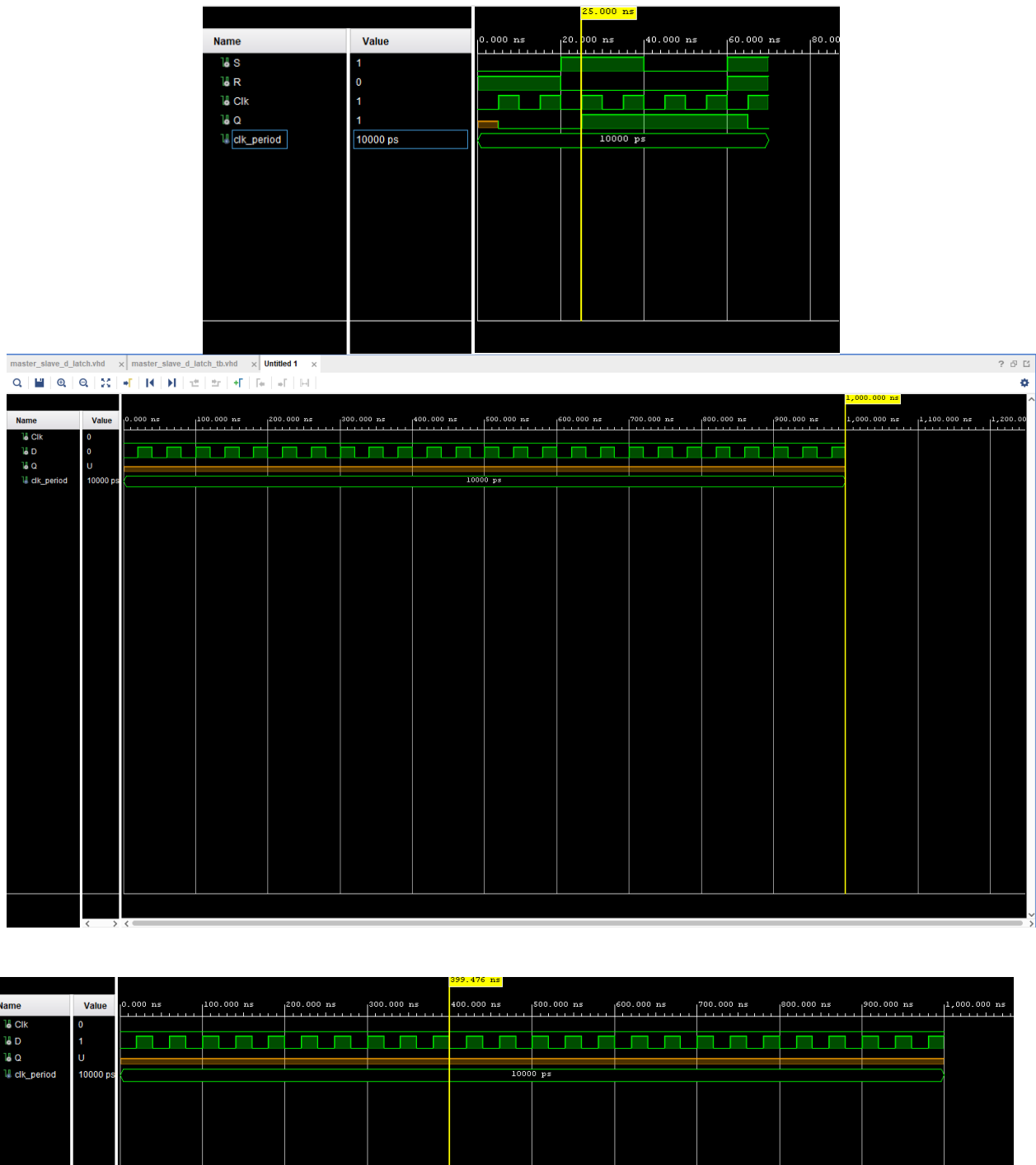
Project Summary  x master_slave_d_latch_tb.vhd  x master_slave_d_latch.vhd  x
H:/Documents/EECS140-Lab/lab12-d/srcs/sources_1/new/master_slave_d_latch_tb.vhd

1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity master_slave_d_latch_tb is
5   -- Port (/)
6 end master_slave_d_latch_tb;
7
8 architecture bench of master_slave_d_latch_tb is
9
10   component master_slave_d_latch
11     Port ( Clk, D : in STD_LOGIC;
12           Q : out STD_LOGIC);
13   end component;
14
15   signal Clk, D : STD_LOGIC;
16   signal Q : STD_LOGIC;
17
18   constant clk_period : time := 10 ns;
19
20   begin
21
22     uut : master_slave_d_latch port map (Clk => Clk, D => D, Q => Q);
23
24     clk_process : process
25     begin
26       clk <= '0';
27       wait for clk_period/2;
28     end process;
29
30     rslatch_proc : process
31     begin
32       d <= '0';
33       wait for 20ns;
34       d <= '1';
35       wait for 20ns;
36     end process;
37
38   END;
39

```

3. Evaluation Process

The evaluation process comprised of running simulations and viewing how the differing rates from the gates differed.



4. Results and Discussion

I ran the simulation and was able to see the results as the expected outputs from the logical expressions. I changes the code from the first source and used it in the master slave flip-flop.

5. Conclusion and Recommendations

I enjoyed coding these logic blocks, and I'm going to miss doing these labs for their difficulty was high, however the outputs and results were very high rewarding.