OBJECTIVES:

• Examine S-R, gated S-R, and gated D-type latches.
• Create the designs for the S-R, gated S-R, and gated D latches in schematic mode.
• Test the designs on the target board.

MATERIALS:

• Xilinx Vivado software, student or professional edition V2018.2 or higher.
• IBM or compatible computer with Pentium III or higher, 128 M-byte RAM or more, and 8 G-byte Or larger hard drive.
• BASYS 3 Board.

DISCUSSION:

In this experiment, we will discuss sequential circuits. The main difference between combinational circuits and sequential circuits is that combinational circuits do not have memory elements. So the output of a combinatorial circuit depends only on the present inputs. But the output of a sequential circuit depends on the effects of prior inputs (the memory) as well as the present inputs. Latches are simple, but very important, class of memory elements.

S-R NOR Latch

The S-R NOR latch has two inputs: S and R (SET and RESET) and two outputs: Q and not Q. The Q is the normal output and not Q is the complemented output. Any latch has two states: SET and RESET (CLEAR). When Q = 1, we say the latch is in the SET state. When Q = 0, the latch is in the RESET state. Figure 11.1 shows the...
construction of a NOR latch. (The notation S-C, SET & CLEAR, is sometimes used for SR latches.)

The truth table below (Table 11.1) describes the characteristics of this NOR latch.

![SR Latch with NOR gates](image)

The SR NAND Latch

A NOR latch has active-high inputs. When both inputs are low (S=0, R=0), the output will not change. It is “latched”. Normally, one of the inputs in it could be set to high to “set” or “clear” the latch. Yet if both inputs are high (S=1 and R=1), both outputs will be low, which is not valid since Q and not-\(Q\) should be opposites.

**The SR NAND Latch**
A NAND latch has active-low inputs. When both inputs are high (S=1, R=1), the output will not change. It is “latched”. Normally, one of the inputs in it could be set to high to “set” or “clear” the latch. Yet if both inputs are low (S=0 and R=0), both outputs will be low, which is not valid since \( Q \) and not-\( Q \) should be opposites.

**The Gated S-R Latch**

In applications, we often want to make the latch latched and ignore any inputs changes in certain period. An enable line (EN) is added for this purpose. As shown in Figure 11.2, two more gates are added to obtain the gated S-R latch. The gated S-R latch is also called the level-triggered SR flip-flop (S-R FF) since \( Q \) can change only when \( EN \) “pulls the trigger”.

**SR Latch (NAND) Truth Table**

<table>
<thead>
<tr>
<th>Set (S)</th>
<th>Reset (R)</th>
<th>( Q )</th>
<th>( Q' )</th>
<th>State</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Reset</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>After S=1, R=0 (no change)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Set</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>After S=0, R=1 (no change)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Forbidden</td>
<td>Forbidden</td>
</tr>
</tbody>
</table>

Table 11.1.2 Truth Table for the SR NAND Latch
The truth table in Table 11.2 shows how the EN input controls when the latch can respond to the S-R inputs.

<table>
<thead>
<tr>
<th>EN</th>
<th>Set (S)</th>
<th>Reset (R)</th>
<th>Q</th>
<th>Q'</th>
<th>State</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>No change</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Invalid (Q=Q'=0)</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>No change</td>
</tr>
</tbody>
</table>

Table 11.2 Truth Table for the Gated SR Latch with Active High EN

It could be found that the function of the EN input is to enable/disable the inputs S and R.

**The Gated D Latch**

The gated D latch (D for data) can be built by adding an inverter before each of the two inputs in a gated S-R latch. A gated D latch is also called a level-triggered D flip-flop (D FF). Its diagram is shown in Figure 11.3.
By examining the following truth table, we can see that a level-triggered D FF has a simple operation. The output Q simply follows the data input D when the enable input is activated. Q is latched when the enable is low. There is no invalid state in this latch.

<table>
<thead>
<tr>
<th>EN</th>
<th>D(Data)</th>
<th>Q</th>
<th>Q'</th>
<th>State</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>No change</td>
</tr>
</tbody>
</table>

Table 11.3. The Truth Table for the Gated D-Latch with Active High EN
PROCEDURE:
Section I. The NOR Latch and NAND Latch

1. For S-R Latches, write the following code:

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity S_R_latch_top is
  Port ( S : in  STD_LOGIC;
         R : in  STD_LOGIC;
         Q : inout STD_LOGIC;
         notQ : inout STD_LOGIC); -- changed out to inout
end S_R_latch_top;

architecture Behavioral of S_R_latch_top is
--signal notQ : STD_LOGIC;
begin
  Q  <= R nor notQ;
  notQ <= S nor Q;
end Behavioral;
```

2. Implement the simulation that is similar to the ones shown below. It is important to cover all the cases to fill your truth table.
3. Next, we need to add SR-NAND Latch as following to a new file:

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity srlatch_top is
  Port ( S : in STD_LOGIC;
           R : in STD_LOGIC;
           Q : inout STD_LOGIC;
           notQ : inout STD_LOGIC );
end srlatch_top;

architecture Behavioral of srlatch_top is
begin
  srlatch_top: process
  begin
    -- stimulus process
    srlatch_top: process
    begin
      -- define variables
      S <= '0';
      R <= '0';
      wait for 10 ns;
      S <= '1';
      R <= '0';
      wait for 10 ns;
      S <= '0';
      R <= '1';
      wait for 10 ns;
      S <= '1';
      R <= '1';
      wait for 10 ns;
    end process;
  end process;
end Behavioral;
```

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity sr latch is
  Port ( S : in STD_LOGIC;
         R : in STD_LOGIC;
         Q : inout STD_LOGIC;
         notQ : inout STD_LOGIC);
end sr latch;

architecture Behavioral of sr latch is
begin
  -- define variables
  Q <= S NAND notQ;
  notQ <= R NAND Q;
end Behavioral;
```
4. Then, the simulation program similarly:

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity srnandSim is
  -- Port ( )
end srnandSim;

architecture Behavioral of srnandSim is
COMPONENT srnand
  PORT(
    S : IN std_logic;
    R : IN std_logic;
    Q : inout std_logic;
    notQ : inout std_logic
  );
END COMPONENT;

  -- Inputs
  signal S : std_logic := '0';
  signal R : std_logic := '0';

  -- Outputs
  signal Q : std_logic;
  signal notQ : std_logic;
BEGIN

  -- Instantiate the Unit Under Test (UUT)
  uut: srnand PORT MAP (  
    S => S,
    R => R,
    Q => Q,
    notQ => notQ
  );

  -- stimulus process
  stim_proc: process
  begin

```

5. Do not forget to change simulation settings to srNand latch simulation. Simulation results for NAND based latch should look similar to:
6. Fill the truth tables by using your simulation outputs.

### SR Latch (NOR) Truth Table

<table>
<thead>
<tr>
<th>Set (S)</th>
<th>Reset (R)</th>
<th>Q</th>
<th>Q'</th>
<th>State</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 11.4 Experimental Results for the SR(NOR) Latch

### SR Latch (NAND) Truth Table

<table>
<thead>
<tr>
<th>Set (S)</th>
<th>Reset (R)</th>
<th>Q</th>
<th>Q'</th>
<th>State</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 11.5 Experimental Results for the SR(NAND) Latch
7. Compare the characteristics of the NOR latch with those of the NAND latch and comment on the differences and similarities of these two latches.

Section II. The Gated SR and D Latches

1. Write the following VHDL code for gated SR latch:

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Gated_SR_Latch is
  Port ( S, R : in STD_LOGIC;
         clock : in STD_LOGIC;
         Q, Q_n : inout STD_LOGIC);
end Gated_SR_Latch;
architecture Gated_SR_Latch_arch of Gated_SR_Latch is
  signal S_tmp:STD_LOGIC;
  signal R_tmp:STD_LOGIC;
begin
  S_tmp <= clock AND S;
  R_tmp <= clock AND R;
  Q <= R_tmp NOR Q_n;
  Q_n <= S_tmp NOR Q;
end Gated_SR_Latch_arch;
```

2. Implement the simulation changing variables according to your needs to fill the truth table. You should see the simulation output as shown below.
3. Fill the truth table:

Gated SR Latch Truth Table

<table>
<thead>
<tr>
<th>EN</th>
<th>Set (S)</th>
<th>Reset (R)</th>
<th>Q</th>
<th>Q'</th>
<th>State</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 11.6 Experimental Results for the Gated SR Latch with Active High EN

4. Implement the D latch with gate by writing the following code:
5. Prepare the simulation environment and variables. Your output should look like:

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity D_Latch is
  GENERIC (DELAY : time := 2 ns);
  Port ( D : in STD_LOGIC;
         clock : in STD_LOGIC;
         Q : out STD_LOGIC;
         Q_n : out STD_LOGIC);
end D_Latch;

architecture D_Latch_arch of D_Latch is
  signal Q_tmp:STD_LOGIC;
begin
  PROCESS (Din,clock)
  BEGIN
    if (clock = '1') then
      Q_tmp <= Din after DELAY;
    end if;
    Q <= Q_tmp;
    Q_n <= NOT Q_tmp;
  end PROCESS;
end D_Latch_arch;
```

![Simulation waveform](image-url)
6. Fill the truth table:

<table>
<thead>
<tr>
<th>EN</th>
<th>D(Data)</th>
<th>Q</th>
<th>Q'</th>
<th>State</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 11.7 Experimental Results for the Gated-D Latch with Active High EN
QUESTIONS

1) Draw the logic diagram for a gated S-R latch using only NAND gates.

2) How does the Gate or Enable inputs work in the gated latches?

3) If we want the enable input active low, what kind of modification(s) should be applied to the gated D latch? Draw the diagram.

4) How could you avoid the invalid state in an S-R latch?