EXPERIMENT 10

Latches: SR & D-Type

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.)

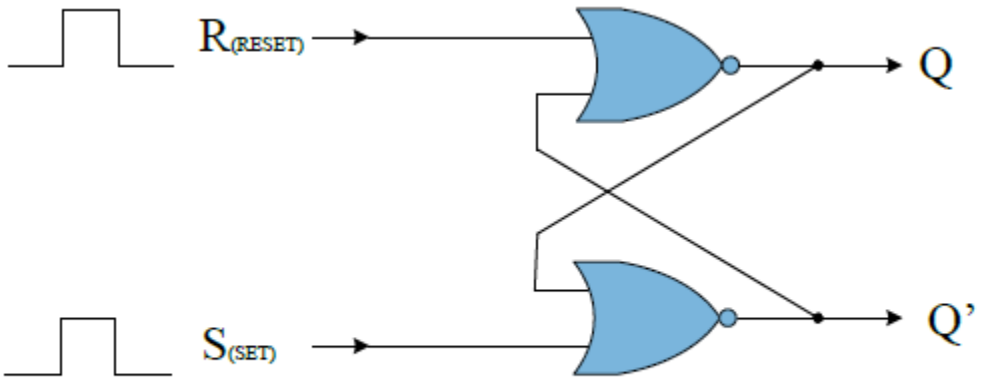


Figure 11.1 SR Latch with NOR gates

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

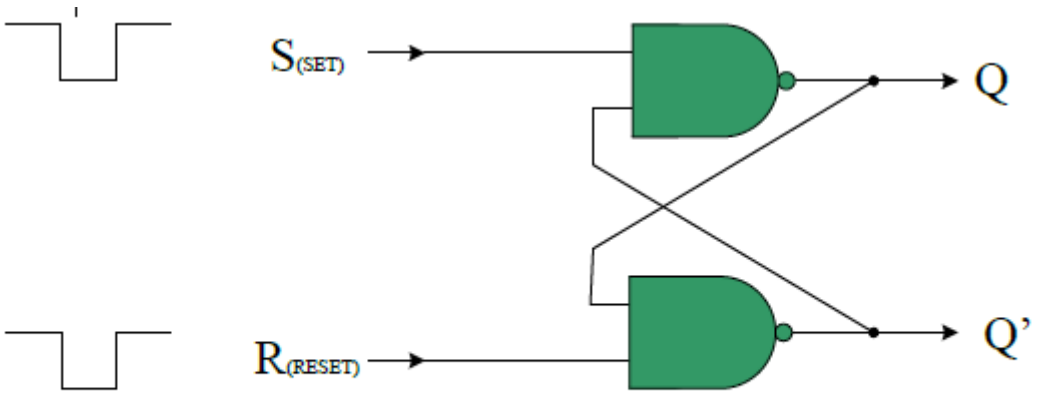
SR Latch (NOR) Truth Table

Set (S)	Reset (R)	Q	Q'	State	Note
1	0	1	0	1	Set
0	0	1	0	1	After S=1, R=0(no change)
0	1	0	1	0	Reset
0	0	0	1	0	After S=0, R=1(no change)
1	1	0	0	Forbidden	Forbidden

Table 11.1.1 The Truth Table for the SR NOR 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



SR Latch with NAND gates

SR Latch (NAND) Truth Table

Set (S)	Reset (R)	Q	Q'	State	Note
1	0	0	1	0	Reset
1	1	0	1	0	After S=1, R=0(no change)
0	1	1	0	0	Set
1	1	1	0	1	After S=0, R=1(no change)
0	0	1	1	Forbidden	Forbidden

Table 11.1.2 Truth Table for 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".

Figure 11.2

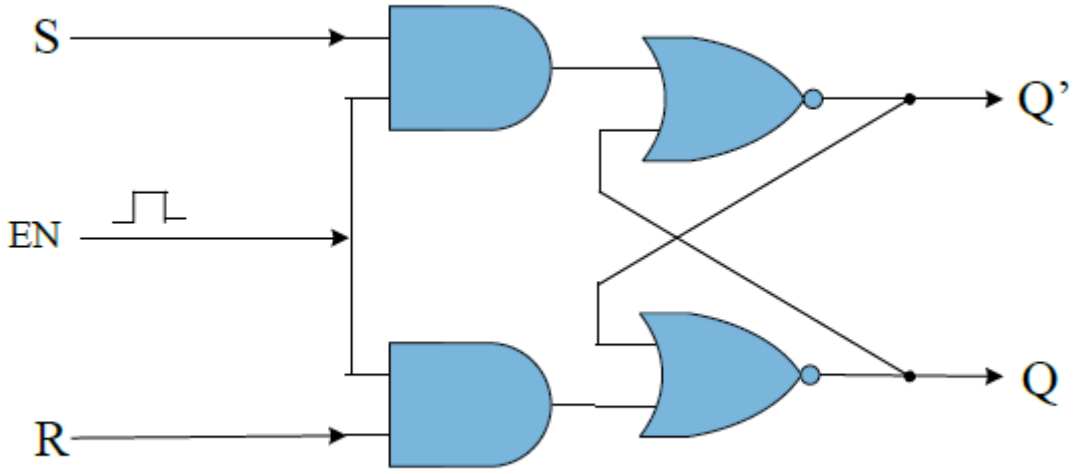


Diagram for the Gated SR Latch with Active High EN

The truth table in Table 11.2 shows how the **EN** input controls when the latch can respond to the **S-R** inputs.

Gated SR Latch Truth Table

EN	Set (S)	Reset (R)	Q	Q'	State	Note
1	0	0	1	0	1	No change
1	0	1	0	1	0	
1	1	0	1	0	1	
1	1	1	0	0	0	Invalid (Q=Q'=0)
0	X	X	X	X	X	No change

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.

Figure 11.3

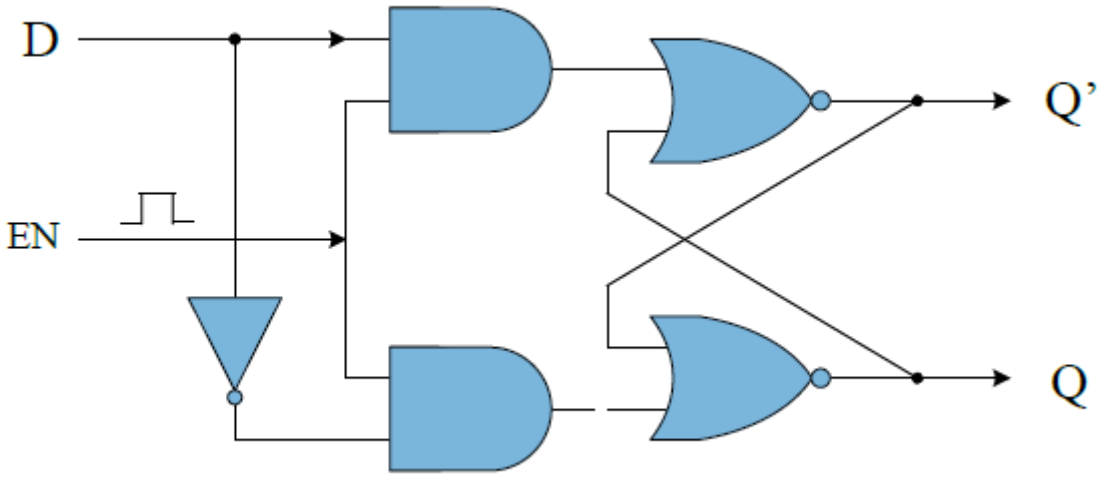


Diagram for Gated D Latch with Active High EN

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.

Gated D-Latch Truth Table

EN	D(Data)	Q	Q'	State	Note
1	0	0	1	0	
1	1	1	0	1	
0	X	X	X	X	No change

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:

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3
4   entity S_R_latch_top is
5       Port ( S : in    STD_LOGIC;
6              R : in    STD_LOGIC;
7              Q : inout STD_LOGIC;
8              notQ : inout STD_LOGIC); -- changed out to inout
9   end S_R_latch_top;
10
11  architecture Behavioral of S_R_latch_top is
12  --signal notQ : STD_LOGIC;
13  begin
14
15  Q    <= R nor notQ;
16  notQ <= S nor Q;
17
18  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.

```
22      library IEEE;
23      use IEEE.STD_LOGIC_1164.ALL;
24
25
26      entity srSim is
27      --  Port ( );
28      end srSim;
29
30      architecture Behavioral of srSim is
31
32      COMPONENT S_R_latch_top
33       PORT(
34       S : IN std_logic;
35       R : IN std_logic;
36       Q : inout std_logic;
37      notQ : inout std_logic
38       );
39       END COMPONENT;
40
41       --Inputs
42       signal S : std_logic := '0';
43       signal R : std_logic := '0';
44
45       --Outputs
46       signal Q : std_logic;
47       signal notQ : std_logic;
48
49      BEGIN
50
51       -- Instantiate the Unit Under Test (UUT)
52       uut: S_R_latch_top PORT MAP (
53       S => S,
54       R => R,
55       Q => Q,
56       notQ => notQ
57       );
58
59       -- Stimulus process
60       stim_proc: process
61
```

```
62      begin
63      -- hold reset state for 100 ns.
64       wait for 100 ns;
65
66       -- insert stimulus here
67       S <= '0';
68       R <= '0';
69       wait for 10 ns;
70
71        S <= '0';
72        R <= '1';
73       wait for 10 ns;
74
75        S <= '1';
76        R <= '0';
77       wait for 10 ns;
78
79
80        S <= '0';
81        R <= '0';
82       wait for 10 ns;
83
84          S <= '0';
85          R <= '1';
86         wait for 10 ns;
87
88
89            S <= '0';
90            R <= '0';
91           wait for 10 ns;
92
93
94           S <= '1';
95           R <= '1';
96          wait for 10 ns;
97      end process;
98
99      end Behavioral;
100
```

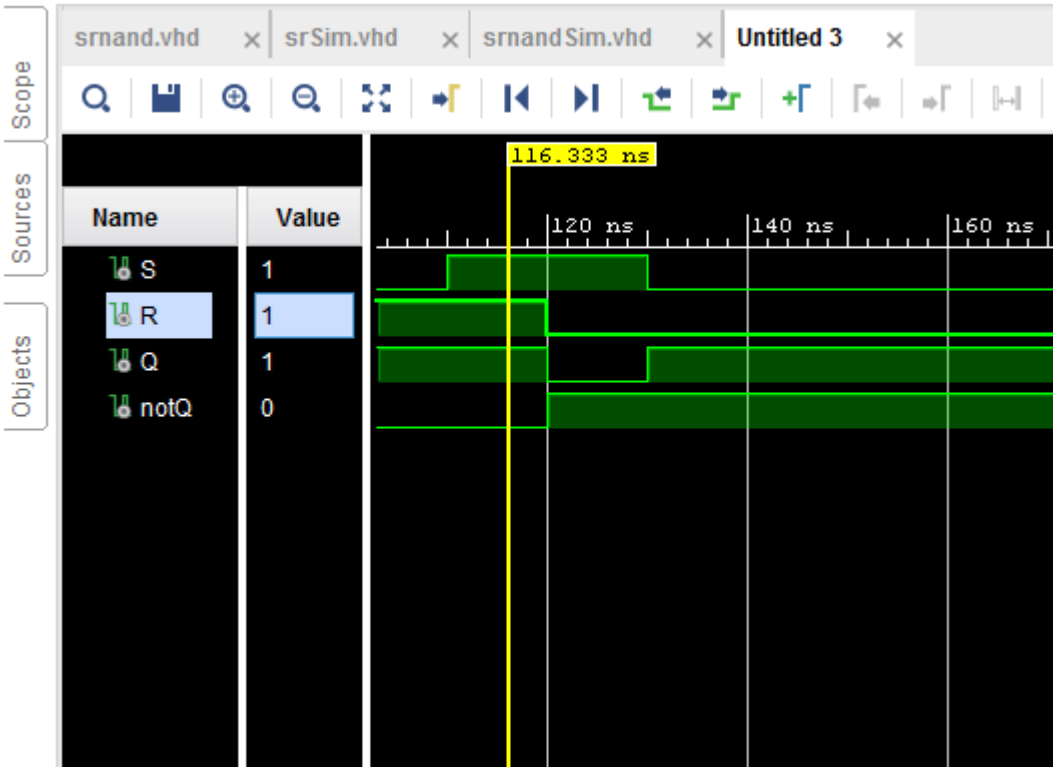
3. Next, we need to add SR-NAND Latch as following to a new file:

```
1       library IEEE;
2       use IEEE.STD_LOGIC_1164.ALL;
3
4       entity srnand is
5           Port ( S : in    STD_LOGIC;
6                  R : in    STD_LOGIC;
7                  Q : inout STD_LOGIC;
8                  notQ : inout STD_LOGIC); -- changed out to inout
9       end srnand;
10
11      architecture Behavioral of srnand is
12      --signal notQ : STD_LOGIC;
13      begin
14
15      Q    <= S NAND notQ;
16      notQ <= R NAND Q;
17
18      end Behavioral;
```

4. Then, the simulation program similarly:

```
22      library IEEE;
23      use IEEE.STD_LOGIC_1164.ALL;
24
25
26      entity srnandSim is
27      --  Port ( );
28      end srnandSim;
29
30      architecture Behavioral of srnandSim is
31
32      COMPONENT srnand
33       PORT(
34       S : IN std_logic;
35       R : IN std_logic;
36       Q : inout std_logic;
37      notQ : inout std_logic
38       );
39       END COMPONENT;
40
41       --Inputs
42       signal S : std_logic := '0';
43       signal R : std_logic := '0';
44
45       --Outputs
46       signal Q : std_logic;
47       signal notQ : std_logic;
48
49      BEGIN
50
51       -- Instantiate the Unit Under Test (UUT)
52       uut: srnand PORT MAP (
53       S => S,
54       R => R,
55       Q => Q,
56       notQ => notQ
57       );
58
59       -- Stimulus process
60       stim_proc: process
61
62      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

Set (S)	Reset (R)	Q	Q'	State	Note
1	0				
0	0				
0	1				
0	0				
1	1				

Table 11.4 Experimental Results for the SR(NOR) Latch

SR Latch (NAND) Truth Table

Set (S)	Reset (R)	Q	Q'	State	Note
1	0				
1	1				
0	1				
1	1				
0	0				

Table 11.5 Experimental Results for the SR(NAND) Latch

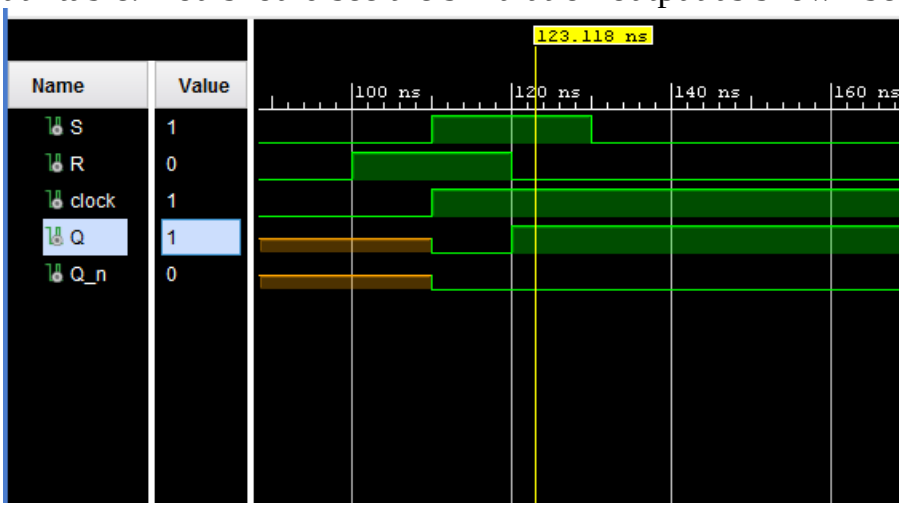
7. Compare the characteristics of the NOR latch with those of the NAND latch and comment on the differences and similarities of these two latches.

Section II. The Gated SR and D Latches

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

```
1  library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   entity Gated_SR_Latch is
4       Port ( S,R : in STD_LOGIC;
5            clock : in STD_LOGIC;
6            Q : inout STD_LOGIC;
7          Q_n : inout STD_LOGIC);
8   end Gated_SR_Latch;
9   architecture Gated_SR_Latch_arch of Gated_SR_Latch is
10  signal S_tmp:STD_LOGIC;
11  signal R_tmp:STD_LOGIC;
12  begin
13       S_tmp <= clock AND S;
14       R_tmp <= clock AND R;
15       Q <= R_tmp NOR Q_n;
16       Q_n <= S_tmp NOR Q;
17  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.



```
22      library IEEE;
23      use IEEE.STD_LOGIC_1164.ALL;
24
25
26      entity srClocksim is
27      --  Port ( );
28      end srClocksim;
29
30      architecture Behavioral of srClocksim is
31
32      COMPONENT Gated_SR_Latch
33       PORT(
34       S : IN std_logic;
35       R : IN std_logic;
36       clock : IN std_logic;
37       Q : inout std_logic;
38       Q_n : inout std_logic
39       );
40       END COMPONENT;
41
42       --Inputs
43       signal S : std_logic := '0';
44       signal R : std_logic := '0';
45       signal clock : std_logic := '0';
46
47       --Outputs
48       signal Q : std_logic;
49       signal Q_n : std_logic;
50
51      BEGIN
52
53       -- Instantiate the Unit Under Test (UUT)
54       uut: Gated_SR_Latch PORT MAP (
55       S => S,
56       R => R,
57       clock => clock,
58       Q => Q,
59       Q_n => Q_n
60       );
61
```

3. Fill the truth table:

Gated SR Latch Truth Table

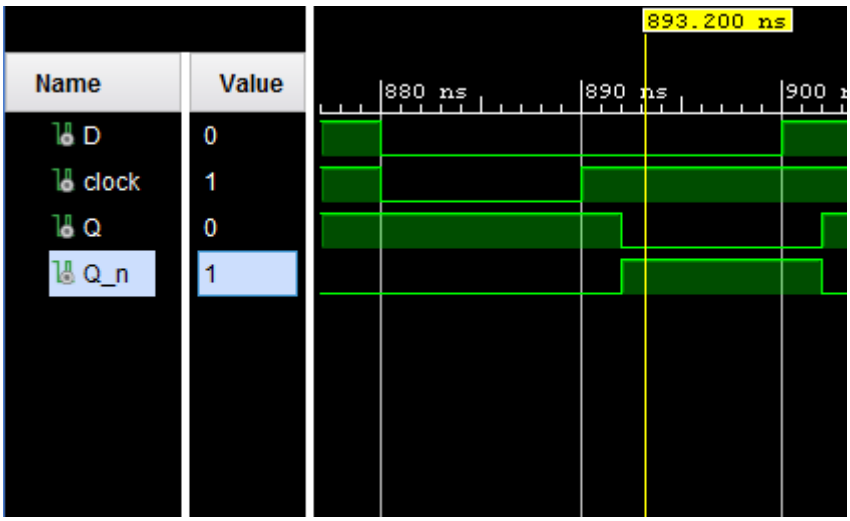
EN	Set (S)	Reset (R)	Q	Q'	State	Note
1	0	0				
1	0	1				
1	1	0				
1	1	1				
0	X	X				

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:

```
1  library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3
4   entity D_Latch is
5     GENERIC (DELAY : time :=2 ns);
6     Port ( Din : in STD_LOGIC;
7            clock : in STD_LOGIC;
8                   Q : out STD_LOGIC;
9          Q_n : out STD_LOGIC);
10  end D_Latch;
11
12  architecture D_Latch_arch of D_Latch is
13  signal Q_tmp:STD_LOGIC;
14  begin
15   PROCESS (Din,clock)
16   BEGIN
17    if (clock = '1') then
18         Q_tmp <= Din after DELAY;
19        end if;
20   END PROCESS;
21  Q <= Q_tmp;
22  Q_n <= NOT Q_tmp;
23  end D_Latch_arch;
```

5. Prepare the simulation environment and variables. Your output should look like:



```
22      library IEEE;
23      use IEEE.STD_LOGIC_1164.ALL;
24
25
26      entity gatedDSim is
27      --  Port ( );
28      end gatedDSim;
29
30      architecture Behavioral of gatedDSim is
31
32      COMPONENT D_Latch
33       PORT(
34       D : IN std_logic;
35       clock : IN std_logic;
36       Q : inout std_logic;
37       Q_n : inout std_logic
38       );
39       END COMPONENT;
40
41       --Inputs
42       signal D : std_logic := '0';
43       signal clock : std_logic := '0';
44
45       --Outputs
46       signal Q : std_logic;
47       signal Q_n : std_logic;
48
49      BEGIN
50
51       -- Instantiate the Unit Under Test (UUT)
52       uut: D_Latch PORT MAP (
53       D => D,
54       clock => clock,
55       Q => Q,
56       Q_n => Q_n
57       );
58
59       -- Stimulus process
60       stim_proc: process
61
62      begin
```

6. Fill the truth table:

Gated D-Latch Truth Table

EN	D(Data)	Q	Q'	State	Note
1	0				
1	1				
0	X				

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?