Chapter 8

Introduction to Sequential Logic

Sequential Circuit

- A digital circuit whose output depends not only on the present combination of inputs, but also on the history of the circuit.

- 有反饋

Sequential Circuit Elements

- Two basic types:
  - **Latch**
  - **Flip-flop**

- The difference is the condition under which the stored bit changes.

Sequential Circuit Inputs

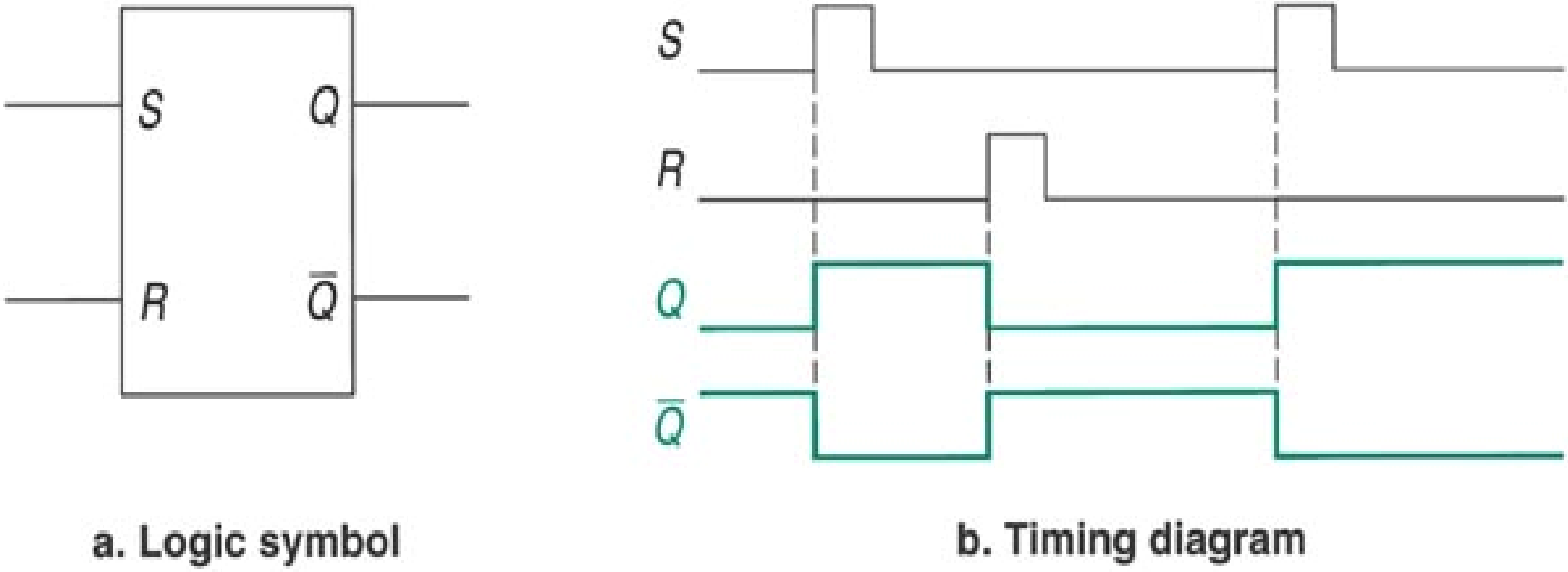
- The LATCH is a sequential circuit with two inputs (SET and RESET).

  → SET – an latch input that makes the latch store a logic 1.

  → RESET – an latch input that makes the latch store a logic 0.

Sequential Circuit Outputs

- Two complementary outputs (Q , $\overline{Q}$).
- Outputs are always in opposite logic states.



a. Logic symbol

b. Timing diagram

Sequential Circuit States

- Definitions:

$$\text{SET}: \quad Q = 1,\ \overline{Q} = 0$$

$$\text{RESET}: \quad Q = 0,\ \overline{Q} = 1$$

Active HIGH or LOW Inputs

- Latches can have either active HIGH or active LOW inputs.

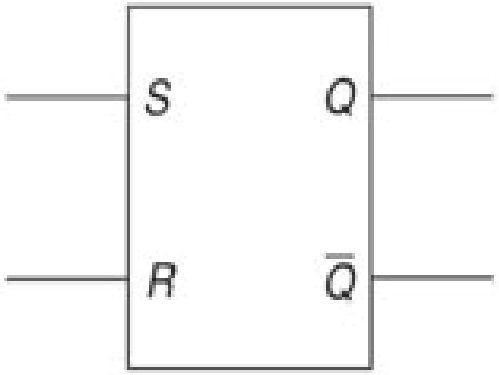
- The output of the LATCH, regardless of the input active level, is still defined as:

$$\text{SET}: \quad Q = 1, \ \overline{Q} = 0$$

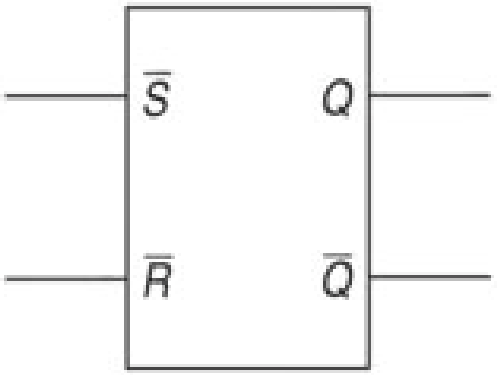
$$\text{RESET}: \quad Q = 0, \ \overline{Q} = 1$$

Example 8.1: Active HIGH or LOW Inputs -1

Latches can have active-HIGH or active-LOW inputs, but in each case $Q = 1$ after the set function is applied and $Q = 0$ after reset. For each latch shown in Figure 8.3, complete the timing diagram shown. Q is initially LOW in both cases. (The state of Q before the first active *SET* or *RESET* is unknown unless specified, since the present state depends on previous history of the circuit.)



a. Latch with active-HIGH inputs

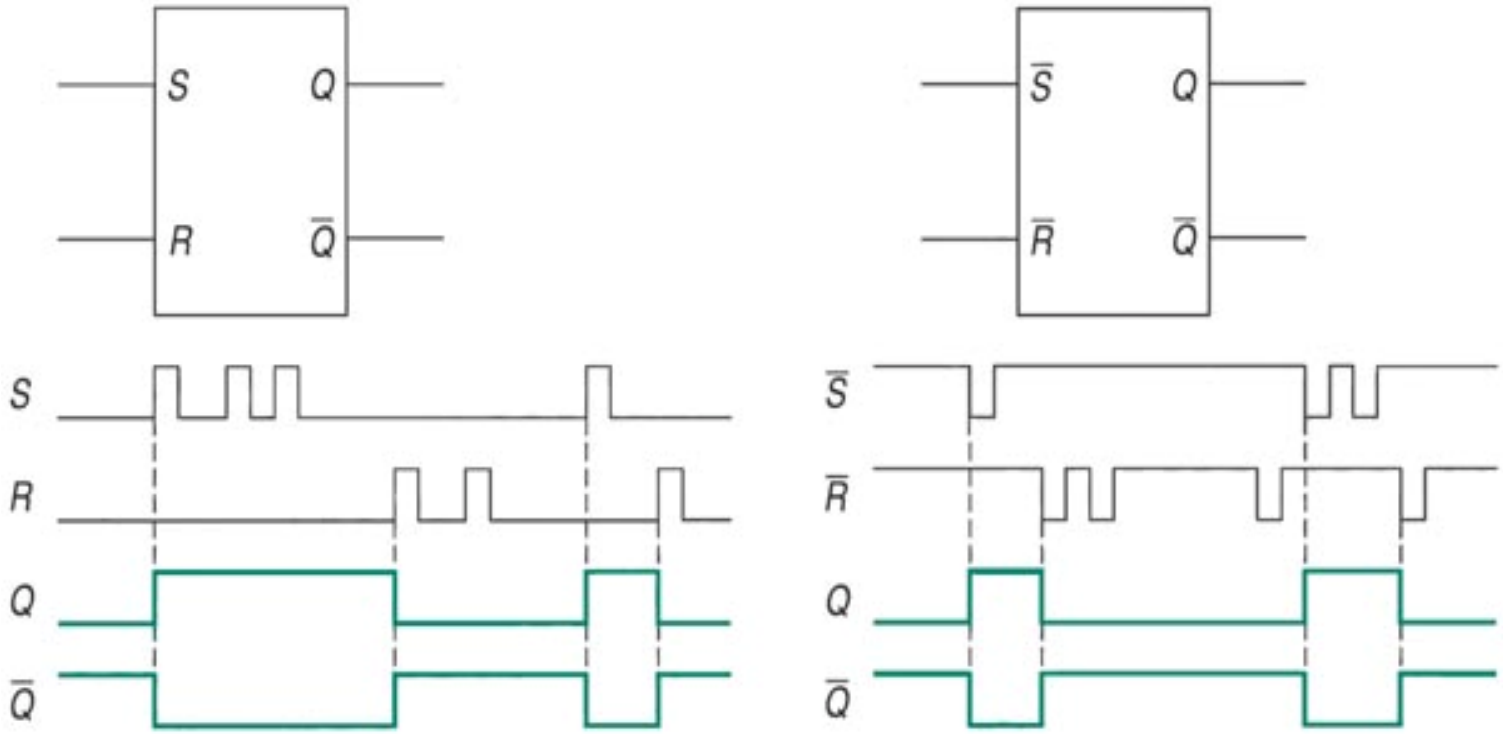


b. Latch with active-LOW inputs

Example 8.1: Active HIGH or LOW Inputs -2

■ Solution

The Q and $\bar{Q}$ waveforms are shown in Figure 8.3. Note that the outputs respond only to the first set or reset command in a sequence of several pulses.



a. Latch with active-HIGH inputs

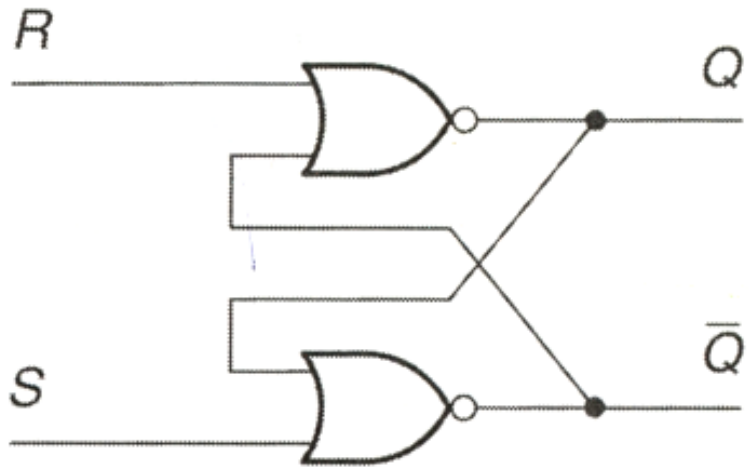
b. Latch with active-LOW inputs

8.2 NAND/NOR Latches

NOR gates and NAND gates

cuit are drawn in DeMorgan equivalent form.

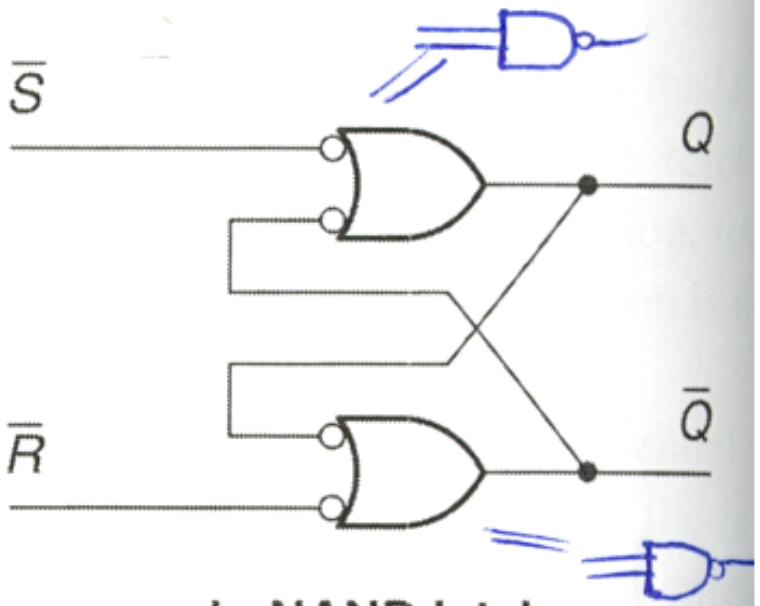
The NAND gates in the second



a. NOR latch

Active High

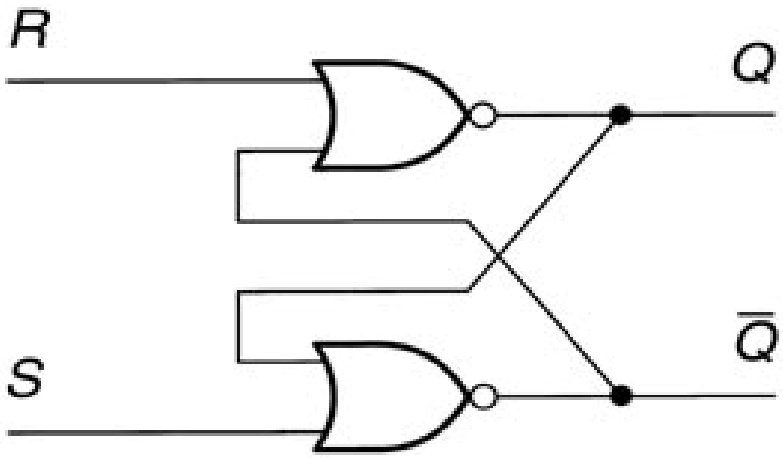
SR Latch Circuits



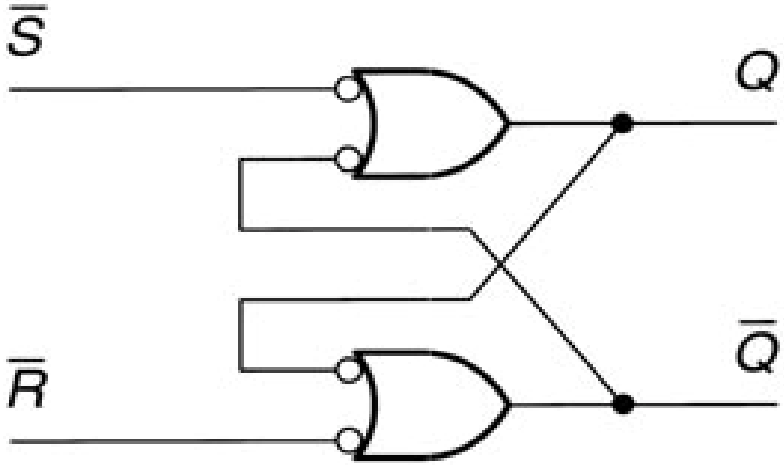
b. NAND latch

Active Low

NAND/NOR Latches -1



a. NOR latch

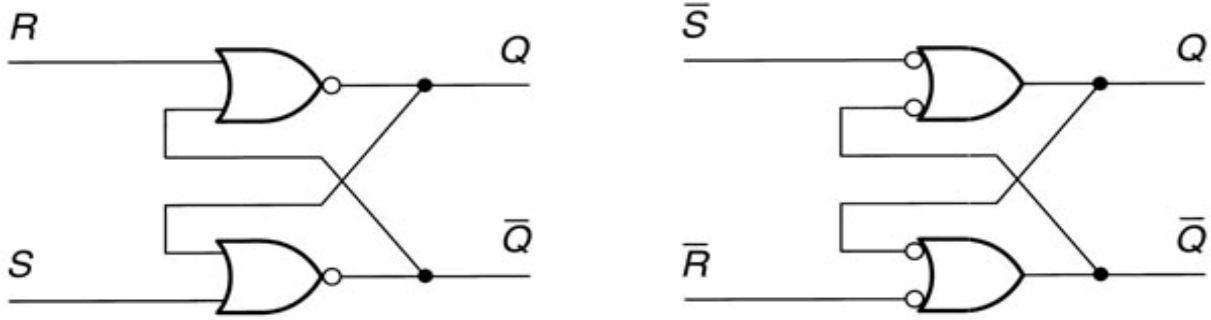


b. NAND latch

The two circuits both have the following three features:

1. OR-shaped gates
2. Logic level inversion between the gate input and output
3. Feedback from the output of one gate to an input of the opposite gate

NAND/NOR Latches -2



a. NOR latch

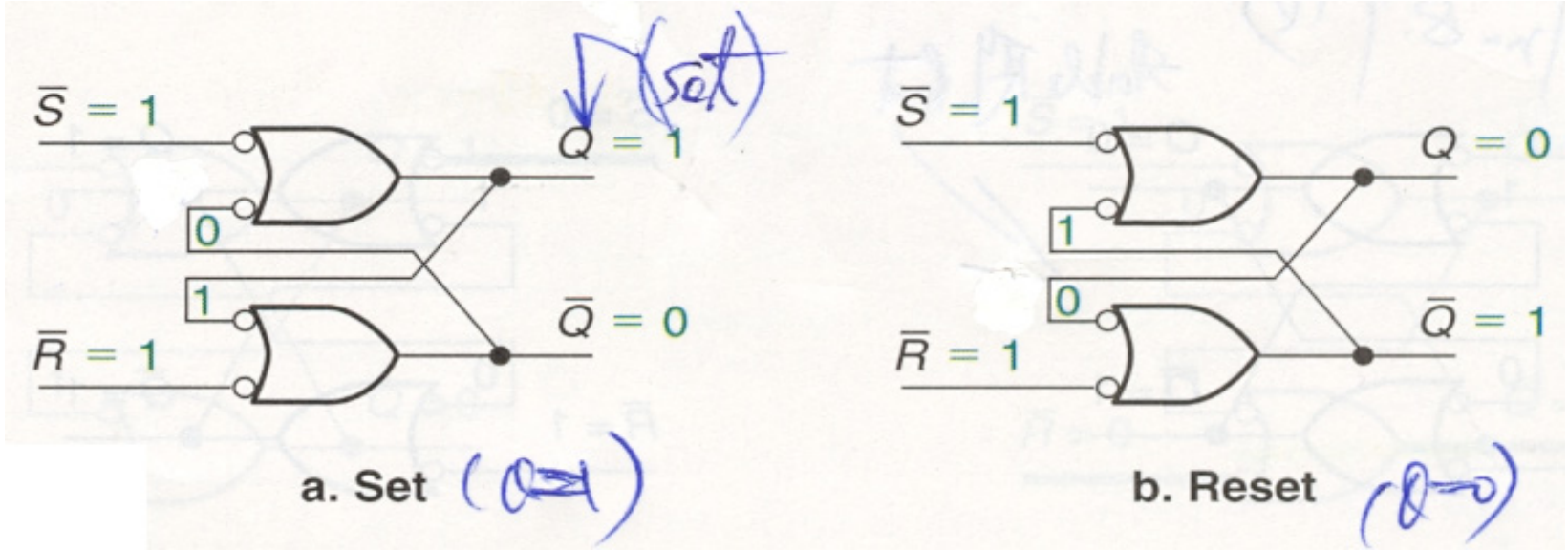
b. NAND latch

NOR and NAND Latch Functions

S	R	Action (NOR Latch)	$\overline{S}$	$\overline{R}$	Action (NAND Latch)
0	0	Neither *SET* nor *RESET* active; output does not change from previous state	0	0	Both *SET* and *RESET* active; forbidden condition
0	1	*RESET* input active	0	1	*SET* input active
1	0	*SET* input active	1	0	*RESET* input active
1	1	Both *SET* and *RESET* active; forbidden condition	1	1	Neither *SET* nor *RESET* active; output does not change from previous state

NAND Latch Operation

Stable States -1



only one of the four inputs to the two NAND gates has a 0. The difference between *SET* and *RESET* states has to do with the placement of the logic 0 on one of these "inner" inputs. Whichever gate has the 0 input will have the HIGH output; the other gate will have a LOW output.

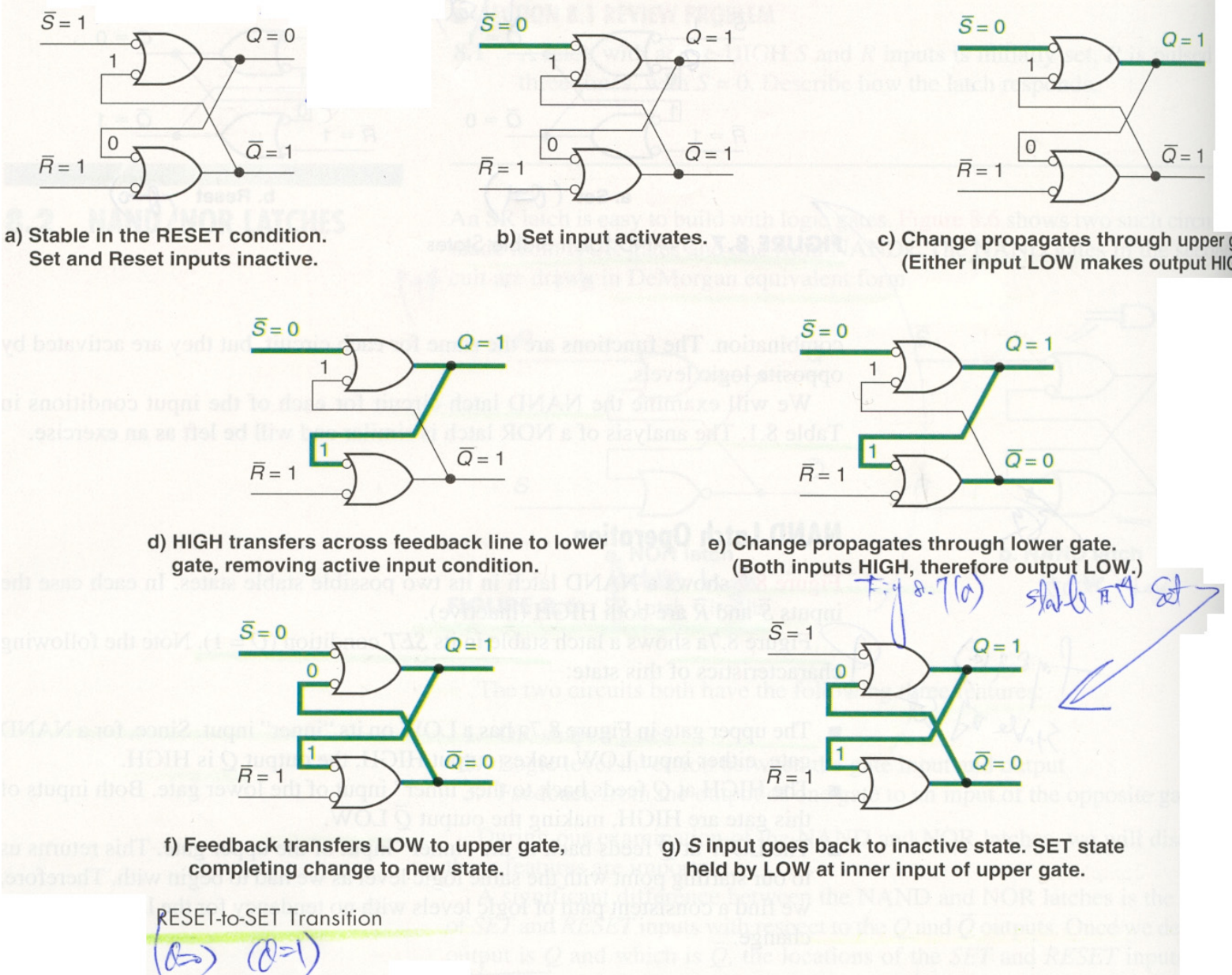
NAND Latch Operation

Stable States -2

- Two possible stable states:
  - SET
  - RESET
- Feedback keeps the latch in a stable condition.

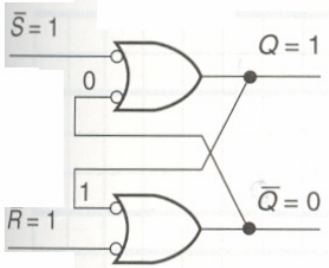
NAND Latch Operation

RESET-to-SET Transition

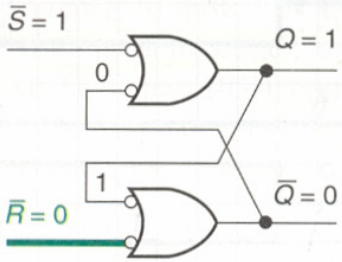


NAND Latch Operation

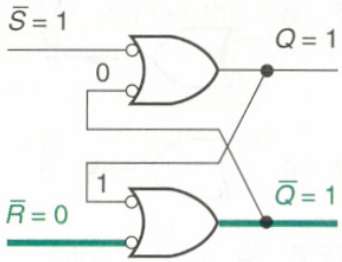
SET-to-RESET Transition



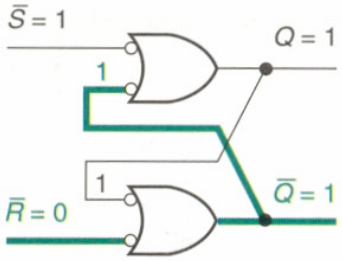
a) Stable in the SET condition. Set and Reset inputs inactive.



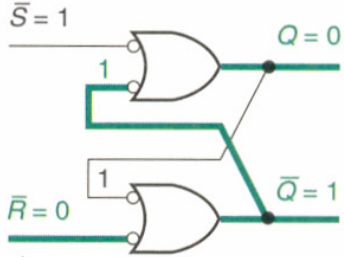
b) Reset input activates.



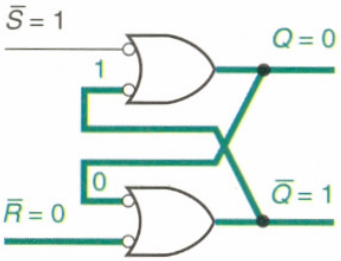
c) Change propagates through lower gate. (Either input LOW makes output HIGH.)



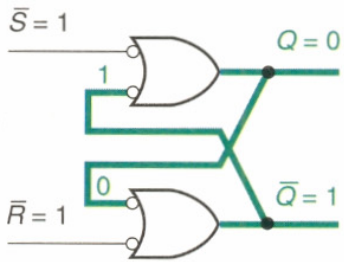
d) HIGH transfers across feedback line to upper gate, removing active input condition.



e) Change propagates through upper gate. (Both inputs HIGH, therefore output LOW.)



f) Feedback line transfers LOW to lower gate, completing transition to RESET state.

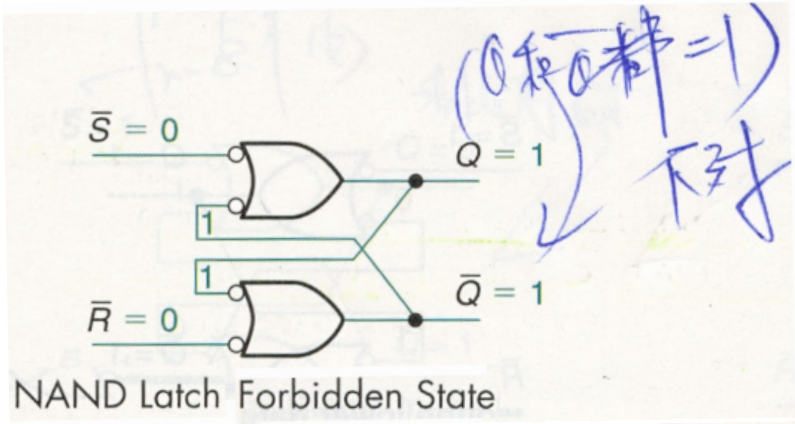


g) R goes back to inactive level. Latch is stable in new state, held by the 0 on inner input of lower gate.

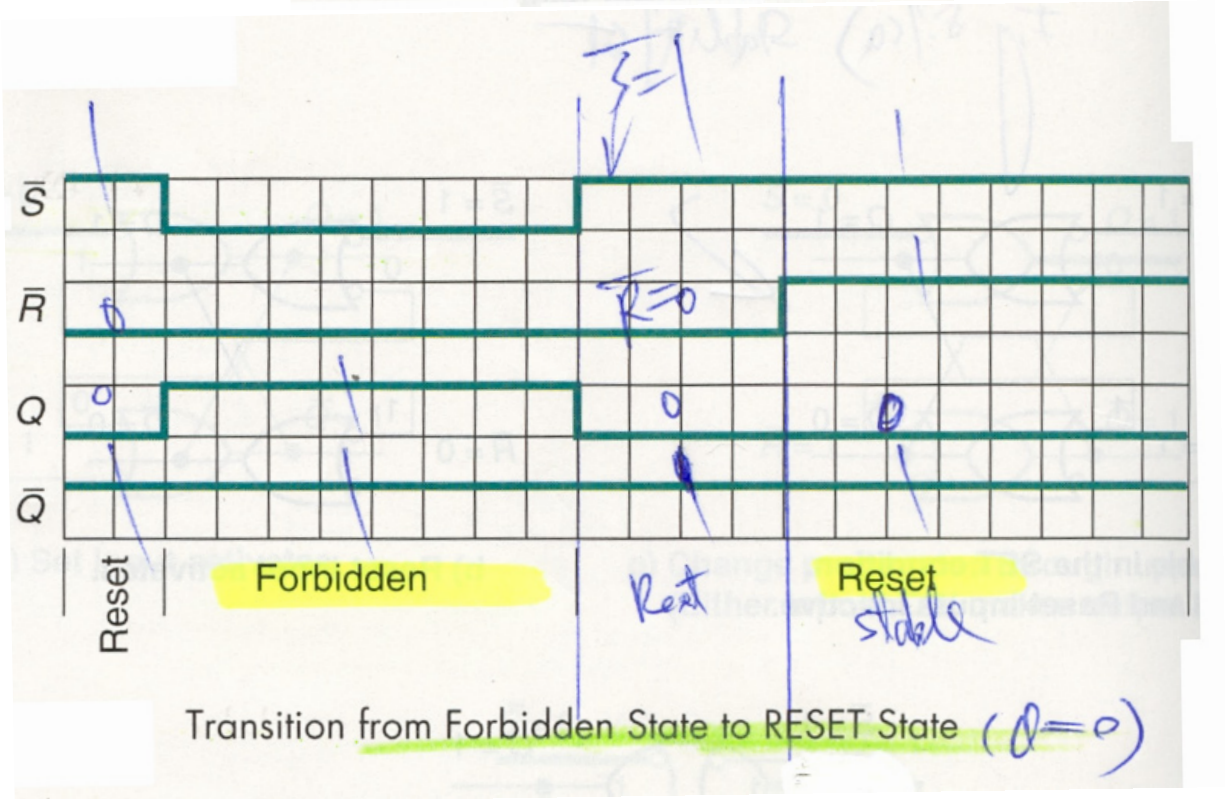
SET-to-RESET Transition

NAND Latch Operation

Forbidden-to-RESET



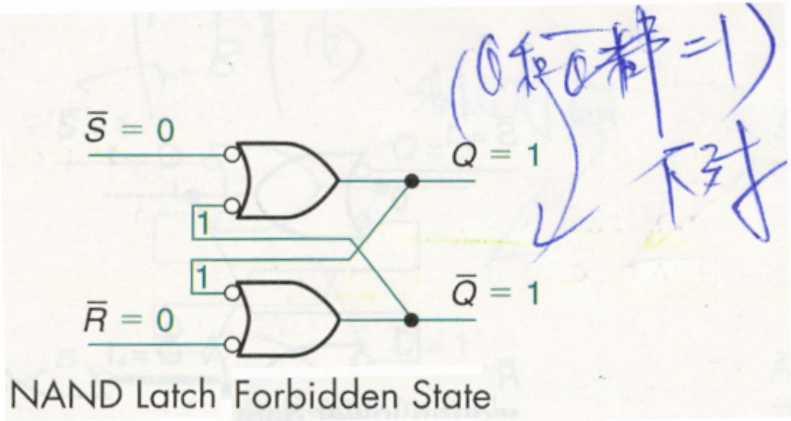
NAND Latch Forbidden State



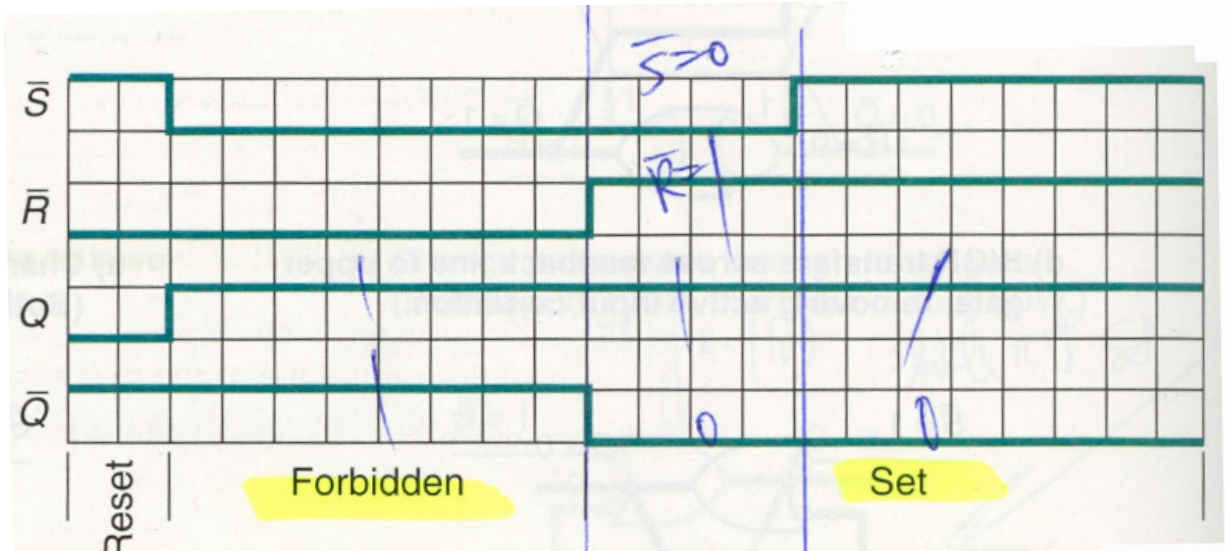
Transition from Forbidden State to RESET State

NAND Latch Operation

Forbidden-to-SET



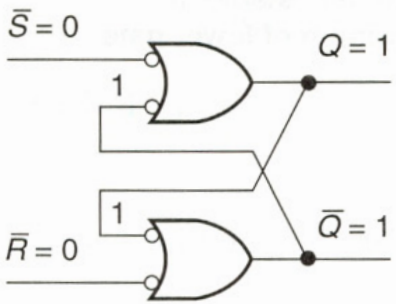
NAND Latch Forbidden State



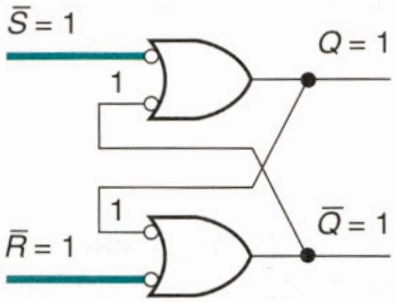
Transition from Forbidden State to SET State

NAND Latch Operation

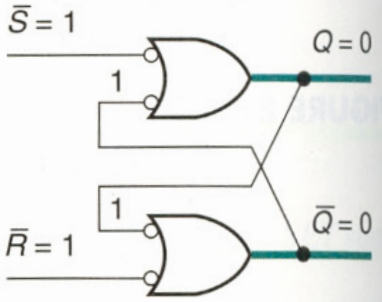
Forbidden-to-Oscillation -1



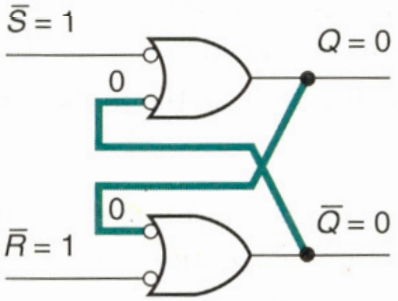
a) Both Set and Reset inputs active. Either input LOW makes output HIGH. Therefore, both outputs HIGH.



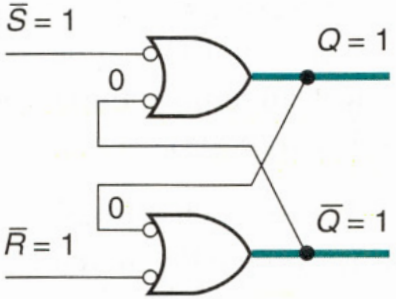
b) Set and Reset inputs deactivate simultaneously.



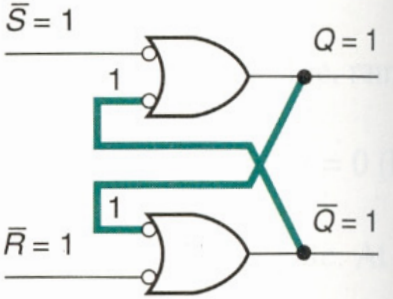
c) Change propagate through gates simultaneously.



d) New output levels cross circuit via feedback lines.



e) Either input LOW makes output HIGH. Changes on feedback lines propagate through both gates simultaneously.

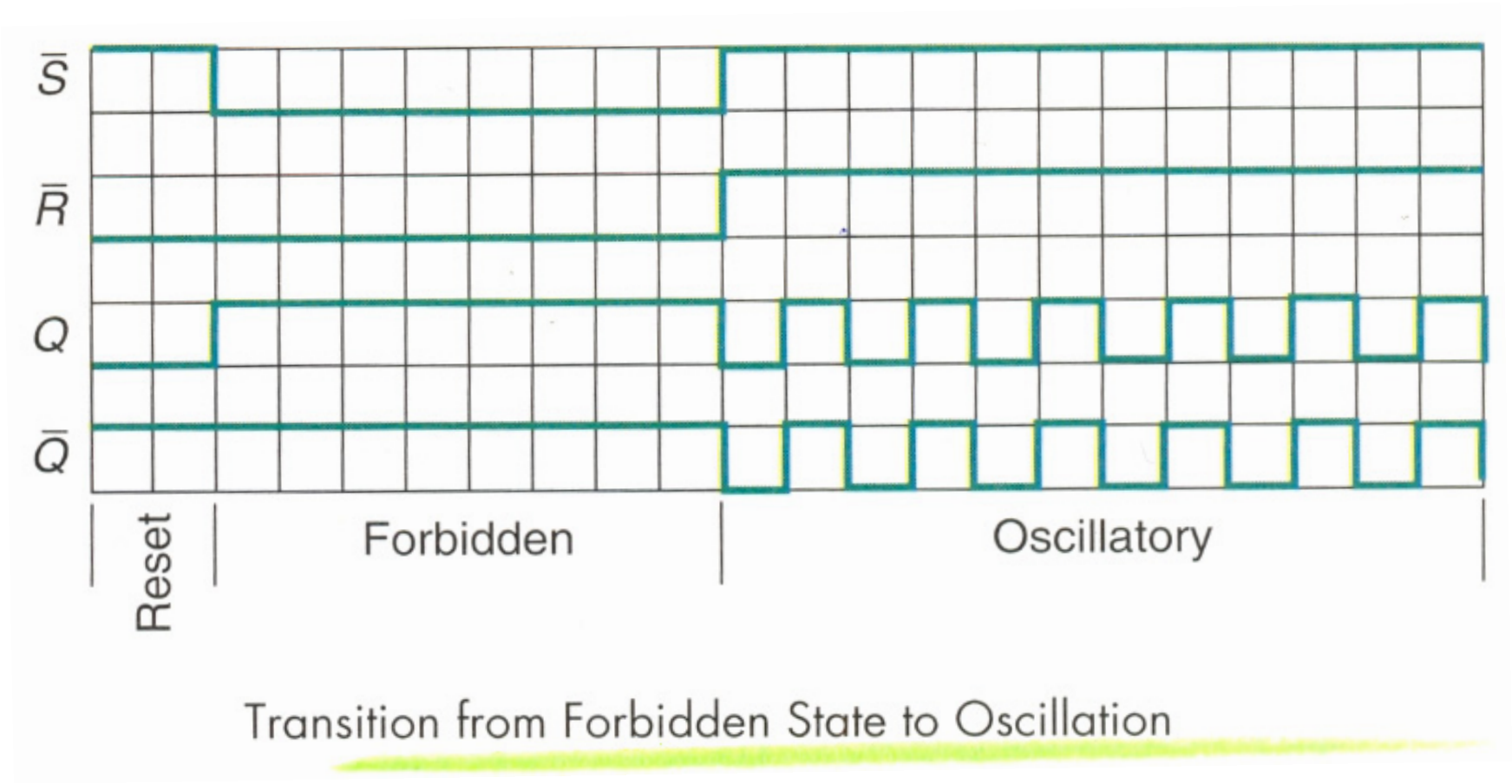


f) Output logic levels cross circuit via feedback lines. Cycle repeats and circuit oscillates.

Transition from Forbidden State to Oscillation

NAND Latch Operation

Forbidden-to-Oscillation -2



Transition from Forbidden State to Oscillation

NAND/NOR Latch Function Table

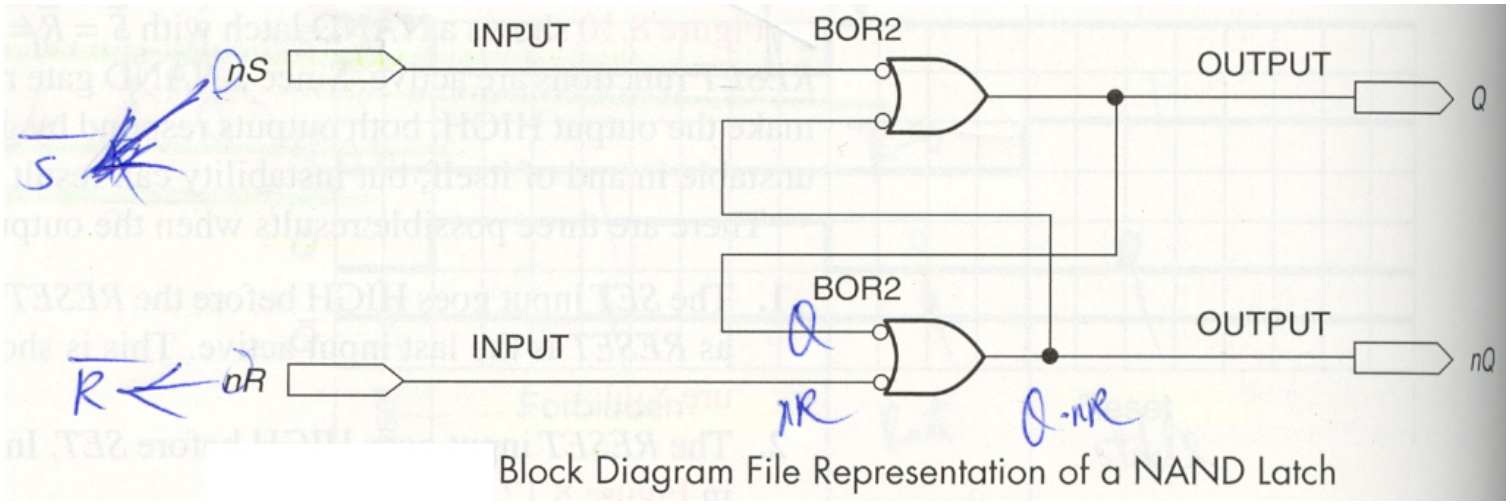
NAND Latch Function Table

$\bar{S}$	$\bar{R}$	Q_{t+1}	$\bar{Q}_{t+1}$	**Function**
0	0	1	1	Forbidden
0	1	1	0	Set
1	0	0	1	Reset
1	1	Q_t	$\bar{Q}_t$	No change

NOR Latch Function Table

S	R	Q_{t+1}	$\bar{Q}_{t+1}$	**Function**
0	0	Q_t	$\bar{Q}_t$	No change
0	1	0	1	Reset
1	0	1	0	Set
1	1	0	0	Forbidden

Practical Synthesis of a NAND Latch in Quartus II



Block Diagram File Representation of a NAND Latch

- Gate components are called BOR2:
  Bubbled-OR, 2-inputs

- Inputs are labeled *nS* and *nR*.

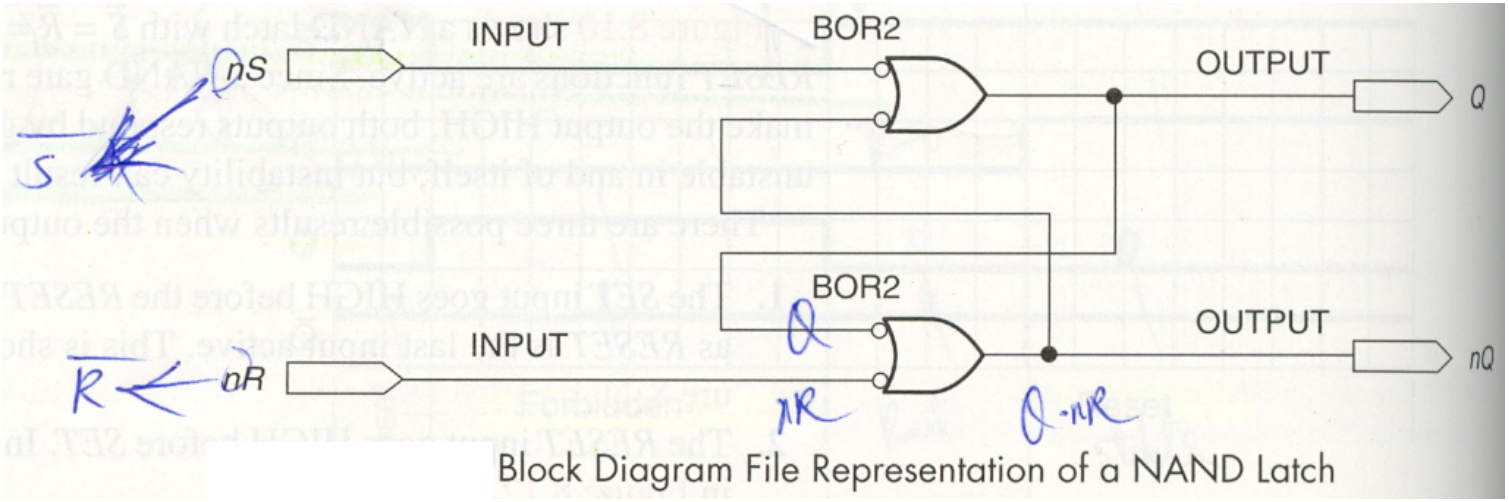
- Outputs are labeled *Q* and *nQ*.
  In Quartus, the *n* prefix takes the place of the logic inversion bar.

Practical Synthesis of the NAND Latch

- Quartus II does not synthesize the LATCH exactly as shown in Figure 8.15 on the previous slide. **(Quartus II合成的Latch跟前頁的圖不同)**

- Quartus II analyzes the Boolean equation of the original LATCH and reformats the circuit to fit the target device.

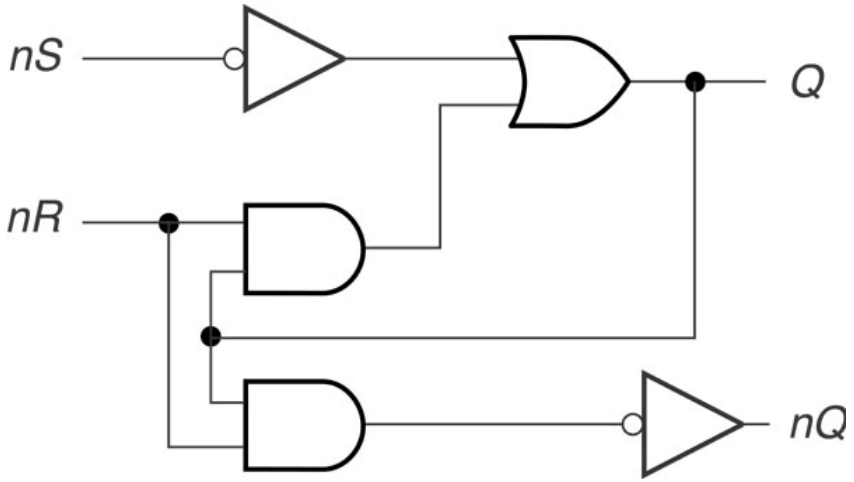
Quartus II NAND Latch Equations



Block Diagram File Representation of a NAND Latch

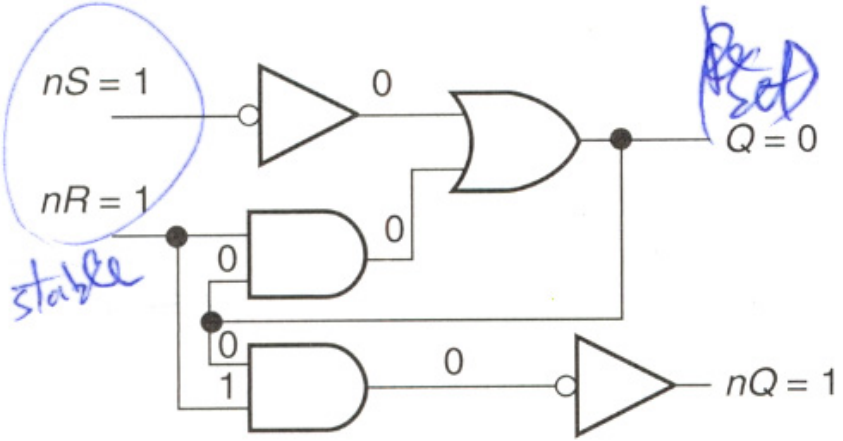
$$Q = Q \bullet nR + \overline{nS}$$

$$nQ = \overline{Q \bullet nR}$$

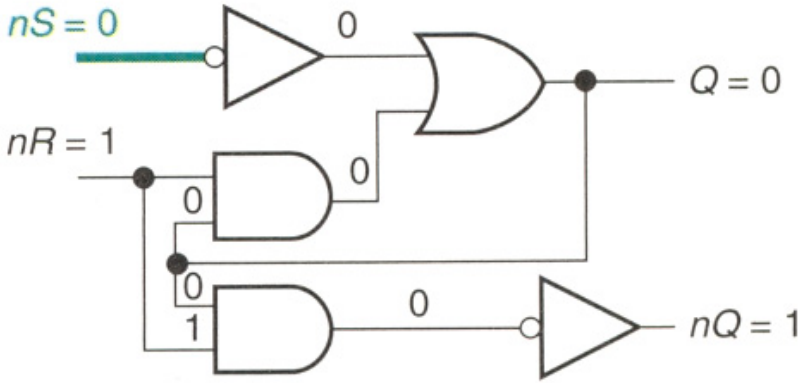


Quartus II NAND Latch

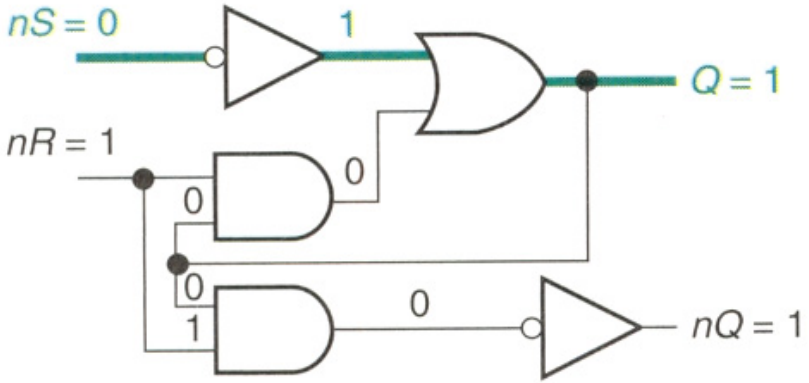
RESET-to-SET Transition -1



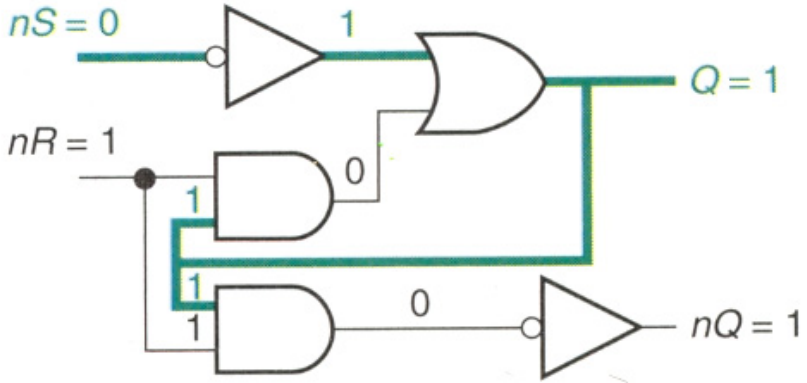
a) Stable in RESET state.



b) Set input activates.



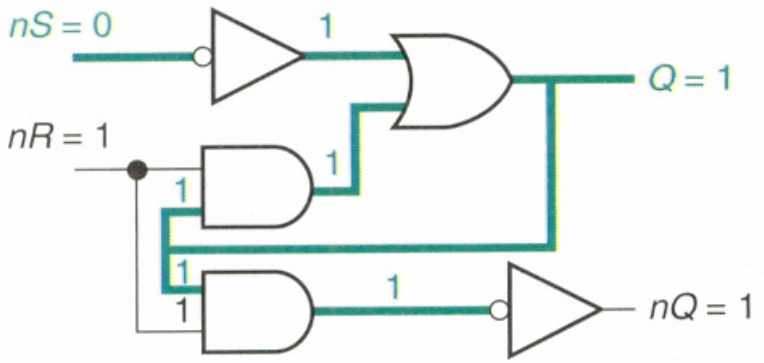
c) Change propagates through OR gate and sets $Q = 1$.



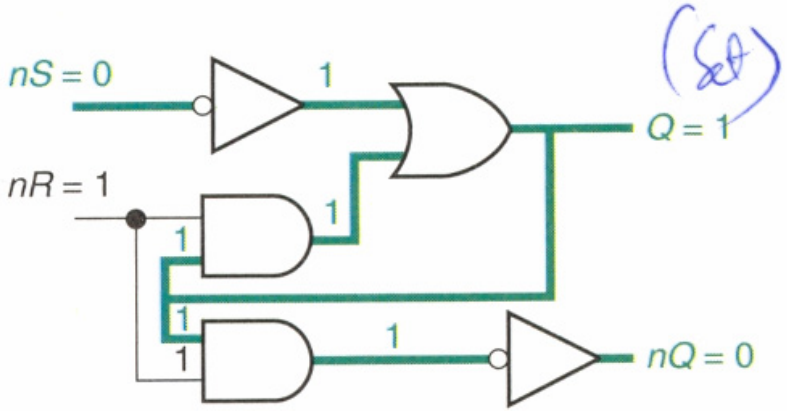
d) Feedback takes HIGH to AND gates.

Quartus II NAND Latch

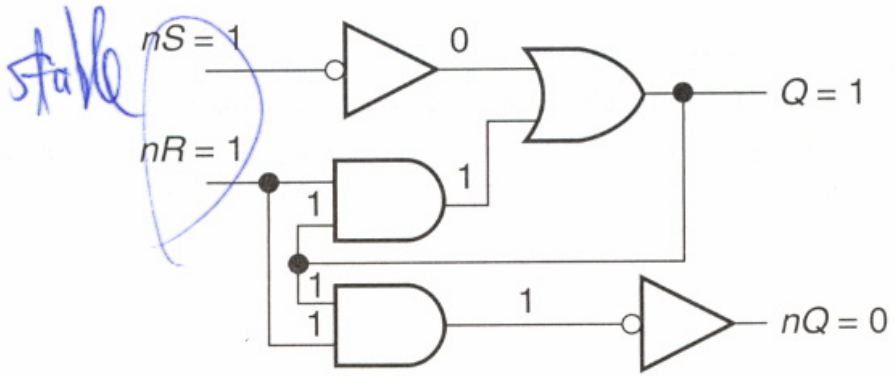
RESET-to-SET Transition -2



e) HIGH propagates through upper AND gate, completing latch transition.



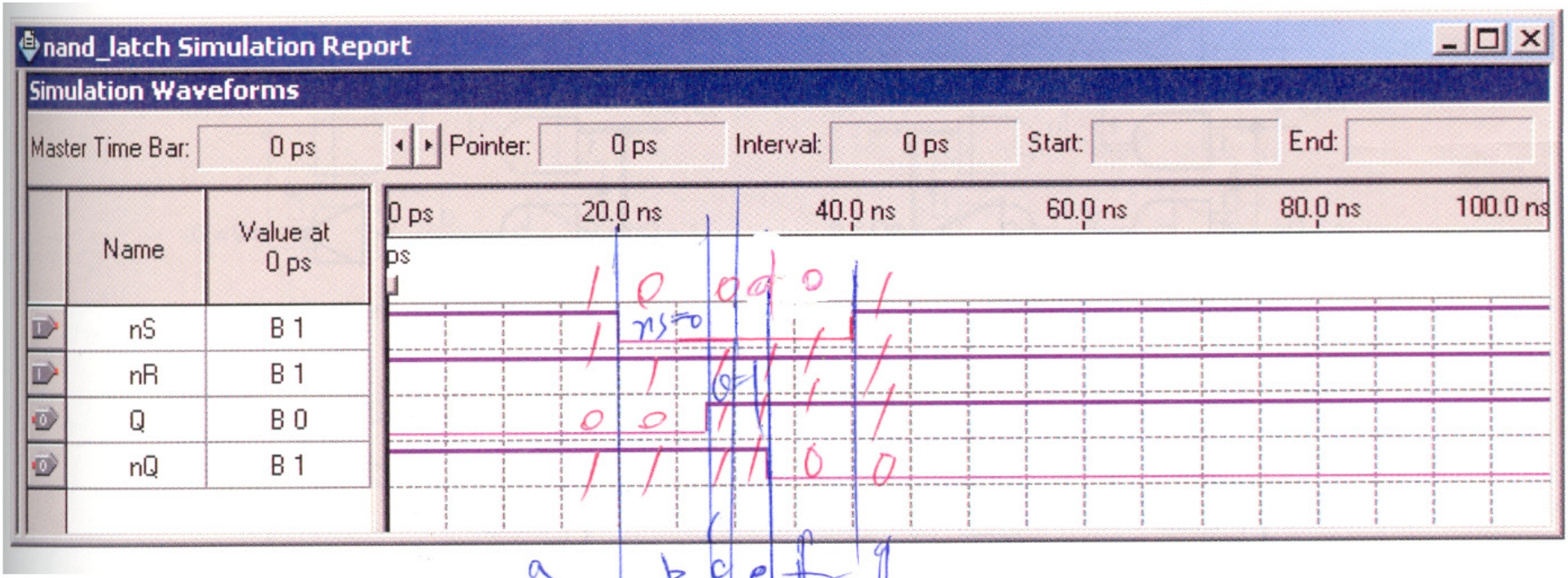
f) Change reaches *nQ*.



g) HIGH supplied to OR gate from AND gate keeps latch SET. LOW at *nS* can be removed.

Quartus II NAND Latch

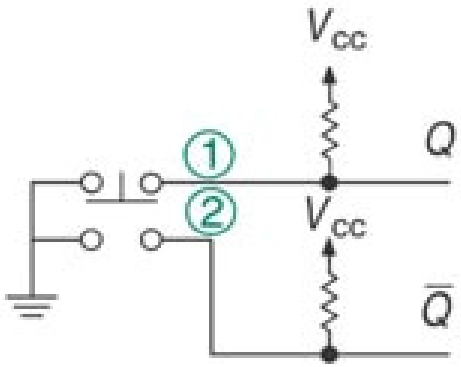
RESET-to-SET Transition -3



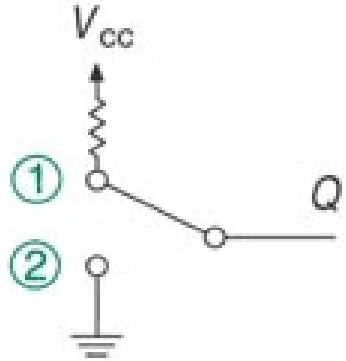
Waveforms for RESET-to-SET Transition of a NAND Latch Synthesized by Quartus II

Latch as a Switch Debouncer

- Switch as Pulse Generator

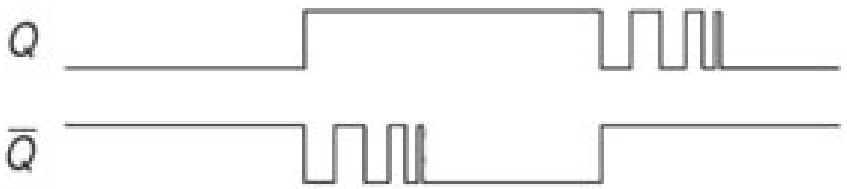


a. Pushbutton



b. Toggle

c. Ideal waveform



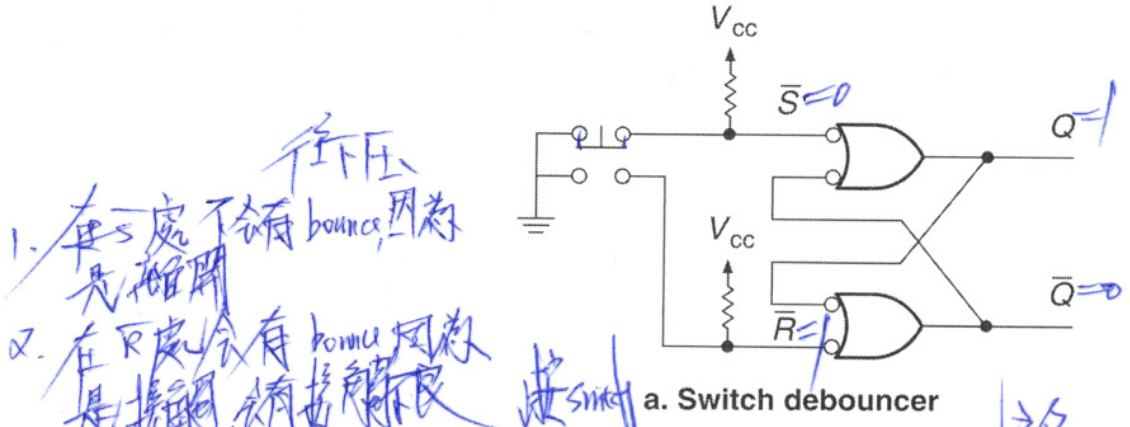
d. Effect of contact bounce

Switch Bounce

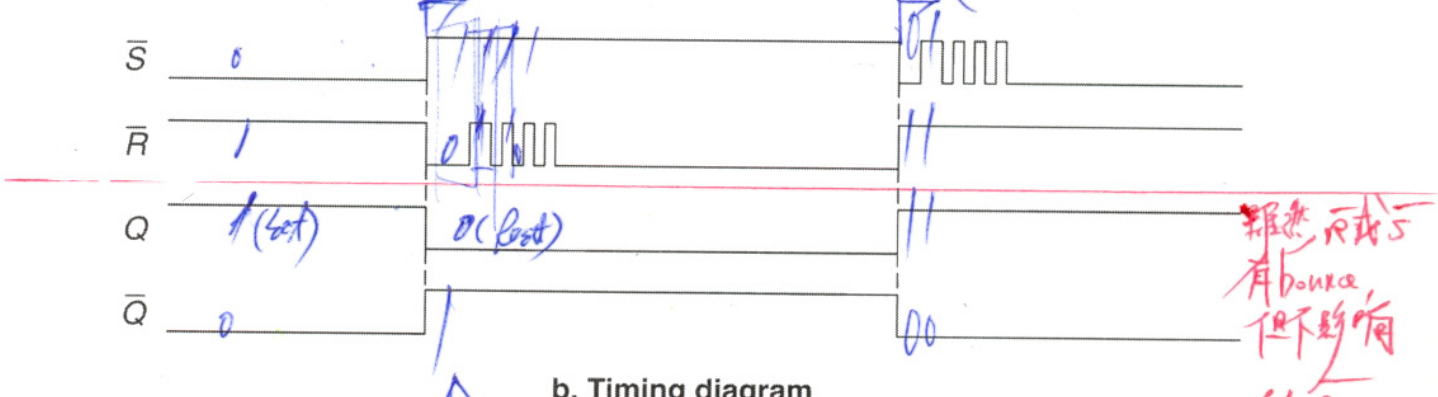
- The condition where the closure of a switch contact results in a **mechanical bounce** before the final contact is made.

- In logic circuits, switch bounce causes several pulses when a switch is closed.
  - Can cause circuit to behave unpredictably.

NAND Latch ad a Switch Debouncer



a. Switch debouncer



b. Timing diagram

$\overline{S}$	$\overline{R}$	Action (NAND Latch)
0	0	Both *SET* and *RESET* active: forbidden condition
0	1	*SET* input active
1	0	*RESET* input active
1	1	Neither *SET* nor *RESET* active; output does not change from previous state

8.3 Gated Latches

■ KEY TERMS

Gated SR Latch An SR latch whose ability to change states is controlled by an extra input called the *ENABLE* input.

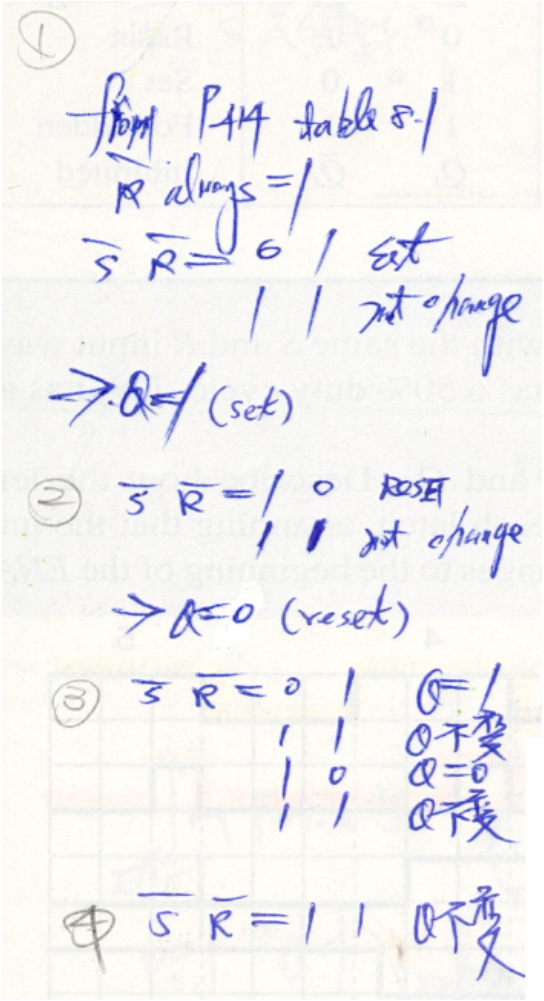
Steering Gates Logic gates, controlled by the *ENABLE* input of a gated latch, that steer a *SET* or *RESET* pulse to the correct input of an SR latch circuit.

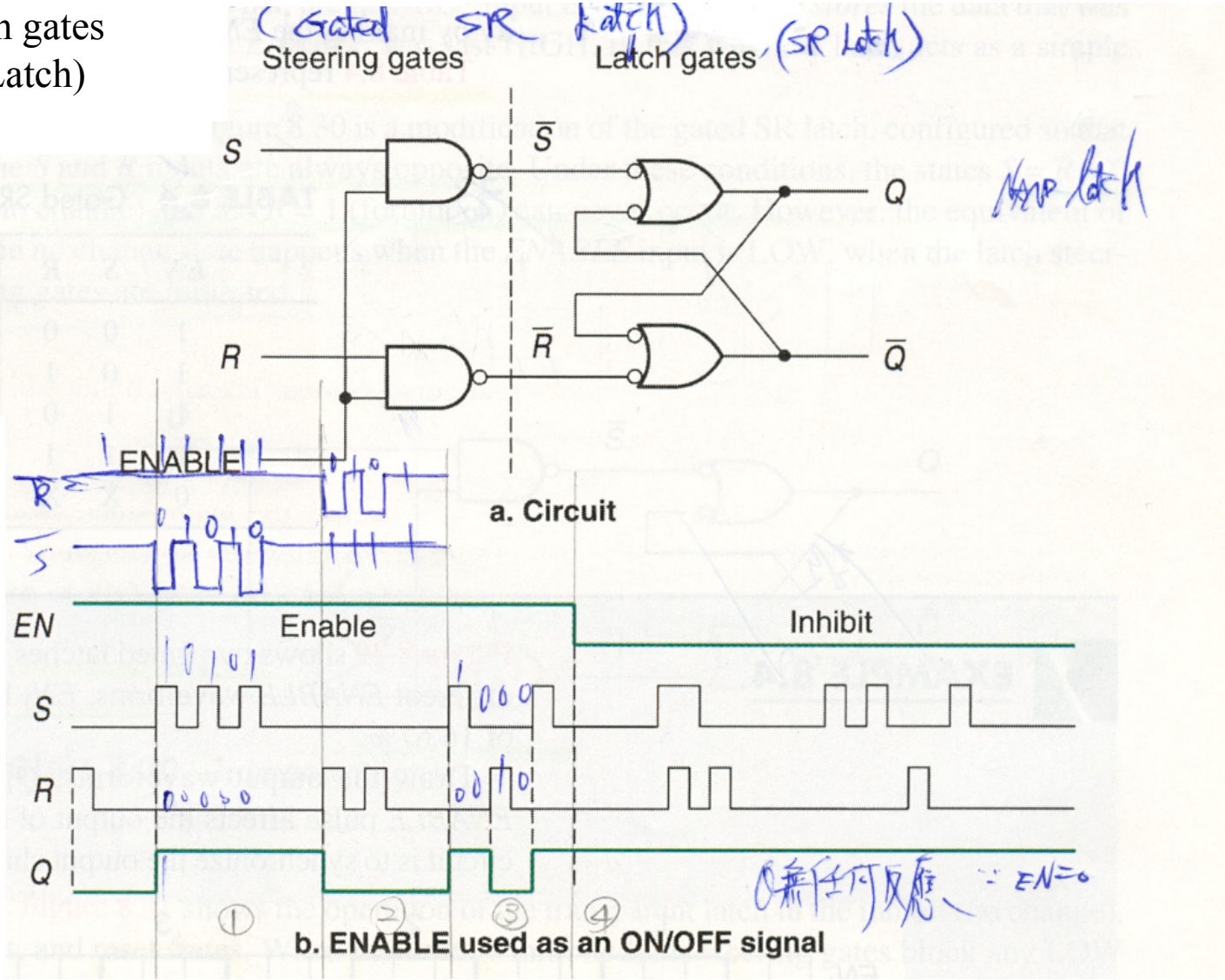
Transparent Latch (Gated D Latch) A latch whose output follows its data input when its *ENABLE* input is active.

Gated SR Latch -1

$\bar{S}$	$\bar{R}$	Action (NAND Latch)
0	0	Both *SET* and *RESET* active; forbidden condition
0	1	*SET* input active
1	0	*RESET* input active
1	1	Neither *SET* nor *RESET* active; output does not change from previous state

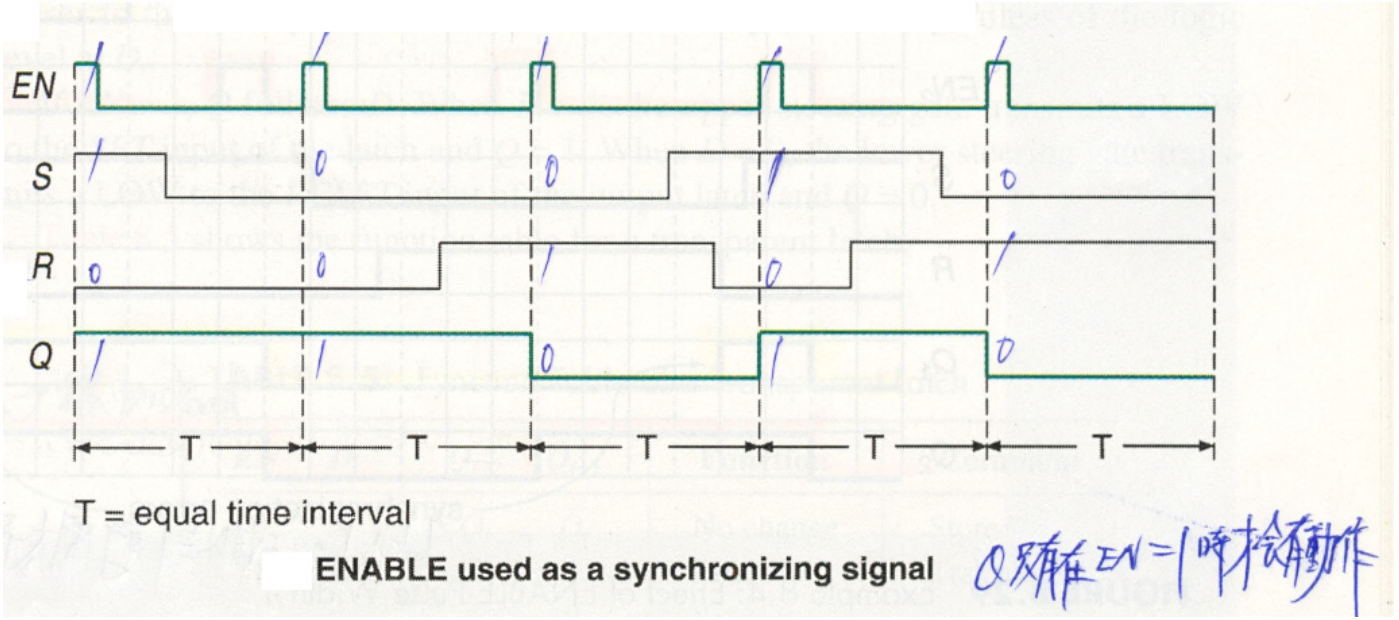
- Gated SR Latch=Steering gates + Latch gates (SR Latch)





Gated SR Latch -2

$\bar{S}$	$\bar{R}$	Action (NAND Latch)
0	0	Both *SET* and *RESET* active; forbidden condition
0	1	*SET* input active
1	0	*RESET* input active
1	1	Neither *SET* nor *RESET* active; output does not change from previous state



Latch ENABLE Input

- Used in two principal ways:
  - As an ON/OFF signal 當作On/Off訊號
  - As a synchronizing signal 當作同步訊號

Gated SR Latch Function

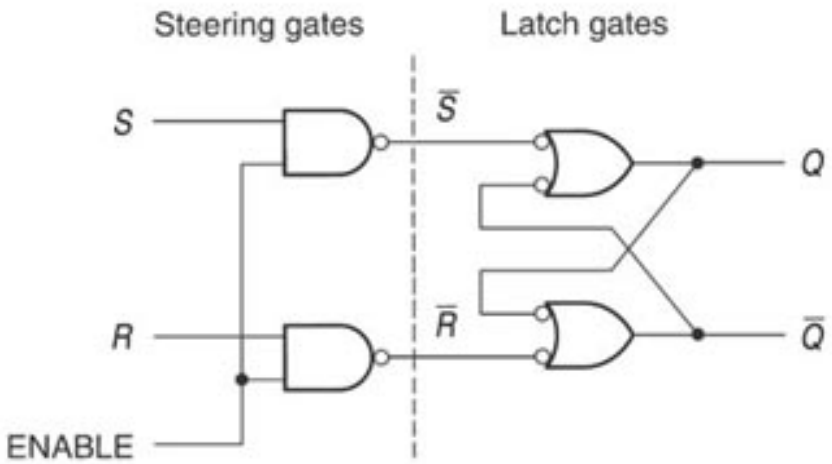
EN	*S*	*R*	Q_{t+1}	$\overline{Q}_{t+1}$	**Function**
1	0	0	Q_t	$\overline{Q}_t$	No change
1	0	1	0	1	Reset
1	1	0	1	0	Set
1	1	1	1	1	Forbidden
0	**X**	**X**	Q_t	$\overline{Q}_t$	**Inhibited**

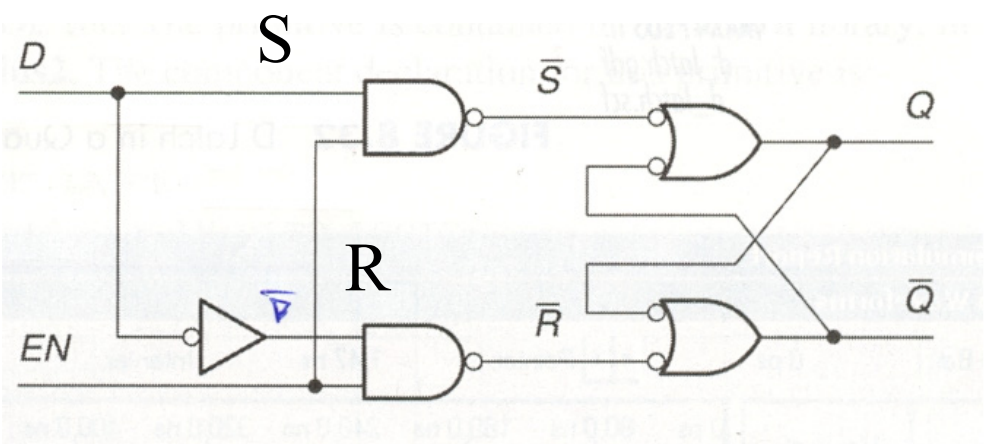
Transparent Latch (Gated D Latch)

- A latch whose output follows its data input when its ENABLE input is active.

- When ENABLE is inactive, the latch stores the data that was present when ENABLE was last active.

Transparent Latch

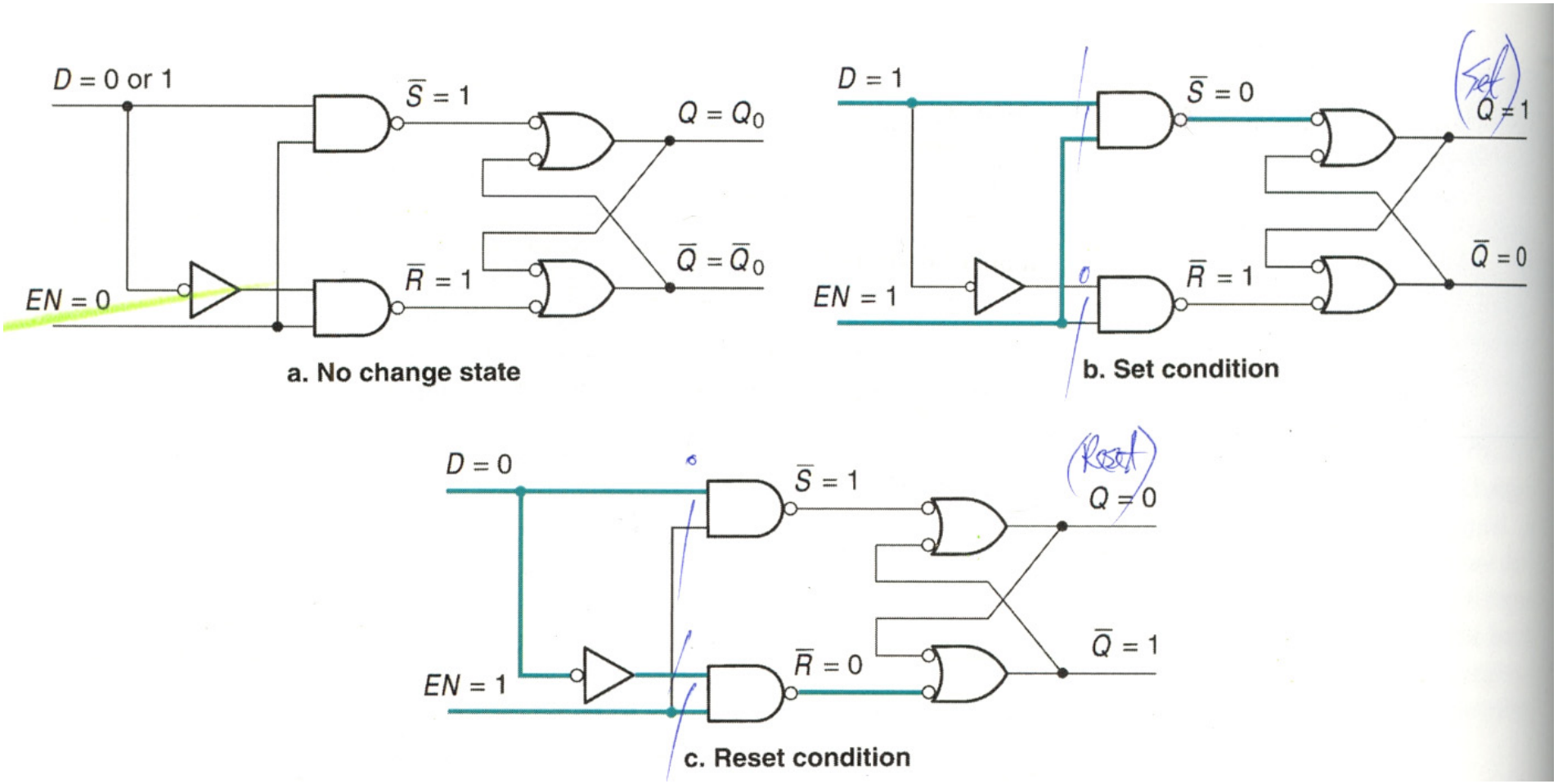




Gated SR Latch ***Transparent Latch***

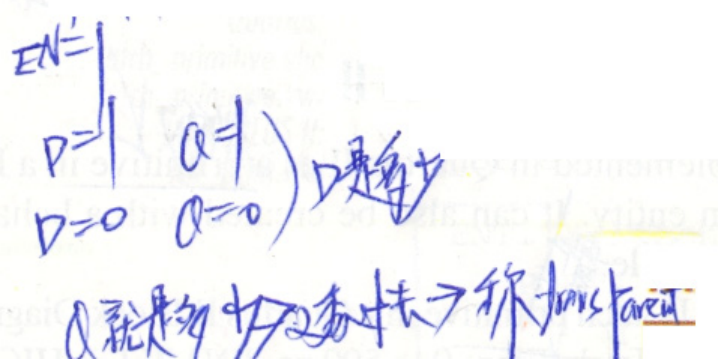
the S and R inputs are always opposite. Under these conditions, the states $S = R = 0$ (no change) and $S = R = 1$ (forbidden) can never occur.

Operation of Transparent Latch

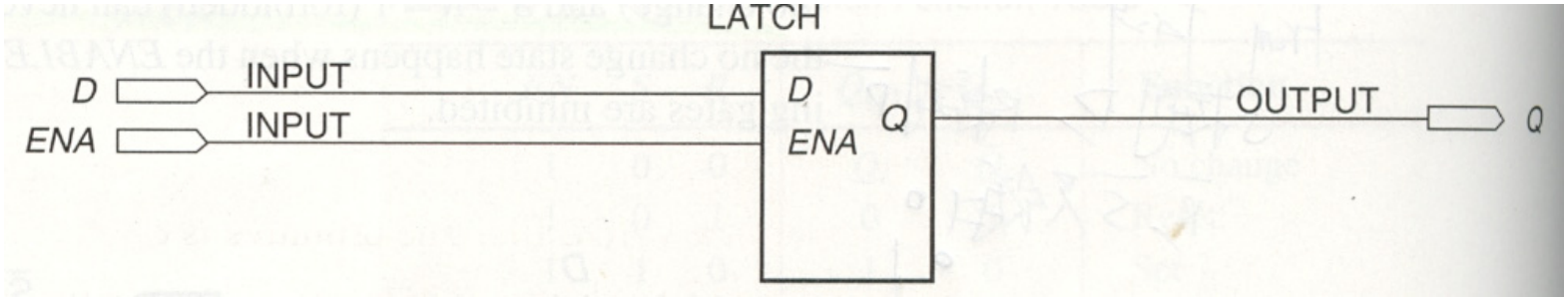


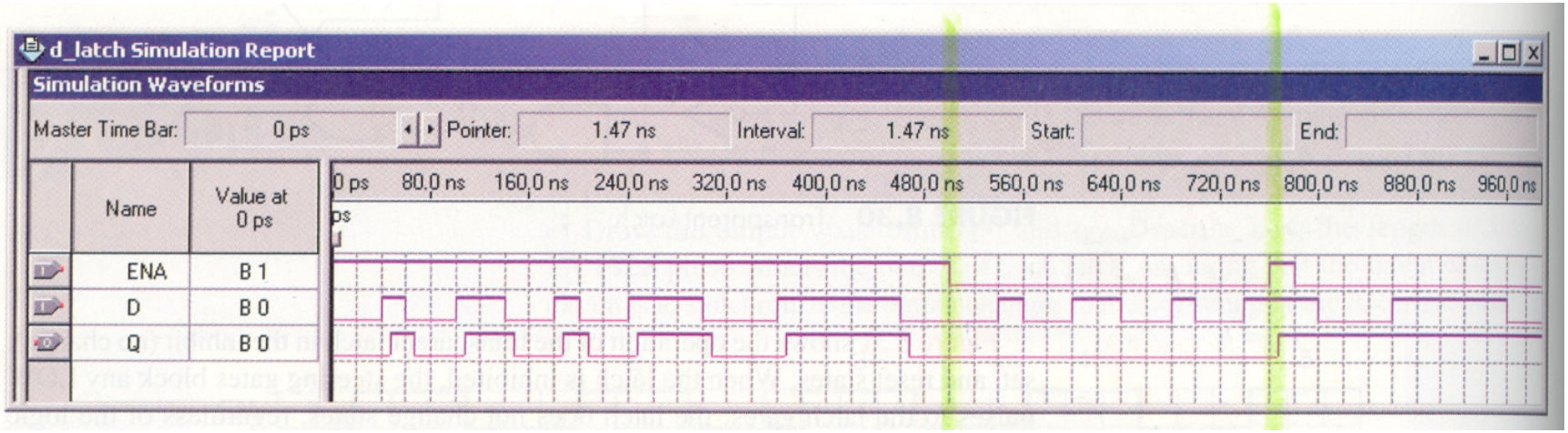
Gated D Latch Function Table

EN	*D*	Q_{t+1}	$\bar{Q}_{t+1}$	Function	Comment
0	X	Q_t	$\bar{Q}_t$	No change	Store
1	0	0	1	Reset	Transparent
1	1	1	0	Set	



D Latches in Quartus II





From 0 to 500 ns, *ENABLE* is HIGH and the latch is in the transparent mode (*Q* follows *D*). When *ENABLE* goes LOW, the last value of *D* (0) is stored until *ENABLE* goes high again, just before 800 ns. When *ENABLE* goes LOW again, a new value of *D* (HIGH) is stored until the end of the simulation.

VHDL Process Statement

- PROCESS statement is concurrent.
- Statements inside the PROCESS are sequential.

D Latch in VHDL -1

```
-- d_latch_vhdl.vhd
-- D latch with active-HIGH level-sensitive enable

ENTITY d_latch_vhdl IS
   PORT(
      d, ena : IN  BIT;
      q      : OUT BIT);
END d_latch_vhdl;

ARCHITECTURE a OF d_latch_vhdl IS
BEGIN
   PROCESS (d, ena)
   BEGIN
      IF (ena = '1') THEN
          q  <= d;
      END IF;
   END PROCESS;
END a;
```

D Latch in VHDL
Instantiating a Latch Primitive -1

- Primitive is contained in the Altera library, in a package called maxplus2.
- Component declaration in maxplus2 package.
- Unnecessary to declare it in the file used.

The component declaration for this primitive is:

```
COMPONENT LATCH
    PORT (d   : IN  STD_LOGIC;
          ena : IN  STD_LOGIC;
          q   : OUT STD_LOGIC);
END COMPONENT;
```

D Latch in VHDL
Instantiating a Latch Primitive -2

```
-- latch_primitive.vhd
-- D latch with active-HIGH level-sensitive enable

LIBRARY ieee;                                  Required for STD_LOGIC types
USE ieee.std_logic_1164.ALL;
LIBRARY altera;                                Required for latch component
USE altera.maxplus2.ALL;

ENTITY latch_primitive IS
   PORT(
        d_in, enable : IN  STD_LOGIC;
        q_out        : OUT STD_LOGIC);
END latch_primitive;

ARCHITECTURE a OF latch_primitive IS
BEGIN
   -- Instantiate a latch from a primitive
   latch_prim: latch
      PORT MAP (d    => d_in,
                ena  => enable,
                q    => q_out);                User names (defined in
END a;                                         entity declaration)
                                               Component names (defined in maxplus2 package)
```

Multibit Latches in VHDL -1

- **VHDL can be used to implement latches with multiple *D* inputs and *Q* outputs and a common *ENABLE* line.(可以用下列3種方法完成)**
  - **Use behavioral description with STD_LOGIC_VECTOR types.**
  - **Use primitives – predefined components.**
  - **Use component from Library of Parameterized Modules (LPM).**

Multibit Latches in VHDL -2

Behavioral Description

```
-- latch4_behavioral.vhd
-- D latch with active-HIGH level-sensitive enable

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY latch4_behavioral IS
   PORT(d     : IN  STD_LOGIC_VECTOR (3 downto 0);
      enable : IN  STD_LOGIC;
      q      : OUT STD_LOGIC_VECTOR (3 downto 0));
END latch4_behavioral;

ARCHITECTURE a OF latch4_behavioral IS
BEGIN
   PROCESS (enable, d)
   BEGIN
      IF (enable = "1") THEN
         q <= d;
      END IF;
   END PROCESS;
END a;
```

Multibit Latches in VHDL -3

4 LATCH Primitives and a GENERATE Statement

```
-- latch4_primitive.vhd
-- D latch with active-HIGH level-sensitive enable

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
LIBRARY altera;
USE altera.maxplus2.ALL;

ENTITY latch4_primitive IS
   PORT(d_in  : IN  STD_LOGIC_VECTOR (3 downto 0);
        enable : IN  STD_LOGIC;
        q_out  : OUT STD_LOGIC_VECTOR (3 downto 0));
END latch4_primitive

ARCHITECTURE a OF latch14_primitive IS
BEGIN
   -- Instantiate a latch primitive from maxplus2 package
   latch4:
   FOR i IN 3 downto 0 GENERATE
   latch_primitive: latch
      PORT MAP (d => d_in(i), ena => enable, q => q_out(i));
   END GENERATE;
END a;
```

Multibit Latches in VHDL -4

LPM Latch

```
-- latch4_lpm.vhd
-- 4-BIT D latch with active-HIGH level-sensitive enable
-- Uses a latch component from
-- the Library of Parameterized Modules (LPM)

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
LIBRARY lpm;
USE lpm.lpm_components.ALL;

ENTITY latch4_lpm IS
   PORT(d_in : IN  STD_LOGIC_VECTOR (3 downto 0);
      enable : IN  STD_LOGIC;
      q_out  : OUT STD_LOGIC_VECTOR (3 downto 0));
END latch4_lpm;

ARCHITECTURE a OF latch4_lpm IS
BEGIN
   -- Instantiate latch from an LPM component
   latch4: lpm_latch
      GENERIC MAP (LPM_WIDTH => 4)
      PORT MAP ( data   => d_in,
                 gate   => enable,
                 q      => q_out);
END a;
```

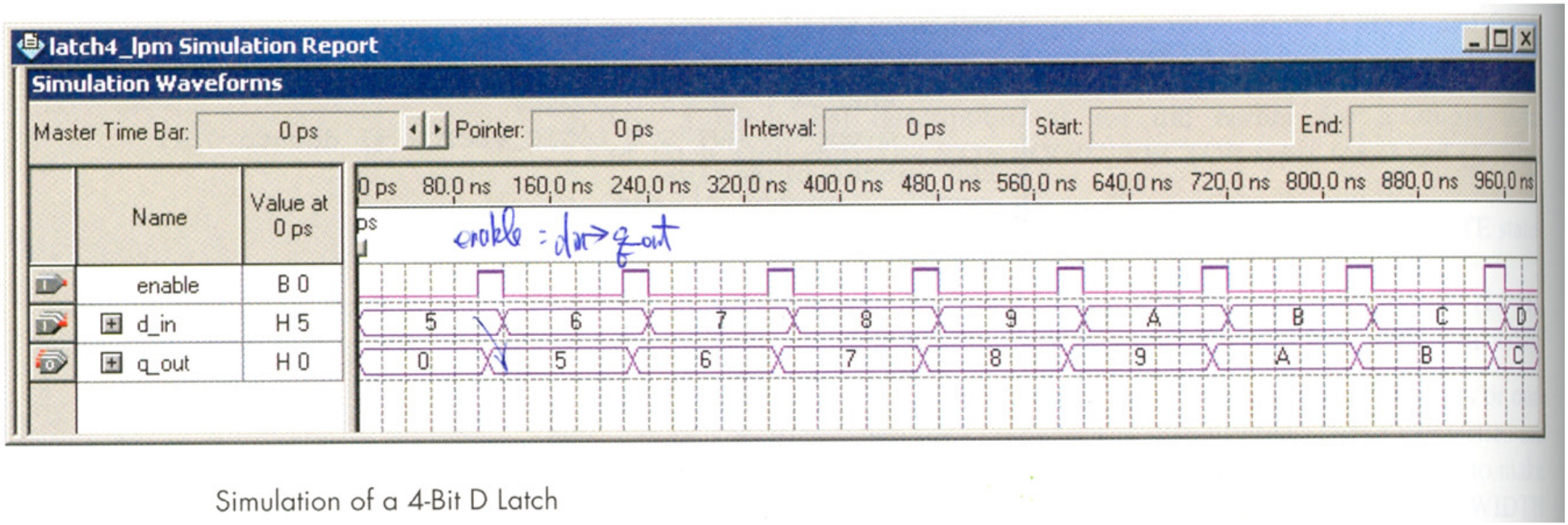
Required for STD_LOGIC types

Required for LPM components

Parameters assigned with GENERIC MAP

Ports assigned with PORT MAP

Multibit Latches in VHDL -5

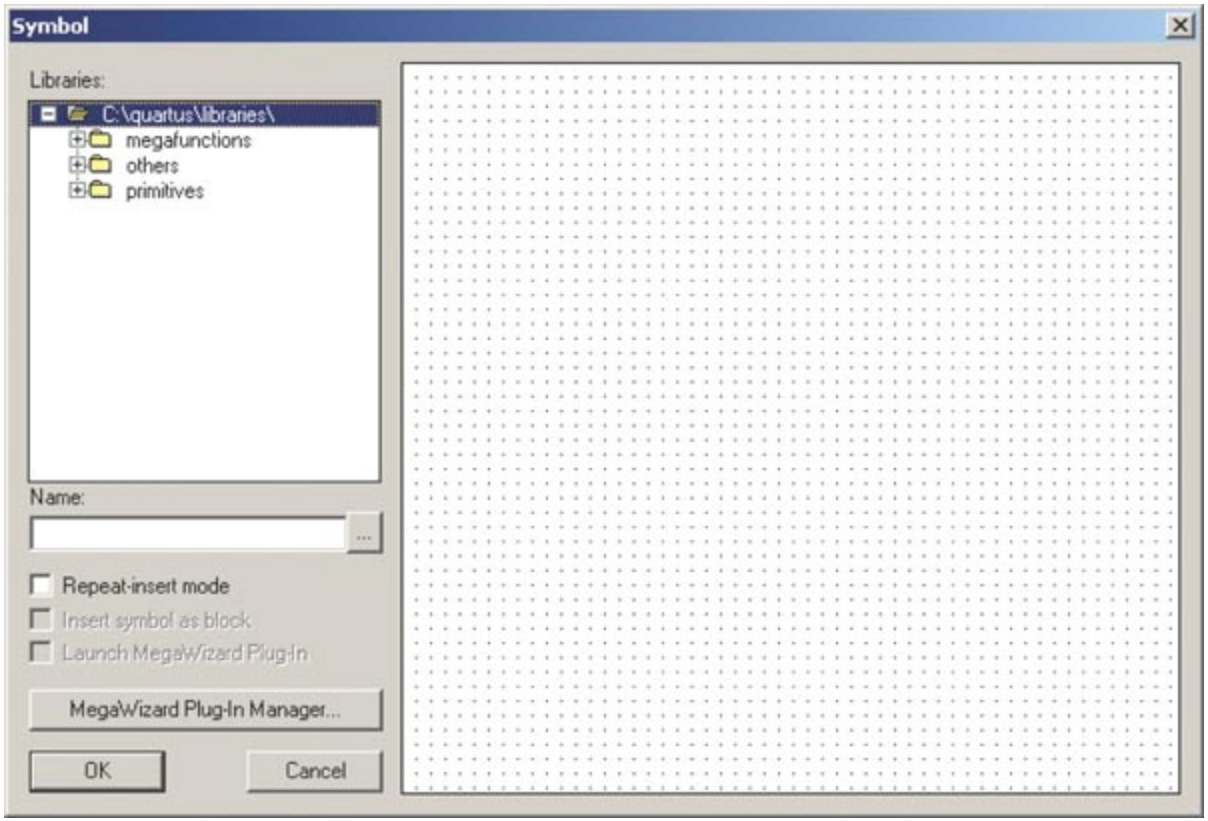


Simulation of a 4-Bit D Latch

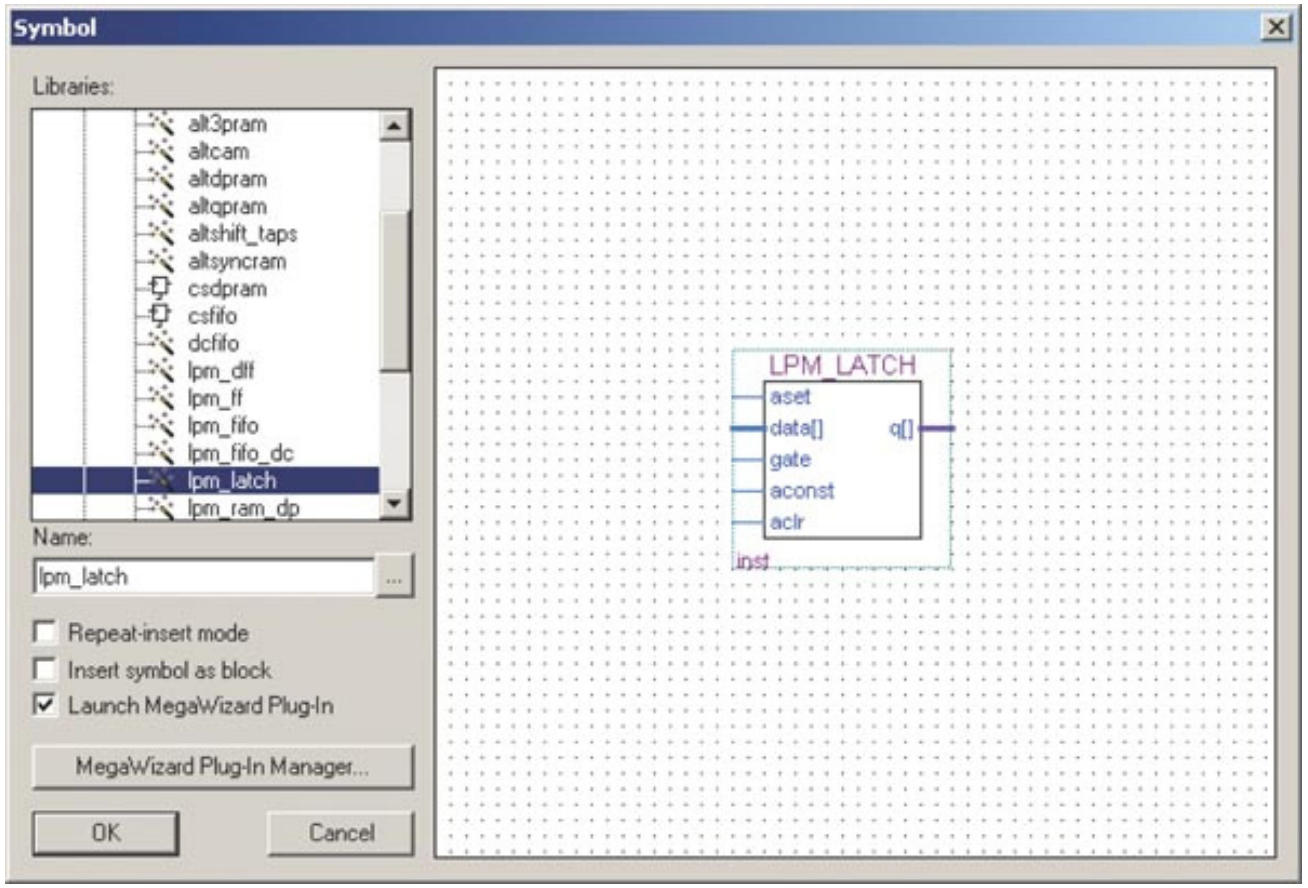
Multibit Latches in Quartus II Block Editor -1

KEY TERM

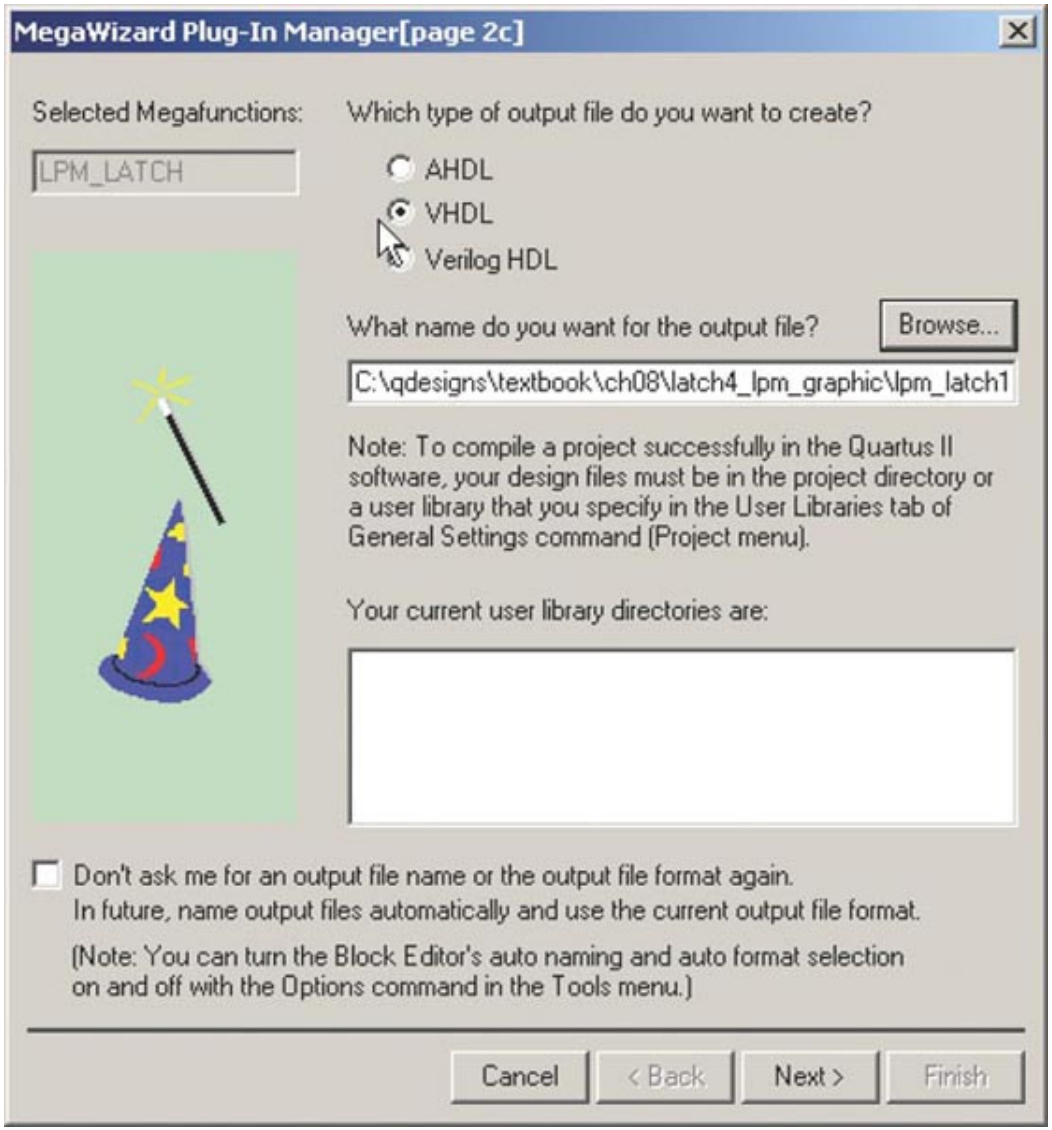
MegaWizard Plug-In Manager A tool provided with Quartus II (and MAX+PLUS II) that allows the user to automatically set the ports and parameter values of an LPM component for use in an HDL or Block Diagram File.



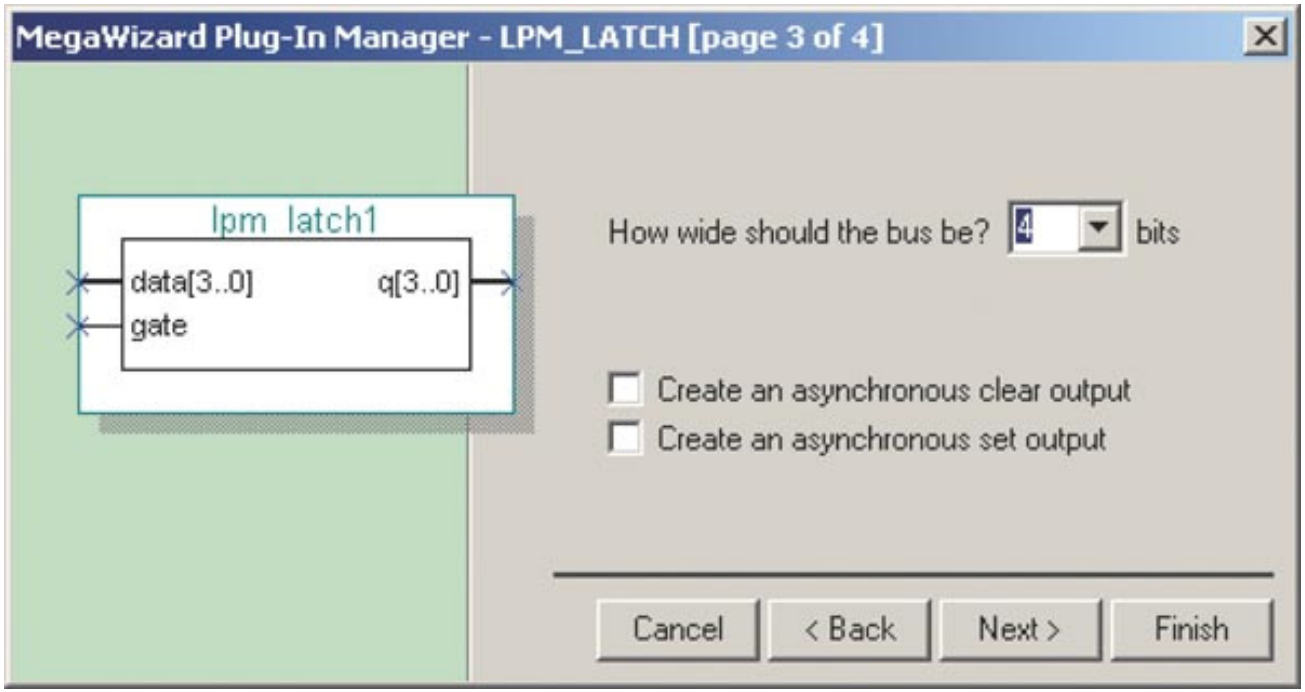
Multibit Latches in Quartus II Block Editor -2



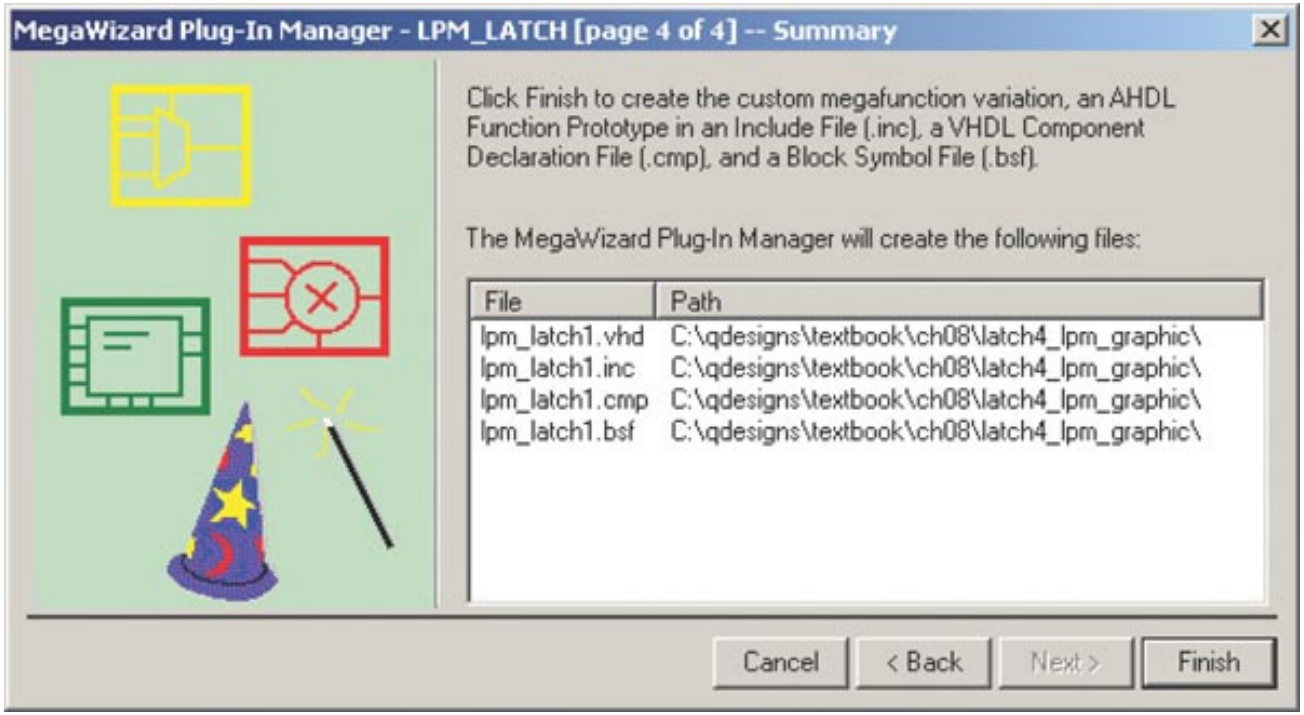
Multibit Latches in Quartus II Block Editor -3



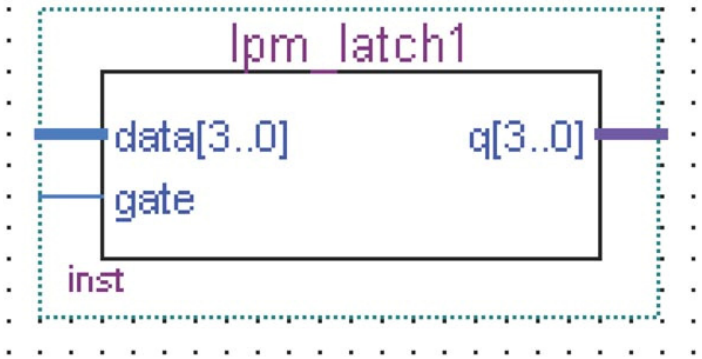
Multibit Latches in Quartus II Block Editor -4

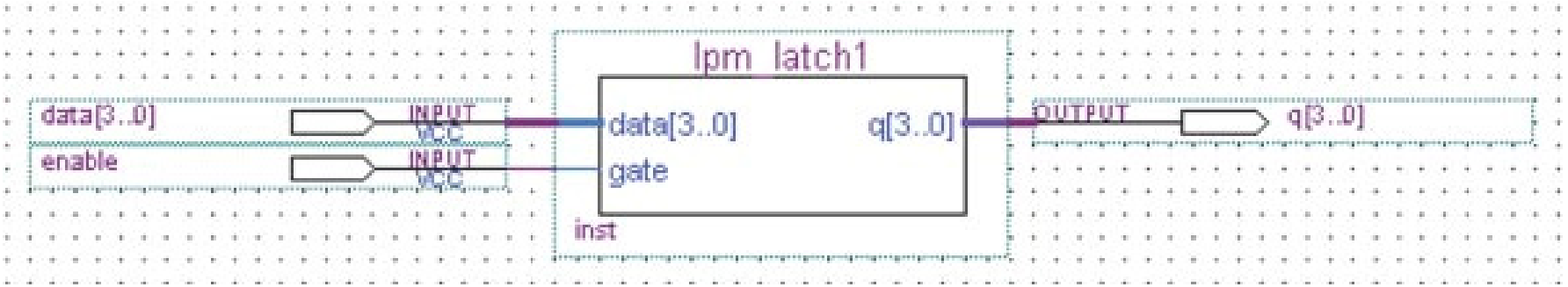


Multibit Latches in Quartus II Block Editor -5



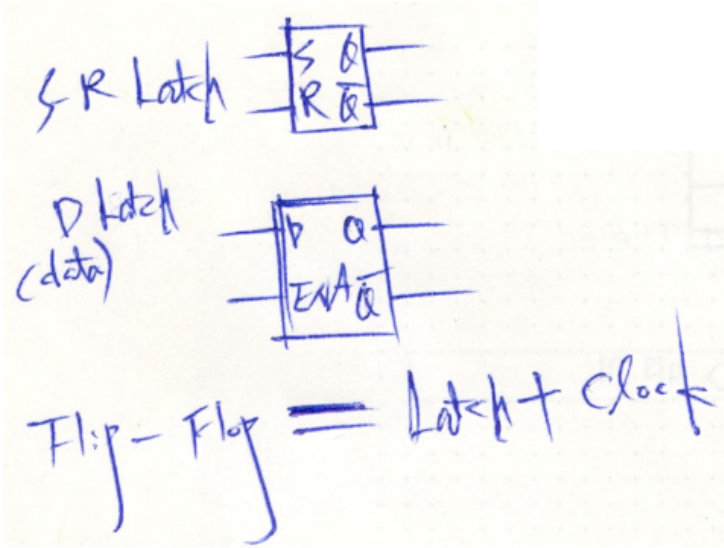
Multibit Latches in Quartus II Block Editor -6





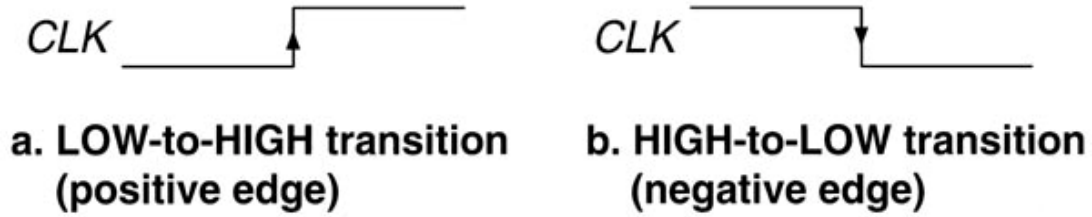
8.4 Edge-Triggered D Flip-Flops

- Flip-flop: A gated latch with a clock input.
- The sequential circuit output changes when its *CLOCK* input detects an edge.
- **Edge-sensitive** instead of level-sensitive.



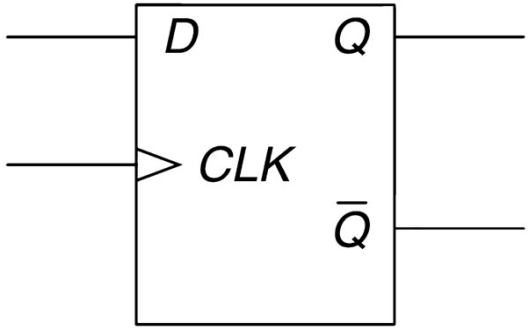
CLOCK *Definitions*

- **Positive edge**:
  - The transition from logic '0' to logic '1'
- **Negative edge**:
  - The transition from logic '1' to logic '0'
- Symbol is a triangle on the *CLK* (clock) input of a flip-flop.

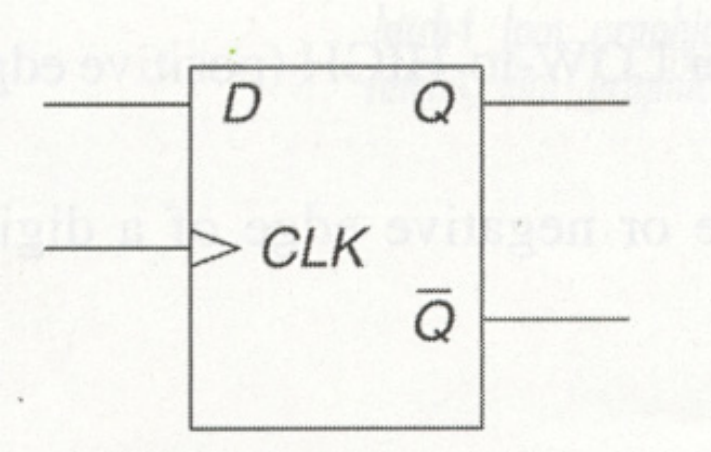


a. LOW-to-HIGH transition (positive edge)

b. HIGH-to-LOW transition (negative edge)



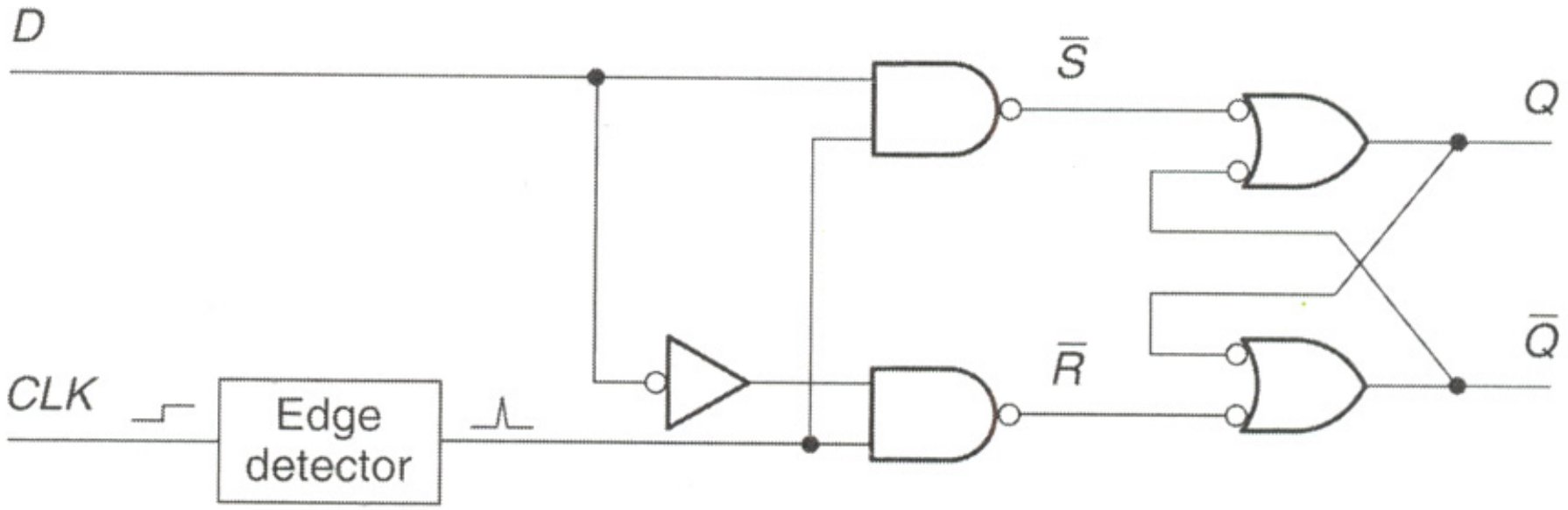
Positive Edge-Triggered D Flip-Flop Function Table



Function Table for a Positive Edge-Triggered D Flip-Flop

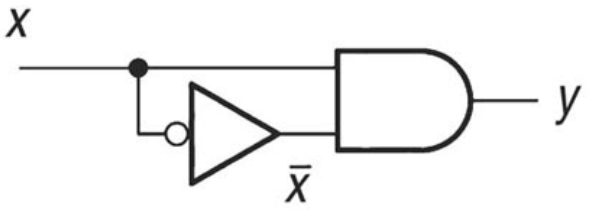
CLK	*D*	Q_{t+1}	$\bar{Q}_t + 1$	**Function**
↑	0	0	1	Reset
↑	1	1	0	Set
0	X	Q_t	$\bar{Q}_t$	Inhibited
1	X	Q_t	$\bar{Q}_t$	Inhibited
↓	X	Q_t	$\bar{Q}_t$	Inhibited

D Flip-Flop Equivalent Circuit

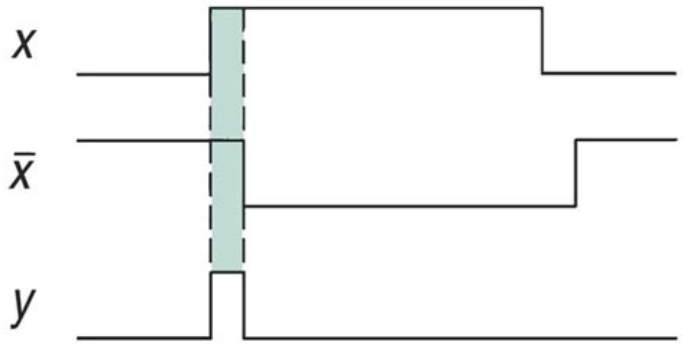


Edge Detector

- A circuit that converts that active-edge of a *CLOCK* input into a brief active-level pulse.
- Created using gate propagation delays.
- Can be positive or negative edge.

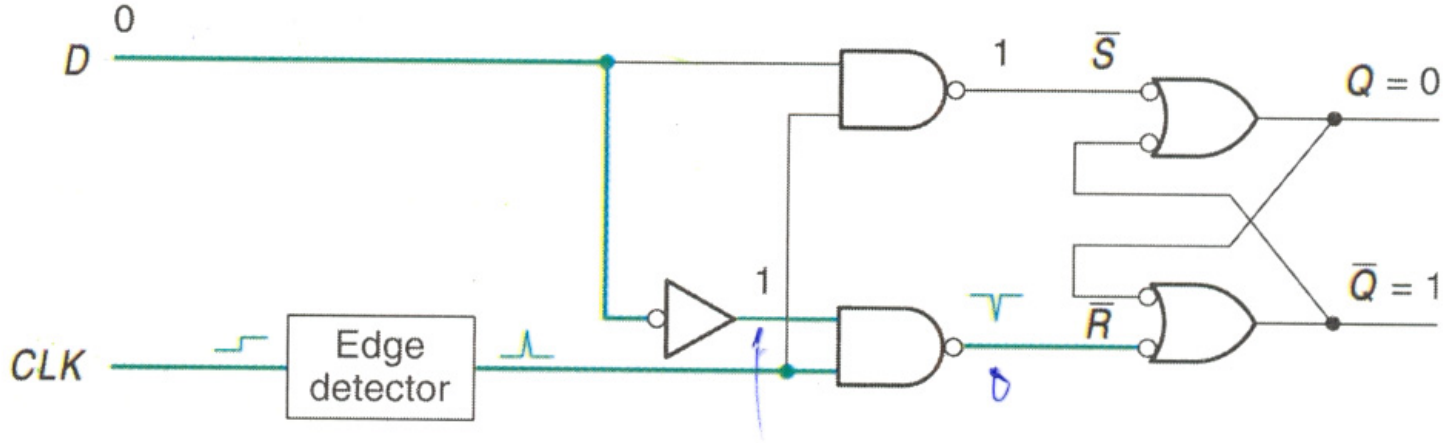


a. Simplified circuit

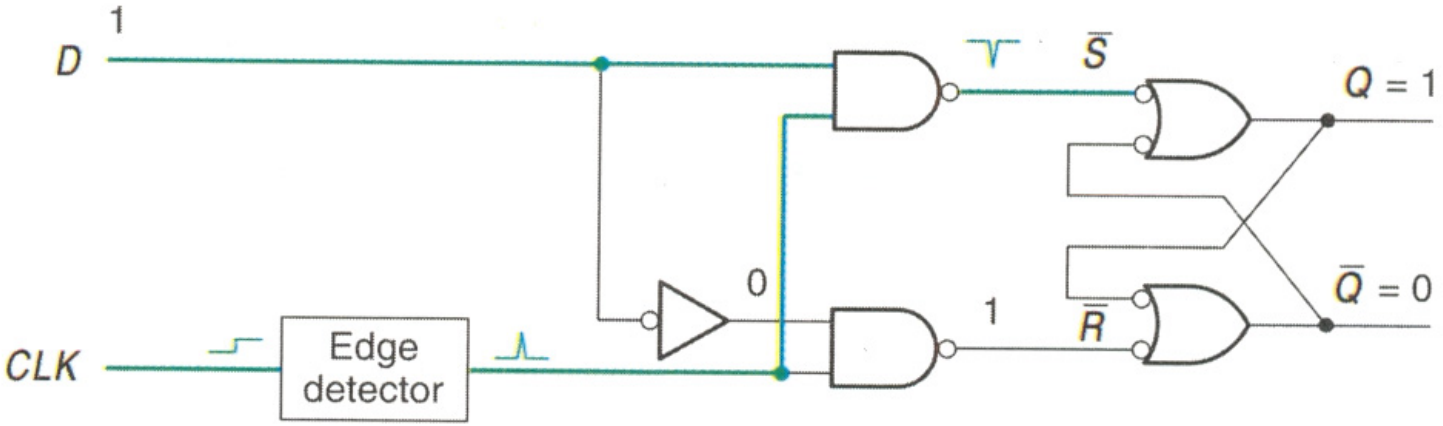


b. Waveforms

Operation of a D Flip-Flop

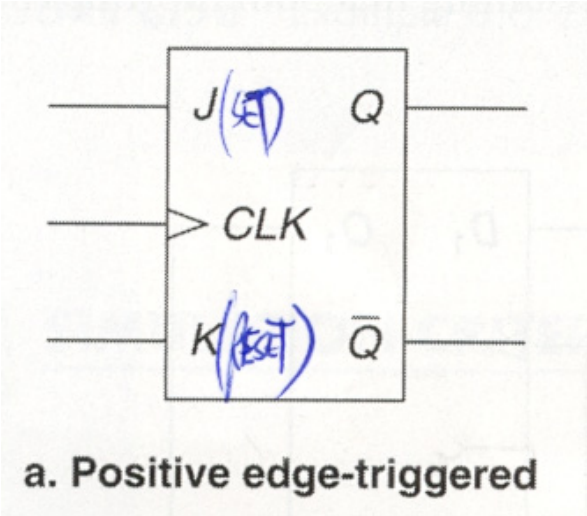


a. Reset action

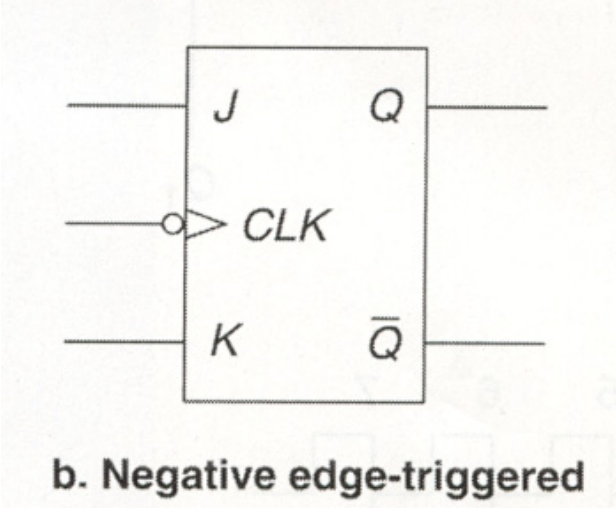


b. Set action

8.5 Edge-Triggered JK Flip-Flop



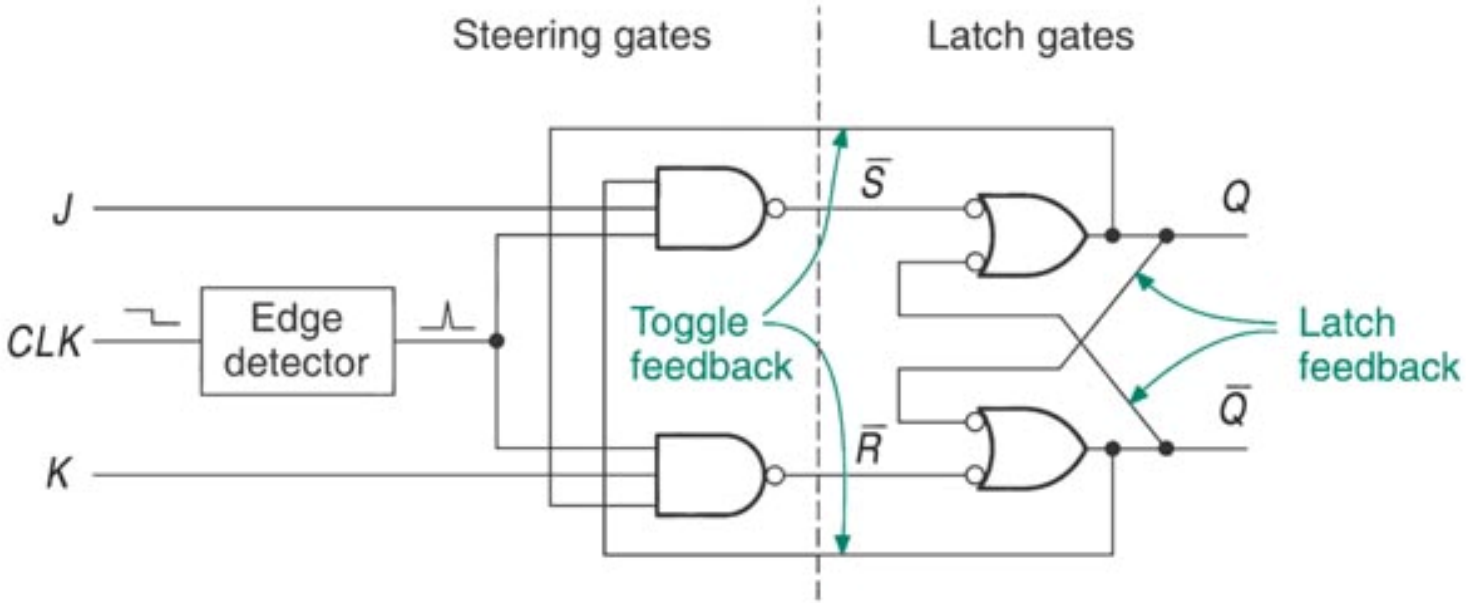
a. Positive edge-triggered



b. Negative edge-triggered

Positive Edge-Triggered						Negative Edge-Triggered					
CLK	*J*	*K*	Q_{t+1}	$\bar{Q}_{t+1}$	**Function**	*CLK*	*J*	*K*	Q_{t+1}	$\bar{Q}_{t+1}$	**Function**
↑	0	0	Q_t	$\bar{Q}_t$	No change	↓	0	0	Q_t	$\bar{Q}_t$	No change
↑	0	1	0	1	Reset	↓	0	1	0	1	Reset
↑	1	0	1	0	Set	↓	1	0	1	0	Set
↑	1	1	$\bar{Q}_t$	Q_t	Toggle	↓	1	1	$\bar{Q}_t$	Q_t	Toggle
0	X	X	Q_t	$\bar{Q}_t$	Inhibited	0	X	X	Q_t	$\bar{Q}_t$	Inhibited
1	X	X	Q_t	$\bar{Q}_t$	Inhibited	1	X	X	Q_t	$\bar{Q}_t$	Inhibited
↓	X	X	Q_t	$\bar{Q}_t$	Inhibited	↑	X	X	Q_t	$\bar{Q}_t$	Inhibited

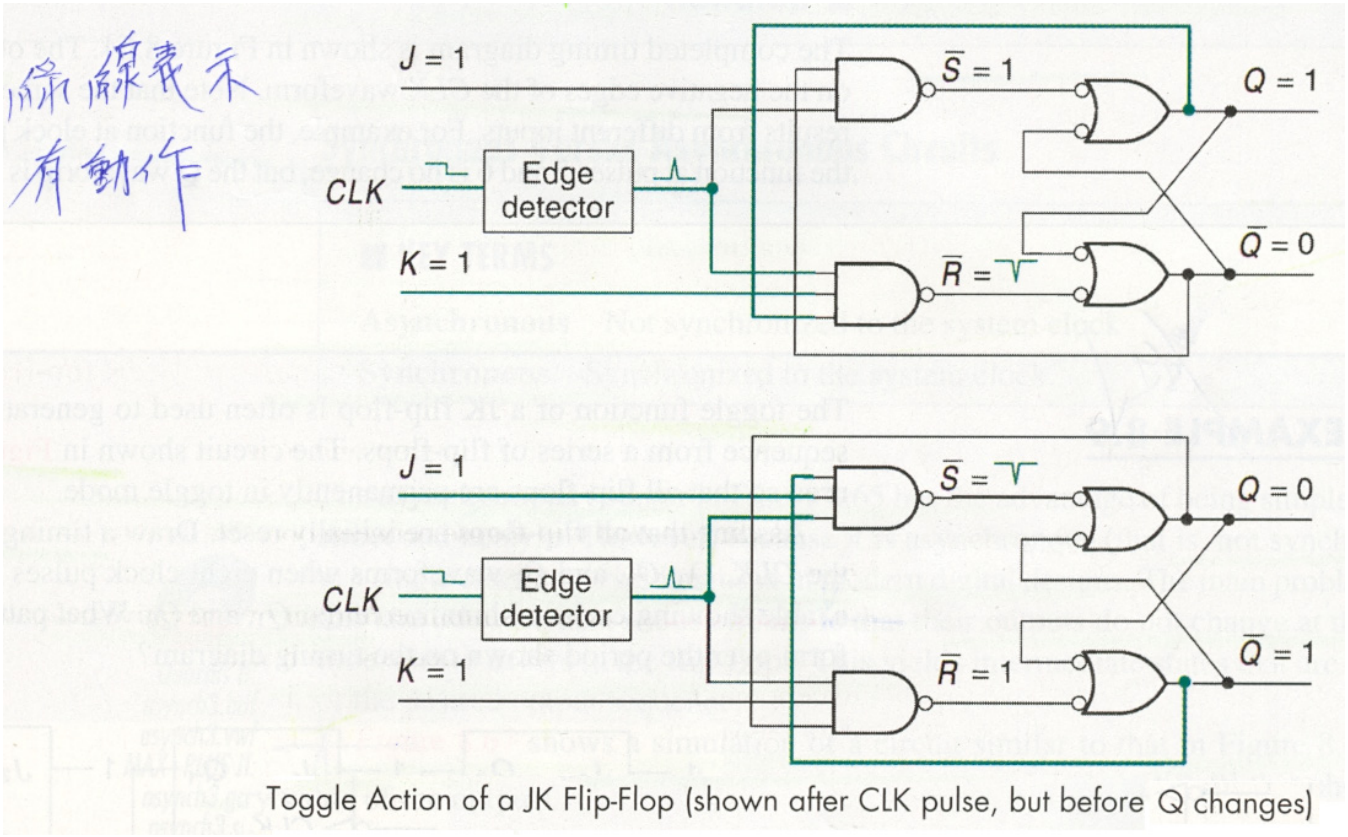
JK Flip-Flop: Feedback



the simplified circuit of a negative-edge triggered JK flip-flop. The circuit is like that of a gated SR latch with an edge detector (an SR flip-flop), except that there are two extra feedback lines from the latch outputs to the steering gate inputs. This extra feedback is responsible for the flip-flop's toggling action.

- With *J* and *K* both HIGH, the flip-flop toggles between opposite logic states with each applied clock pulse.

JK Flip-Flop: Toggle



Toggle Action of a JK Flip-Flop (shown after CLK pulse, but before Q changes)

Figure 8.63 illustrates how the additional two lines cause the flip-flop to toggle. The cross-feedback from Q to K and from $\bar{Q}$ to J enables one, but not both, of the steering gates. The edge detector just after the *CLK* input produces a short positive-going pulse upon detecting a negative edge on the *CLK* waveform. The enabled steering gate complements and transmits this pulse to the latch

Example 8.8

The *J*, *K*, and *CLK* inputs of a negative edge-triggered JK flip-flop are as shown in the timing diagram in Figure 8.64. Complete the timing diagram by drawing the waveforms for Q and $\bar{Q}$. Indicate which function (no change, set, reset, or toggle) is performed at each clock pulse. The flip-flop is initially reset.

Negative Edge-Triggered

CLK	*J*	*K*	Q_{t+1}	$\bar{Q}_{t+1}$	Function
↓	0	0	Q_t	$\bar{Q}_t$	No change
↓	0	1	0	1	Reset
↓	1	0	1	0	Set
↓	1	1	$\bar{Q}_t$	Q_t	Toggle
0	X	X	Q_t	$\bar{Q}_t$	Inhibited
1	X	X	Q_t	$\bar{Q}_t$	Inhibited
↑	X	X	Q_t	$\bar{Q}_t$	Inhibited

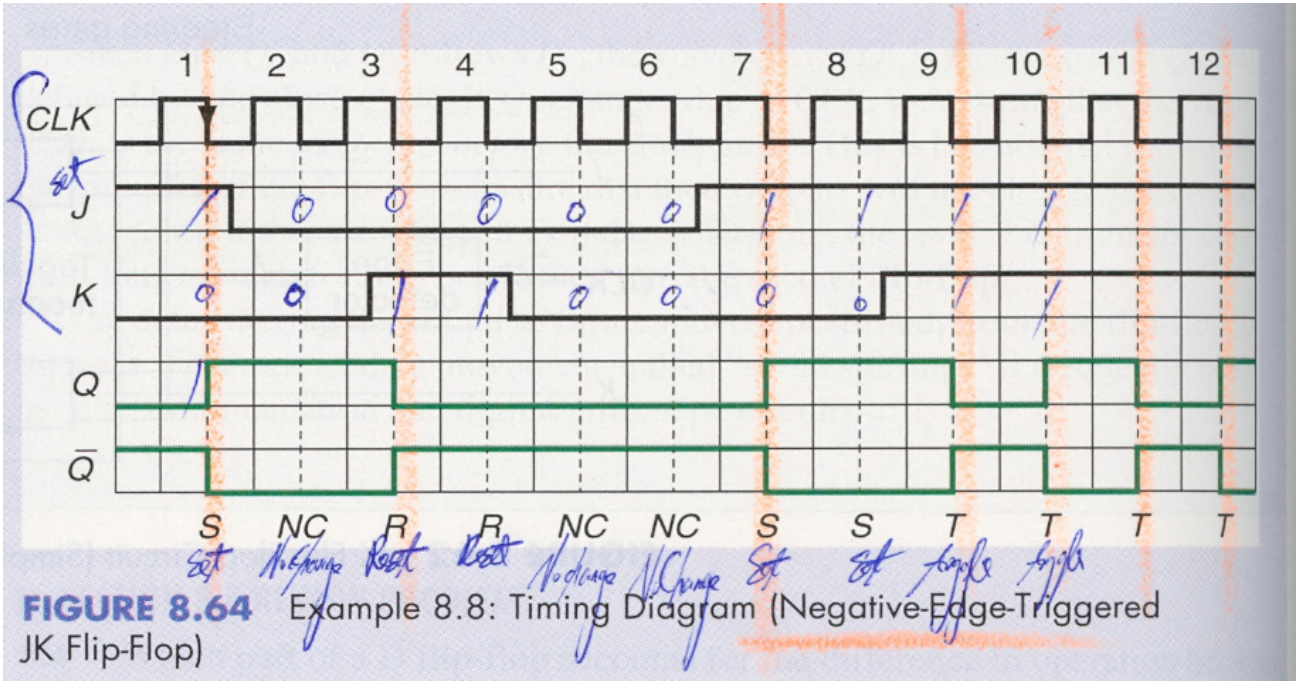


FIGURE 8.64 Example 8.8: Timing Diagram (Negative-Edge-Triggered JK Flip-Flop)

Toggle Applications

- Used to divide an input frequency in half.

- By cascading toggling flip-flops, a counter is created.

Example 8.9 -1

The toggle function of a JK flip-flop is often used to generate a desired output sequence from a series of flip-flops. The circuit shown in Figure 8.65 is configured so that all flip-flops are permanently in toggle mode.

Assume that all flip-flops are initially reset. Draw a timing diagram showing the *CLK*, Q_0, Q_1, and Q_2 waveforms

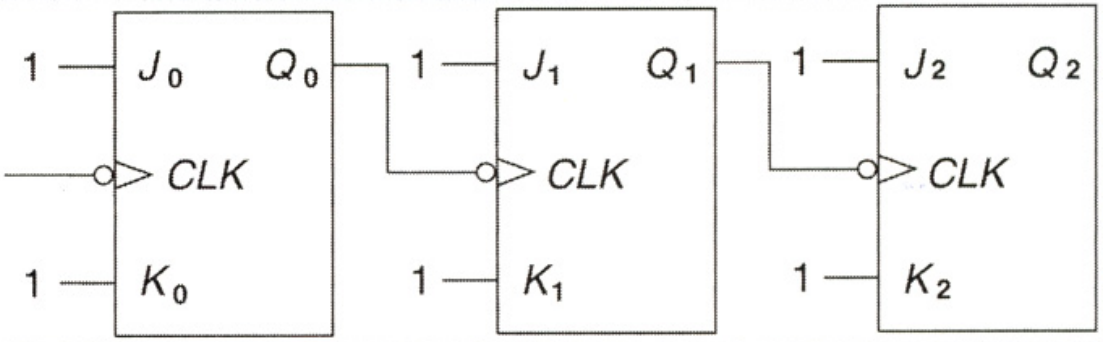
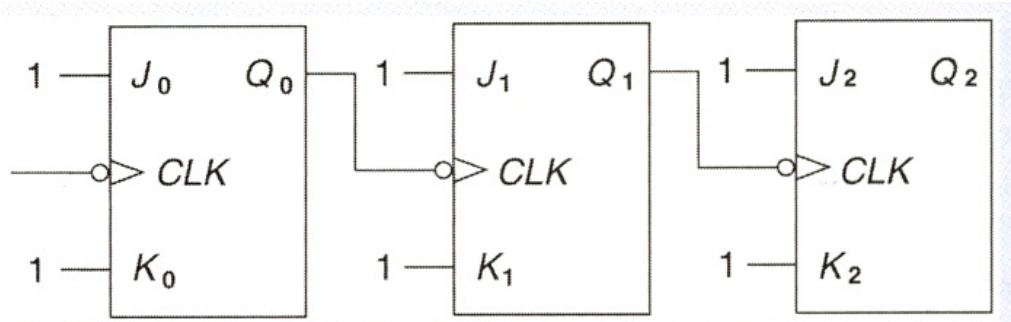


FIGURE 8.65 Example 8.9: Flip-Flop Circuit

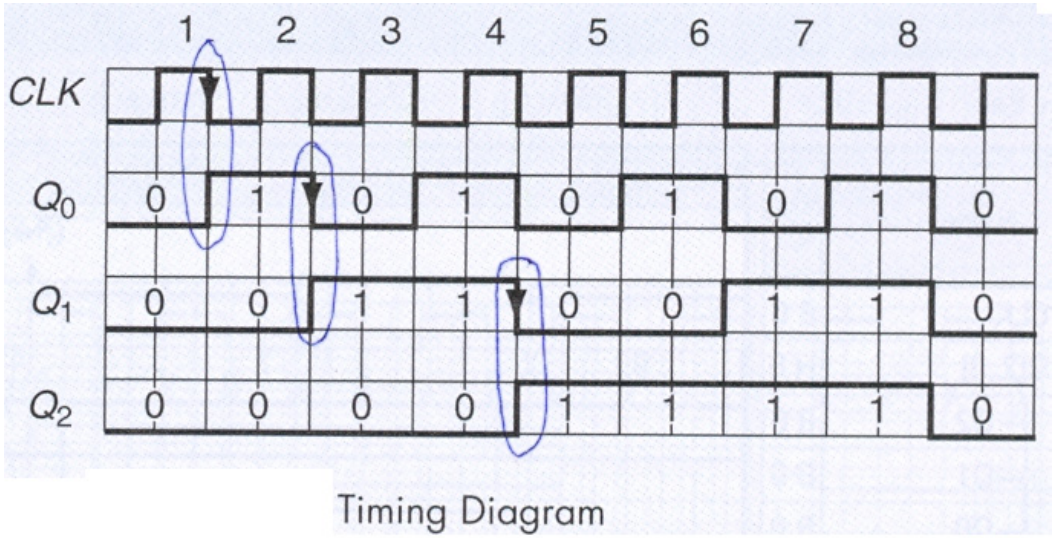
Example 8.9 -2

Sequence of Outputs

Clock Pulse	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0



Flip-Flop Circuit



Timing Diagram

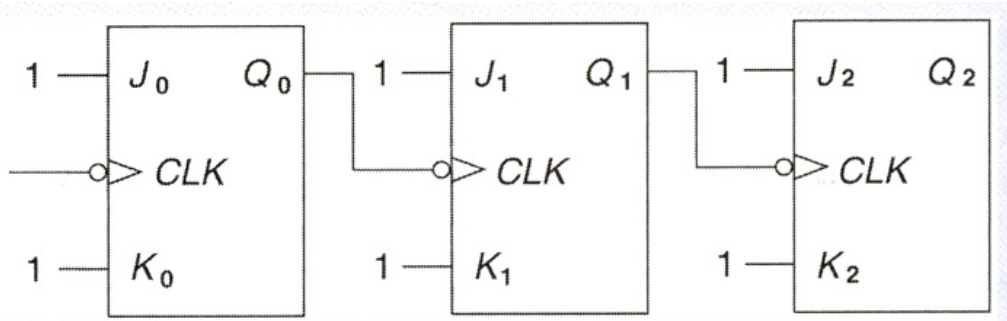
This flip-flop circuit is called a 3-bit asynchronous counter.

Synchronous Versus Asynchronous Circuits

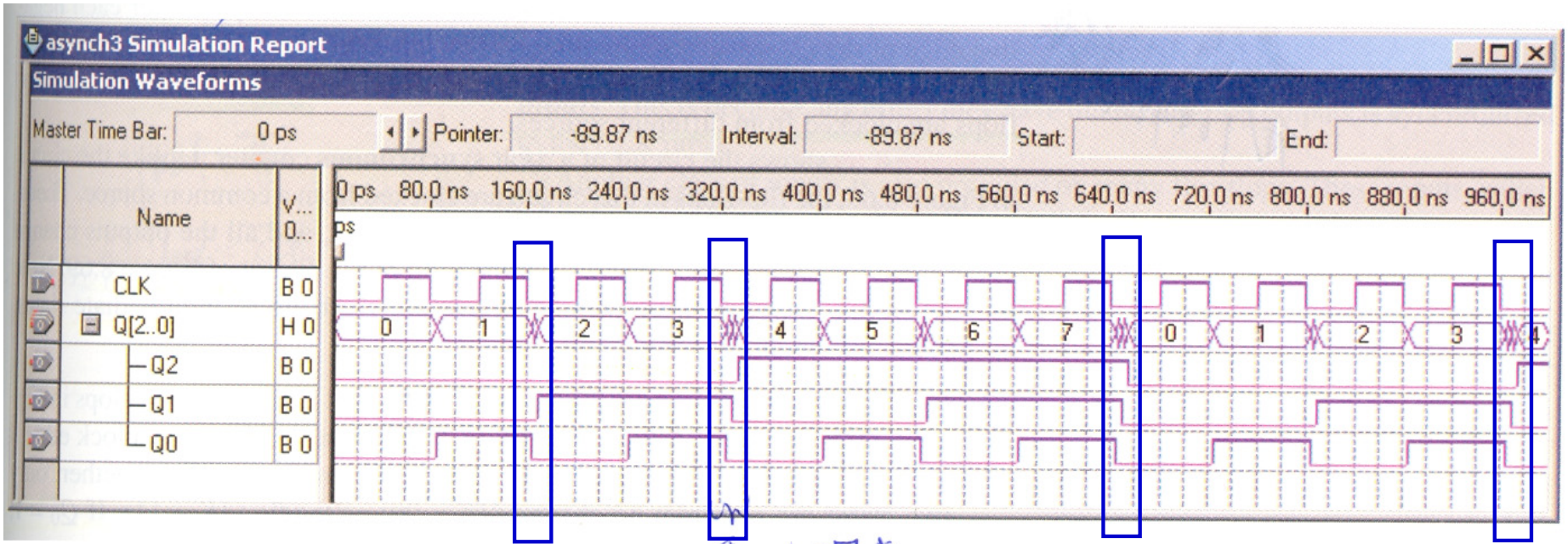
- **Asynchronous** circuits have sequential elements whose outputs change at different times.

- **Synchronous** circuits have sequential elements whose outputs change at the same time.

Asynchronous -1



Flip-Flop Circuit



Simulation for a 3-Bit Asynchronous Counter

Asynchronous -2

- Detail

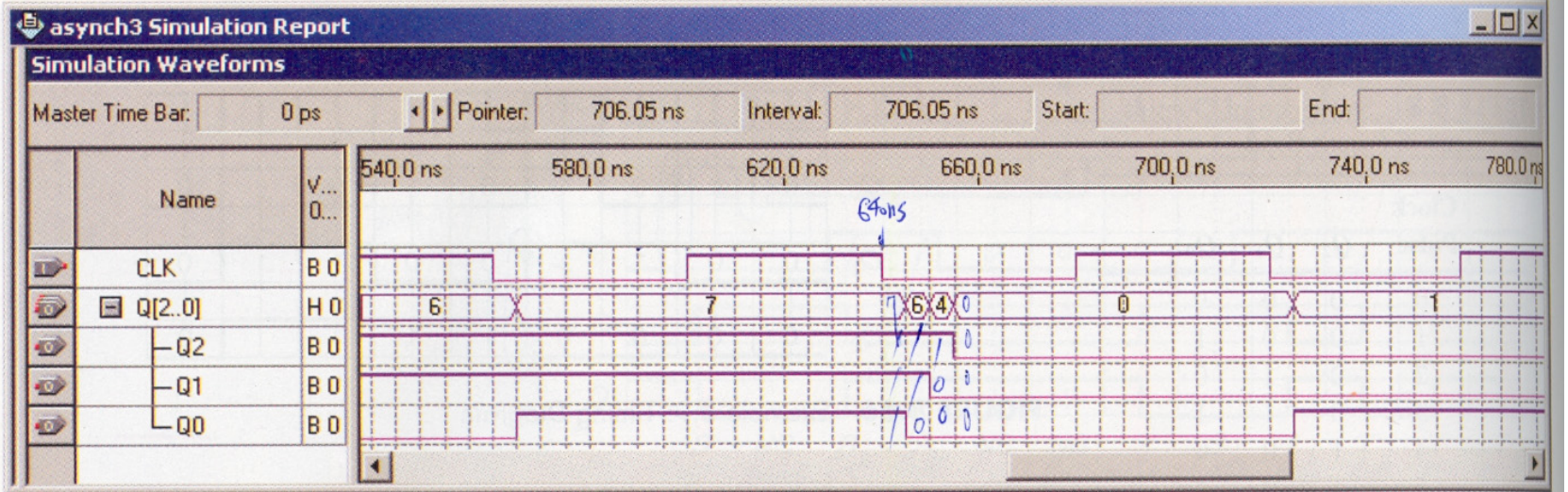


FIGURE 8.68 Detail of Simulation for a 3-Bit Asynchronous Counter

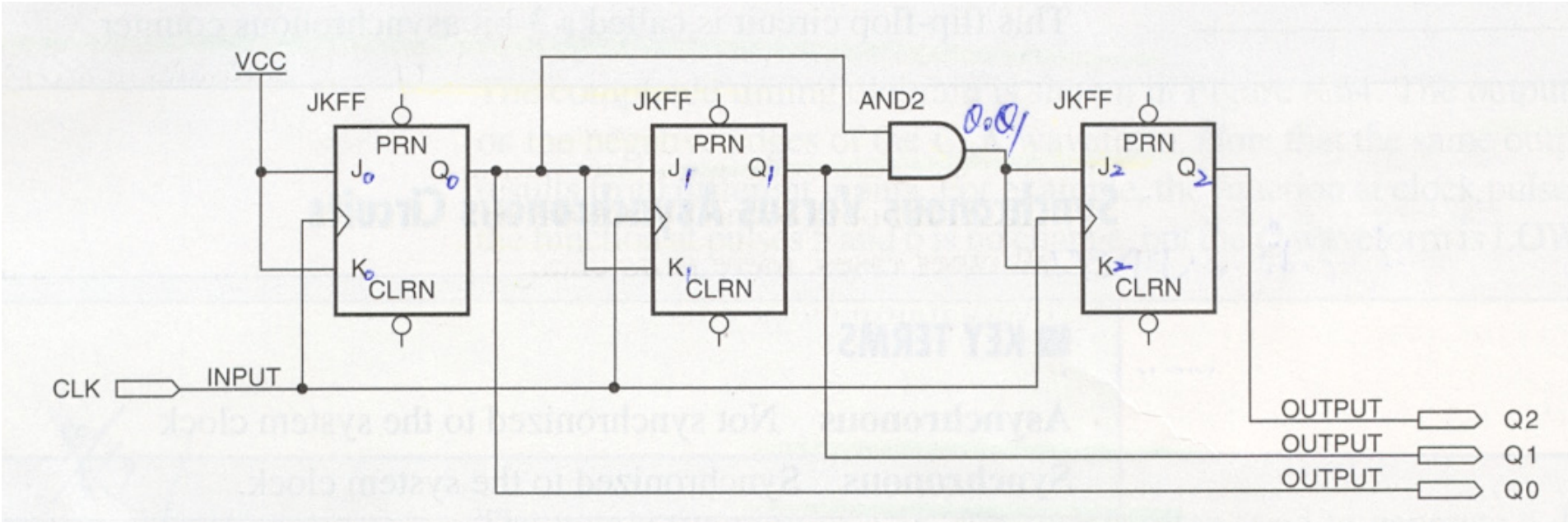
a detail of the simulation at the point where the output goes from 7 to 0 (111 to 000). At 640 ns, the circuit output is 111. A negative clock edge, applied to flip-flop 0, makes Q_0 toggle after a short delay. The output is now 110 ($=6_{10}$). The resulting negative edge on Q_0 clocks flip-flop 1, making it toggle, and yields a new output of 100 ($=4_{10}$). The negative edge on Q_1 clocks flip-flop 2, making the output equal to 000 after a short delay.

Asynchronous -3

- Effects

because it is asynchronous (that is, not synchronized to a single clock), it is seldom used in modern digital designs. The main problem with this and other asynchronous circuits is that their outputs do not change at the same time, due to delays in the flip-flops. This yields intermediate states that are not part of the desired output sequence.

3-Bit Synchronous Counter -1



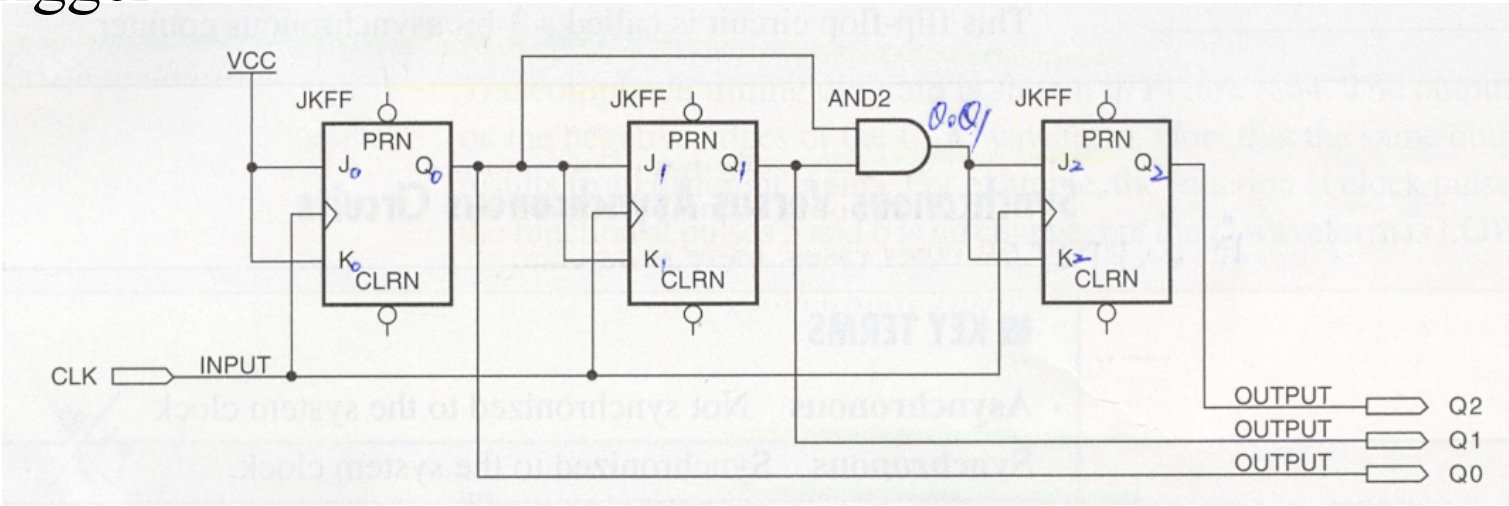
the flip-flops in this circuit are clocked from a common source.

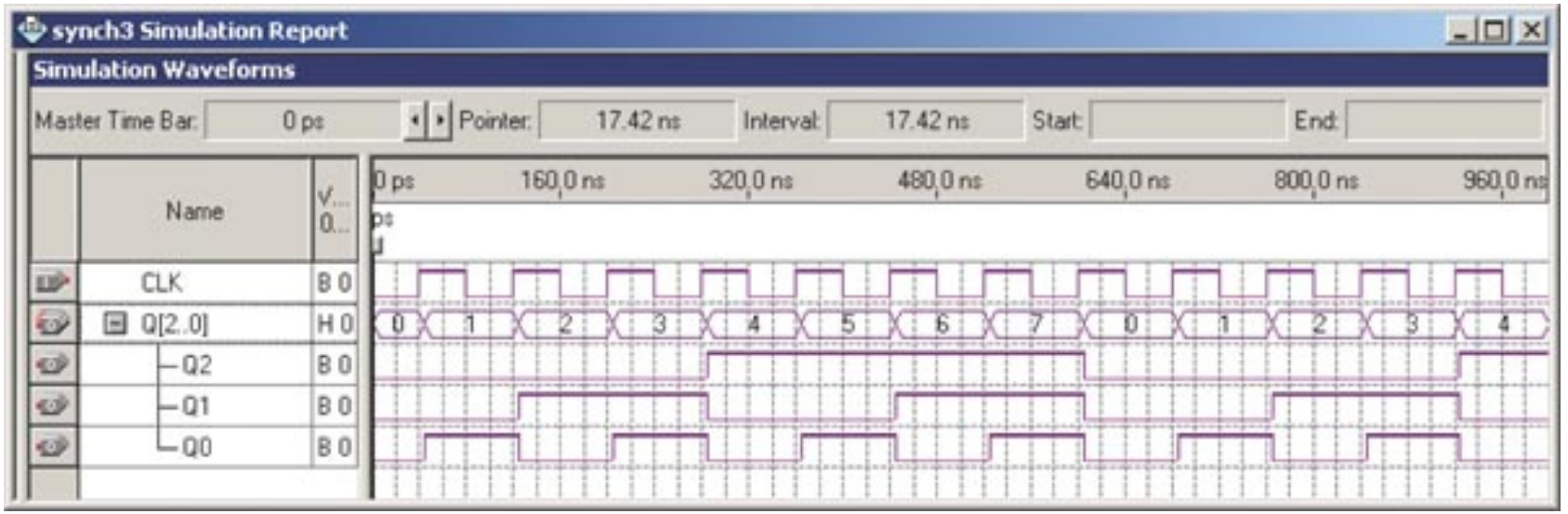
flip-flop delays do not add up through the circuit, and all the outputs change at the same time.

there are no intermediate states.

3-Bit Synchronous Counter -2

- Positive-Edge Trigger





Asynchronous Inputs (Preset and Clear) -1

- Key terms:

Synchronous Inputs The inputs of a flip-flop that do not affect the flip-flop's Q outputs unless a clock pulse is applied. Examples include D, J, and K inputs.

Asynchronous Inputs The inputs of a flip-flop that change the flip-flop's Q outputs immediately, without waiting for a pulse at the CLK input. Examples include preset and clear inputs.

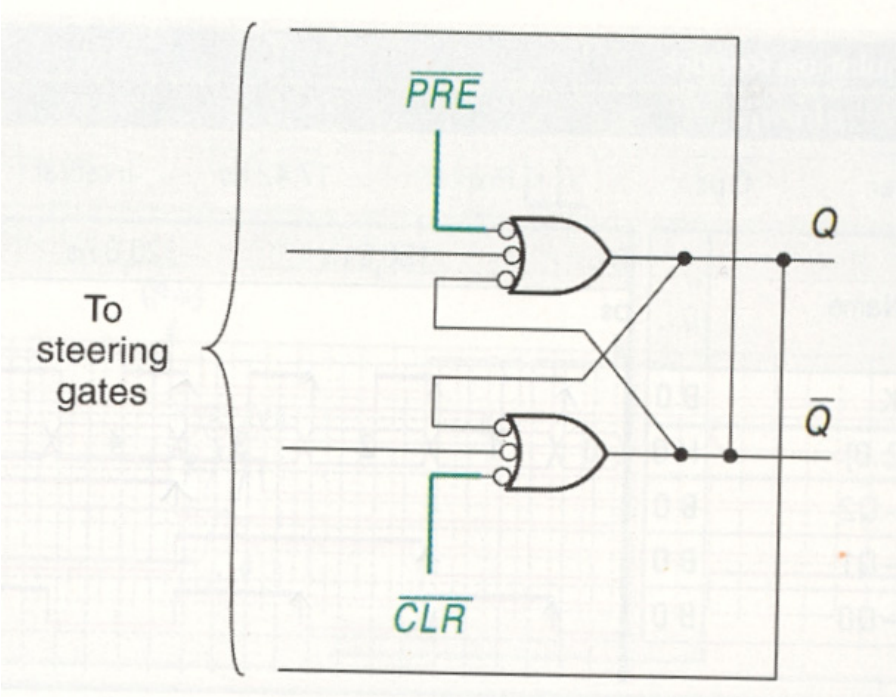
Preset An asynchronous set function.

Clear An asynchronous reset function.

Asynchronous Inputs (Preset and Clear) -2

- Preset:
  - An asynchronous set function, usually designated as $\overline{PRE}$
- Clear:
  - An asynchronous reset function, usually designated as $\overline{CLR}$
- Both Preset and Clear usually have LOW input active levels.

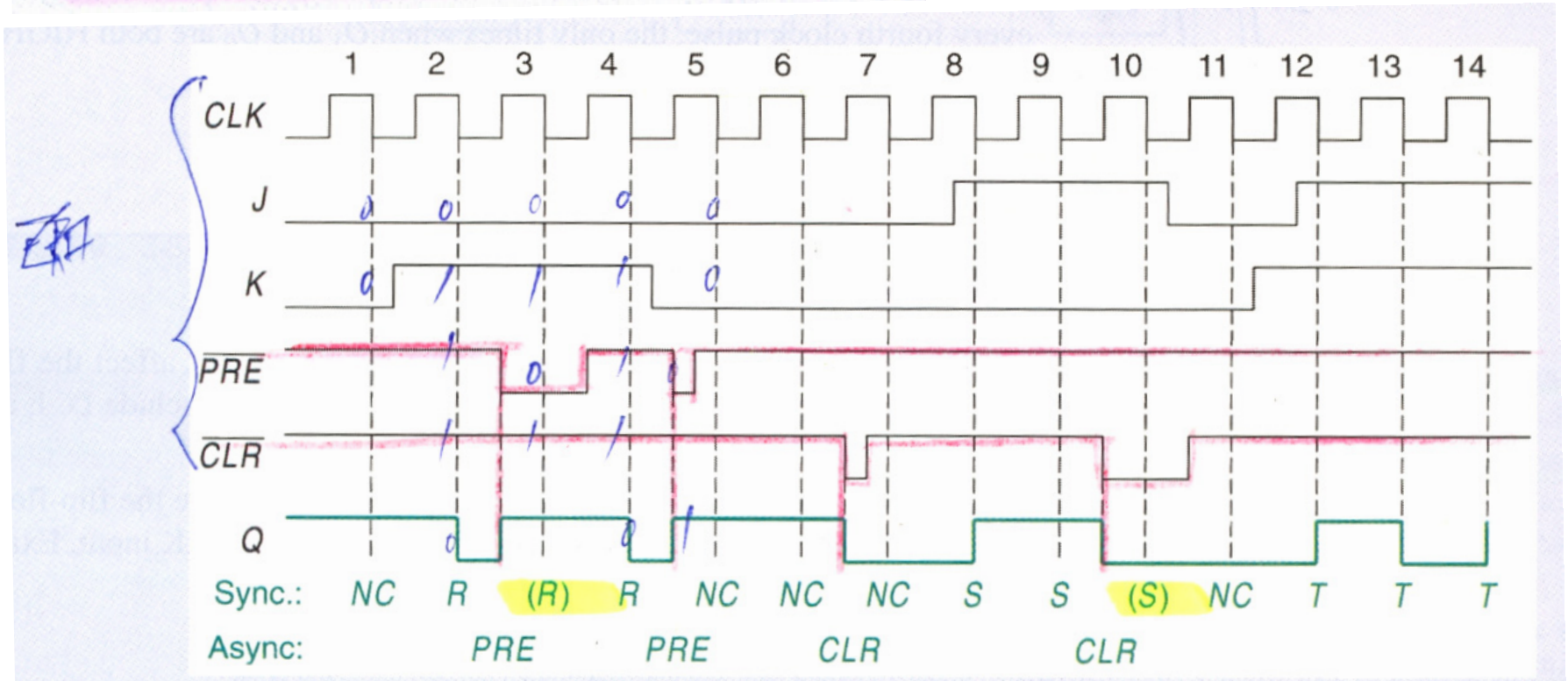
Asynchronous Inputs (Preset and Clear) -3



The $\overline{PRE}$ and $\overline{CLR}$ inputs have direct access to the latch gates of the flip-flop not affected by the *CLK* input. They act exactly the same as the *SET* and *RESET* inputs of an SR latch and will override any synchronous input functions currently active.

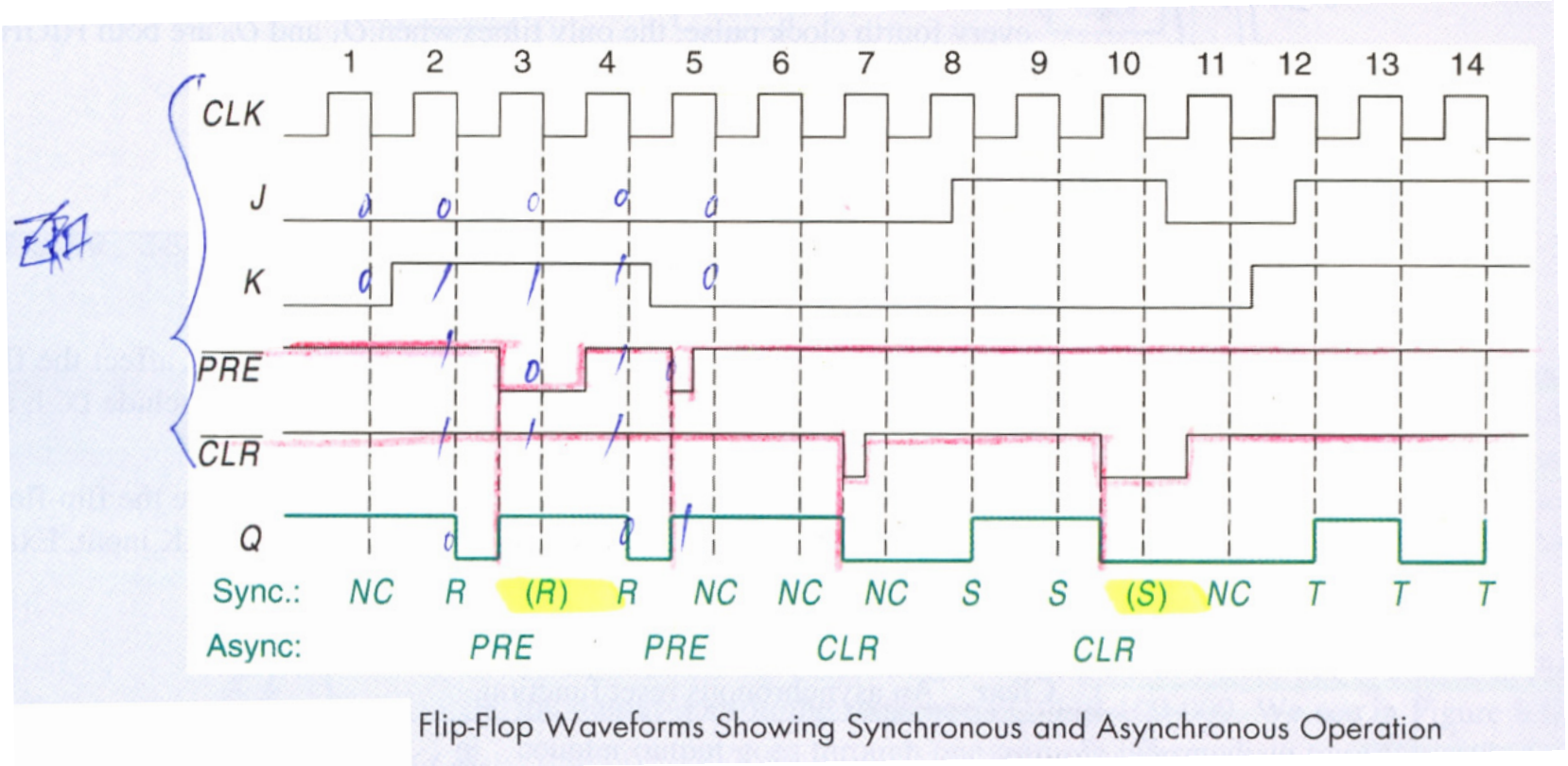
Example 8.10 -1

The waveforms for the *CLK*, *J*, *K*, $\overline{PRE}$ and $\overline{CLR}$ inputs of a negative edge-triggered JK flip-flop are shown in the timing diagram of Figure 8.72. Complete the diagram by drawing the waveform for output *Q*. Assume that *Q* is initially HIGH.



Flip-Flop Waveforms Showing Synchronous and Asynchronous Operation

Example 8.10 -2



Flip-Flop Waveforms Showing Synchronous and Asynchronous Operation

The asynchronous inputs cause an immediate change in Q, whereas the synchronous inputs must wait for the next negative clock edge. If asynchronous and synchronous inputs are simultaneously active, the asynchronous inputs have priority. This occurs in two places: pulse 3 (K, $\overline{PRE}$) and pulse 10 (J, $\overline{CLR}$).

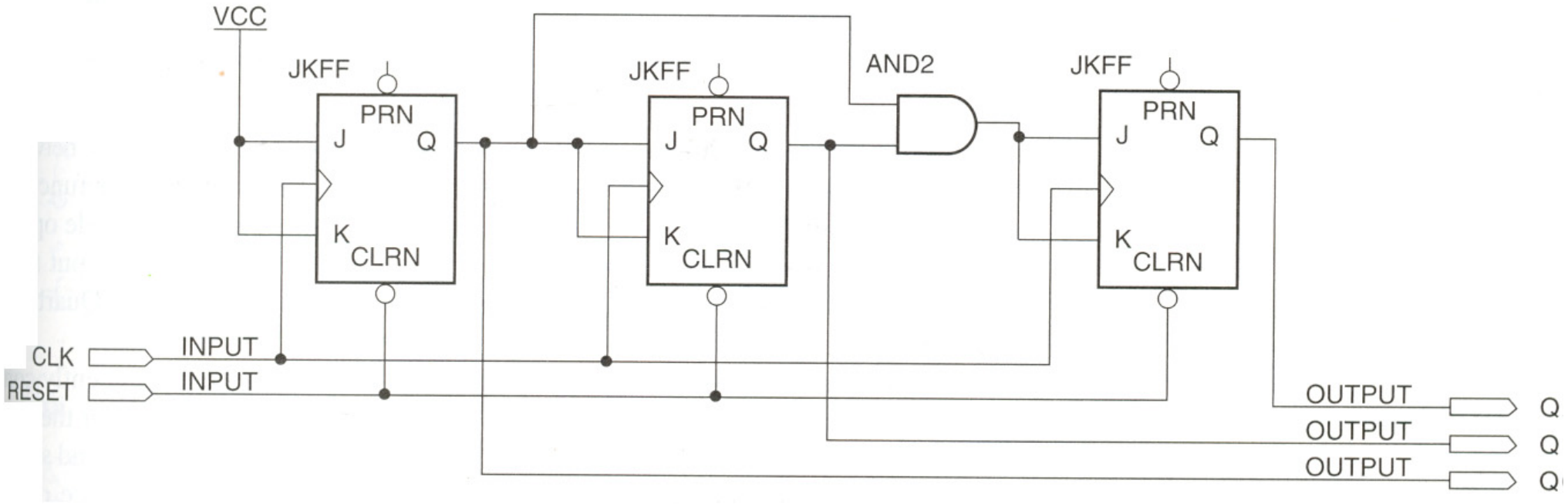
Function Table

TABLE Function Table of a Negative Edge-Triggered JK Flip-Flop with Preset and Clear Functions

	$\overline{PRE}$	$\overline{CLR}$	*CLK*	*J*	*K*	Q_{t+1}	$\bar{Q}_t+1$	**Function**
Synchronous Functions	1	1	↓	0	0	Q_t	$\bar{Q}_t$	No change
	1	1	↓	0	1	0	1	Reset
	1	1	↓	1	0	1	0	Set
	1	1	↓	1	1	$\bar{Q}_t$	Q_t	Toggle
Asynchronous Functions	0	1	X	X	X	1	0	Preset
	1	0	X	X	X	0	1	Clear
	0	0	X	X	X	1	1	Forbidden
	1	1	0	X	X	Q_t	$\bar{Q}_t$	Inhibited
	1	1	1	X	X	Q_t	$\bar{Q}_t$	Inhibited
	1	1	↑	X	X	Q_t	$\bar{Q}_t$	Inhibited

X = Don't care
Q_t = Present state of Q
Q_{t+1} = Next state of Q
↓ = HIGH-to-LOW transition
↑ = LOW-to-HIGH transition

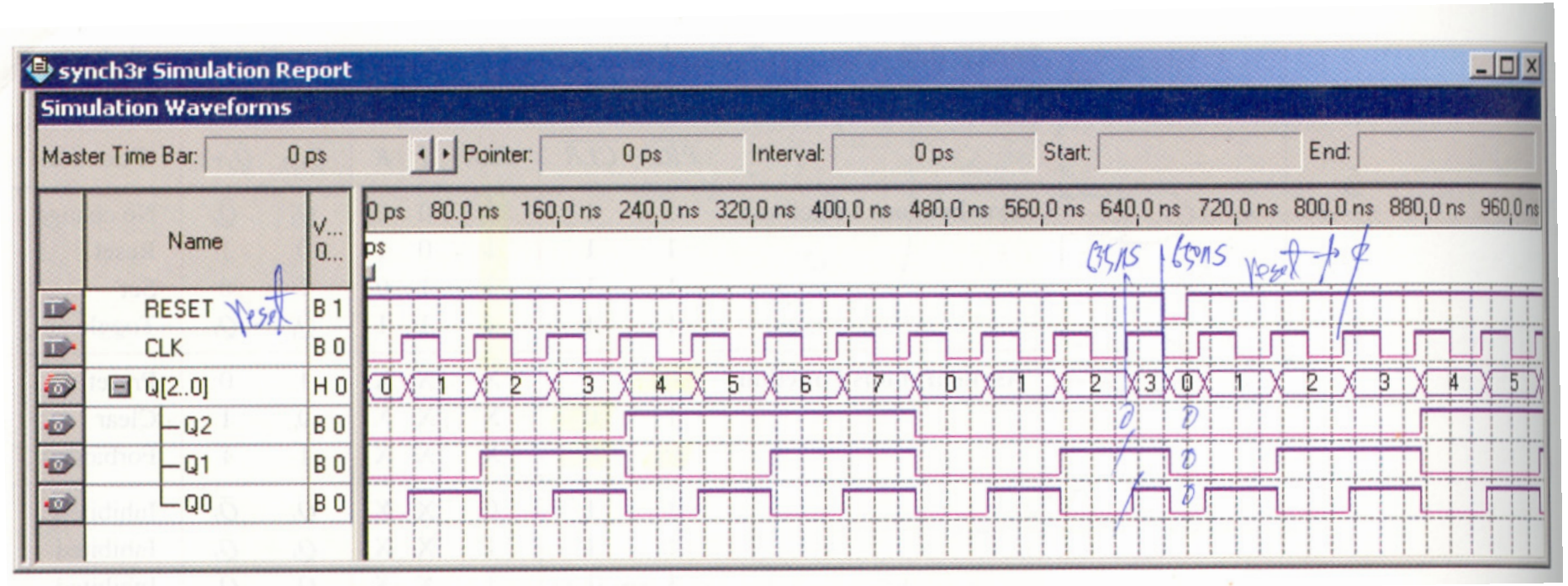
Using Asynchronous Reset in a Synchronous Circuit -1



Synchronous Counter with Asynchronous Reset

An input called *RESET* is tied to the asynchronous $\overline{CLR}$ inputs of all flip-flops. The counter output is set to 000 when the *RESET* line goes LOW.

Using Asynchronous Reset in a Synchronous Circuit -2

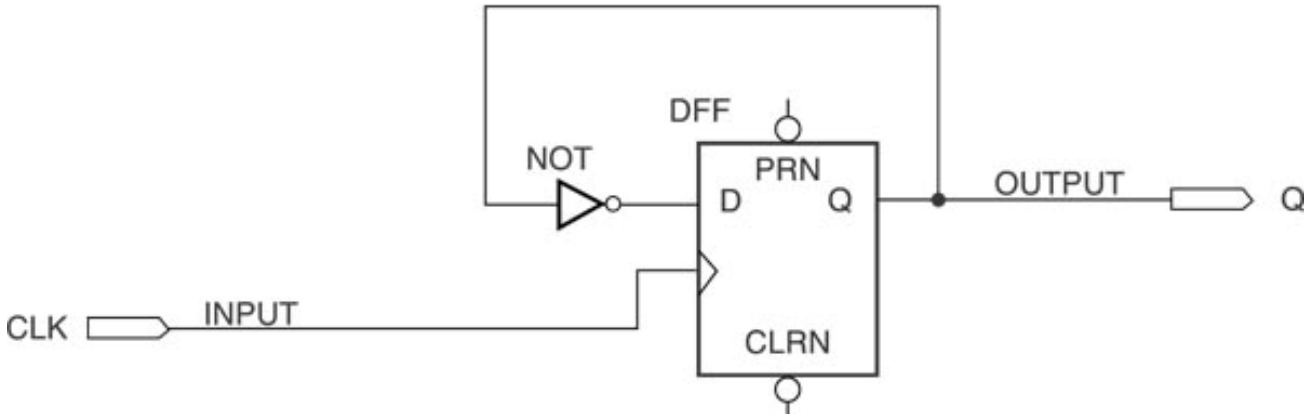


When *RESET* is HIGH, the count proceeds normally. The positive clock edge at 635 ns drives the counter to state 011. The reset pulse at 650 ns sets the counter to 000 as soon as it goes LOW. On the next clock edge, the count proceeds from 000.

The function that sets all flip-flops in a circuit to a known initial state is sometimes called **Master Reset.**

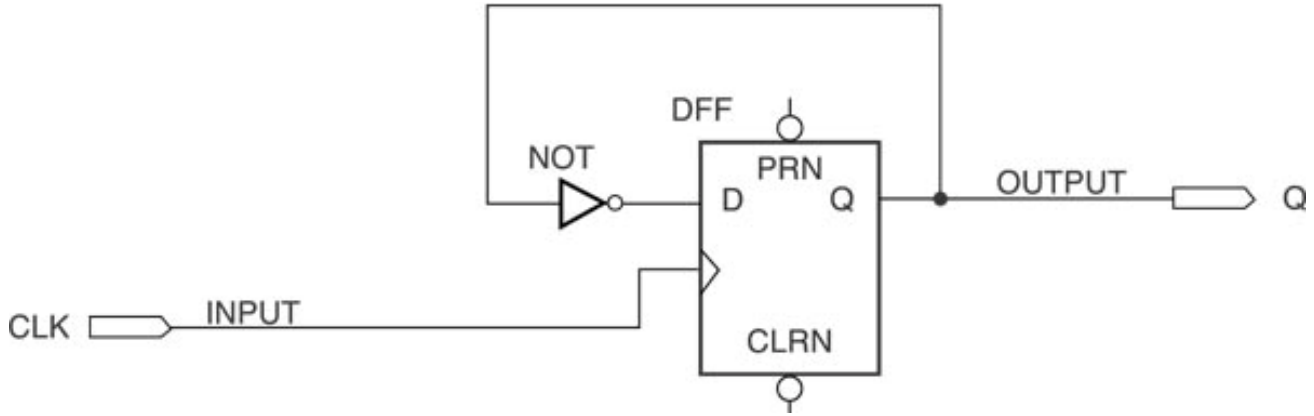
8.6 Edge-Triggered T Flip-Flop

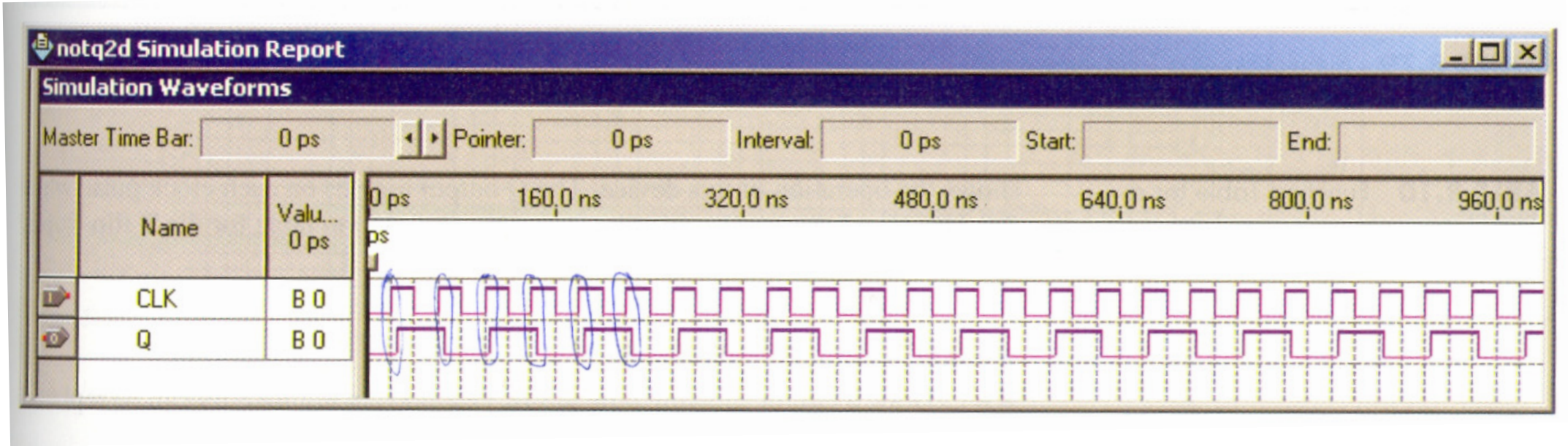
T (Toggle) Flip-Flop A flip-flop whose output toggles between HIGH and LOW states on each applied clock pulse when a synchronous input, called *T*, is active.



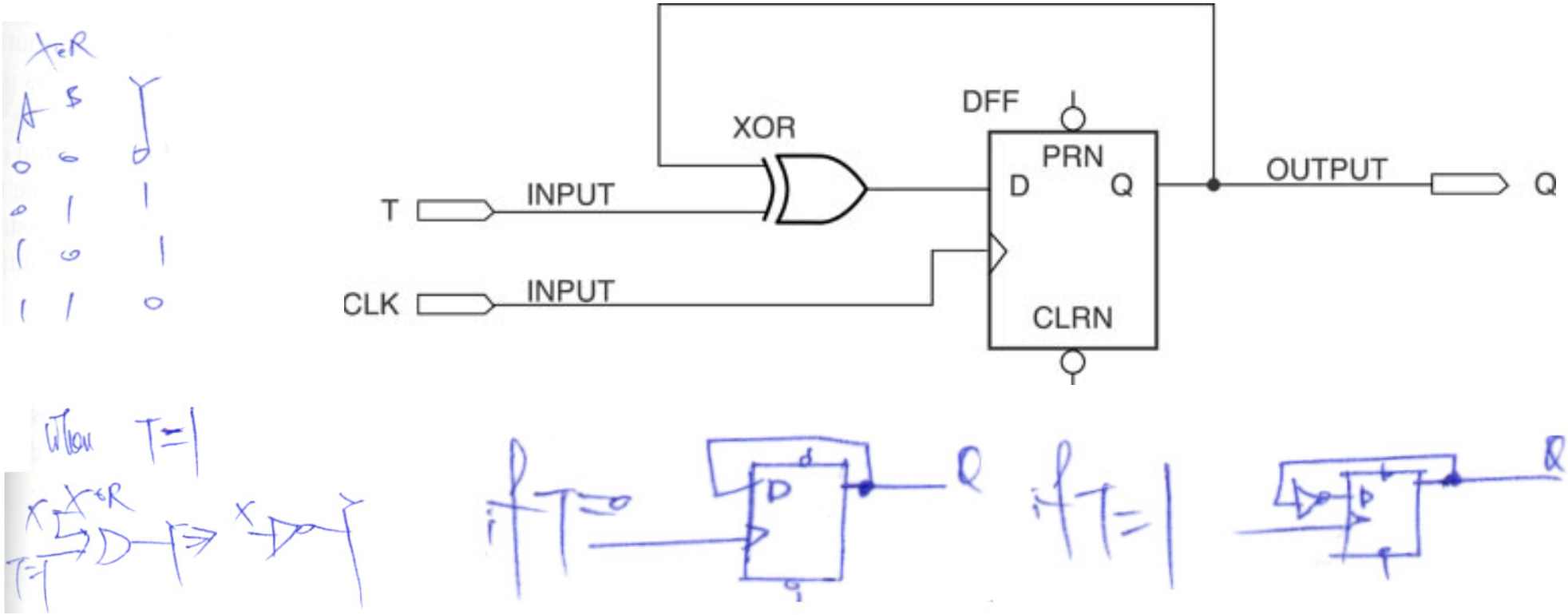
shows a D flip-flop configured for toggle operation. Since Q follows D and $D = \bar{Q}$ in this circuit, then the flip-flop output must change to its opposite state with each clock pulse.

D Flip-Flop's Toggle Function



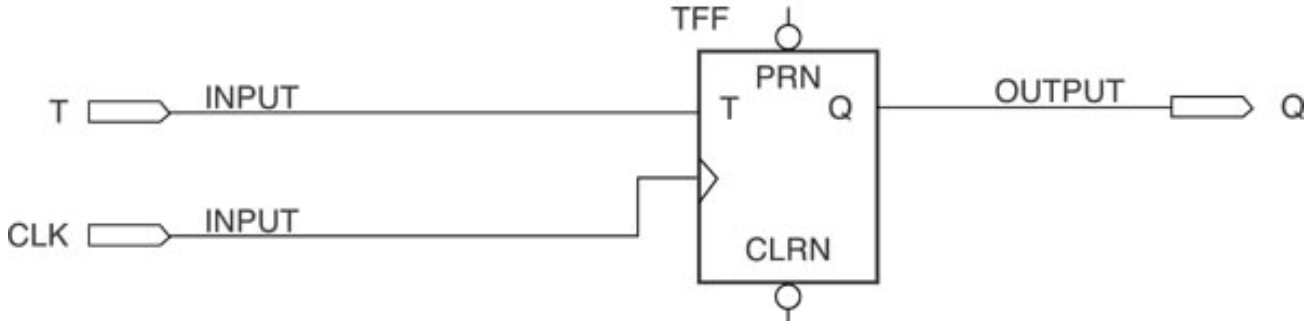


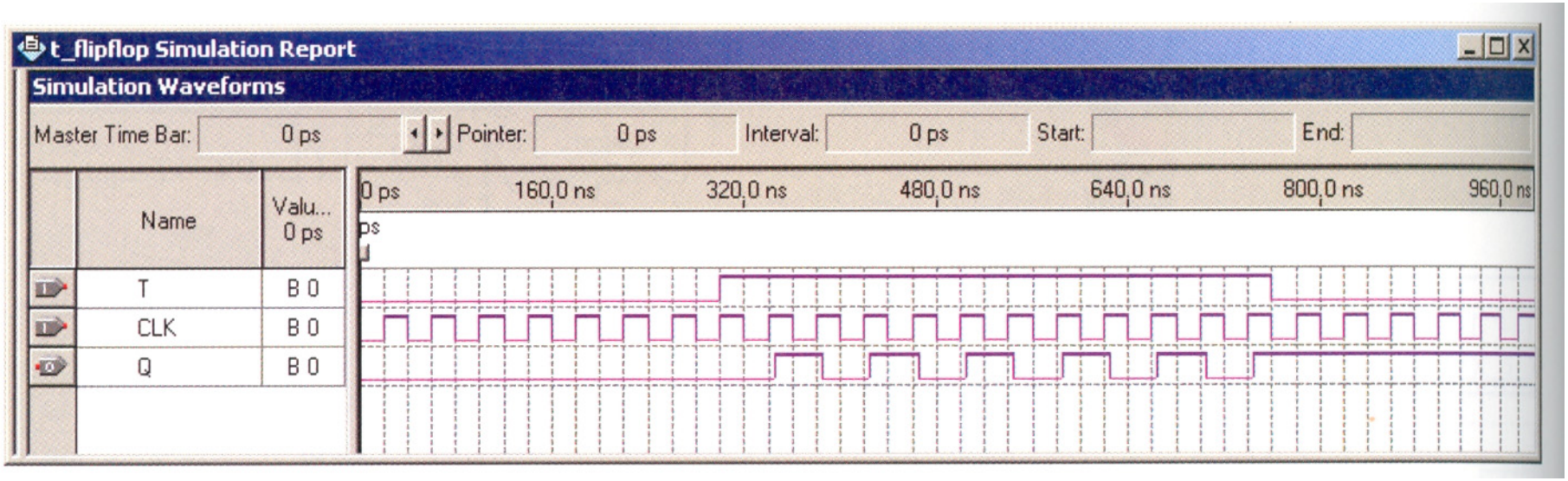
Switchable Toggle Function for a D Flip-Flop



The XOR gate acts as an inverter when the T input is HIGH and as a noninverting buffer when T is LOW. Thus, when T is LOW, the Q output is circulated back to the D input of the flip-flop and the current value of Q is reloaded on the next clock pulse. This is the no change state of the flip-flop. When T is HIGH, the circuit toggles.

T Flip-Flop -1





The Q output toggles on each clock pulse when T is HIGH; otherwise Q retains its last value.

T Flip-Flop -2

Function Table for a T Flip-Flop

CLK	*T*	Q_{t+1}	**Function**
↑	0	$\bar{Q}_t$	No change
↑	1	$\bar{Q}_t$	Toggle
0	X	Q_t	Inhibited
1	X	Q_t	Inhibited
↓	X	Q_t	Inhibited

The Q output toggles on each clock pulse when T is HIGH; otherwise Q retains its last value.

HW