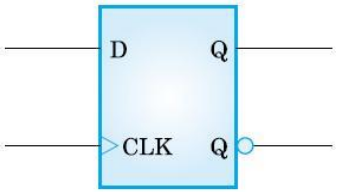
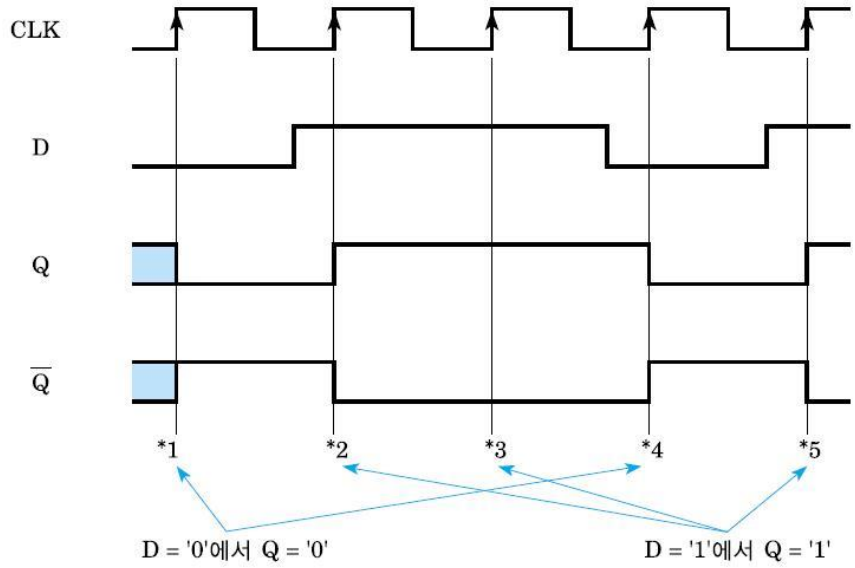
❖ 동작과 회로기호



[그림 7-25] 동기형 D 플립플롭

(1) 상태 천이표

D	Q	Q(t+1)
0	0	0
0	1	0
1	0	1
1	1	1



[그림 7-26] 동기형 D 플립플롭의 타임 차트

❖ 동기형 D 플립플롭의 VHDL 기술

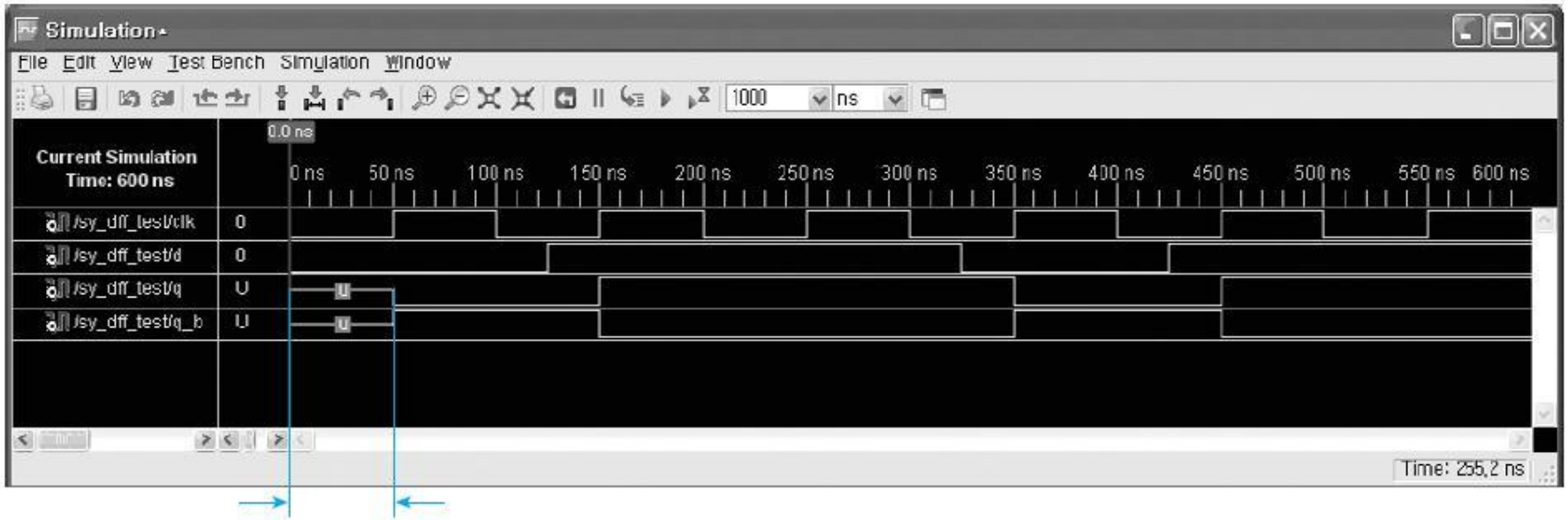
리스트 7-12 동기형 D 플립플롭의 VHDL 기술-동작 레벨(엔티티 「SY_DF

```
library IEEE;
use IEEE.std_logic_1164.all;

--      SY_DFF
entity SY_DFF is
        port ( D, CLK : in  std_logic;
               Q, Q_B : out std_logic );
end SY_DFF;

architecture SY_DFF of SY_DFF is
        signal FF_Q : std_logic;
begin
        Q <= FF_Q; Q_B <= not FF_Q;
        process ( CLK )
        begin
               if ( CLK' event and CLK = '1' ) then
                      FF_Q <= D;
               end if;
        end process;
end SY_DFF;
```

리스트 7-13 동기형 D 플립플롭의 테스트 벤치(엔티티 「SY_DFF_TEST」)

```
library IEEE;
use IEEE.std_logic_1164.all;

entity SY_DFF_TEST is
end SY_DFF_TEST;

architecture SY_DFF_TEST of SY_DFF_TEST is
        component SY_DFF
               port ( D, CLK : in  std_logic;
                      Q, Q_B : out std_logic );
               end component;
        constant STEP : time := 100 ns;
        signal CLK, D : std_logic;
        signal Q, Q_B : std_logic;
begin
        U0:SY_DFF port map ( D => D, CLK => CLK,
                             Q => Q, Q_B => Q_B );
        process
        begin
                                    CLK <= '0';
               wait for STEP / 2; CLK <= '1';
               wait for STEP / 2;
        end process;

        process
        begin
                                       D <= '0';
               wait for STEP * 5 / 4; D <= '1';
               wait for STEP * 2;     D <= '0';
               wait for STEP;         D <= '1';
               wait;
        end process;
end SY_DFF_TEST;
```

❖ 동기형 D 플립플롭의 VHDL 기술

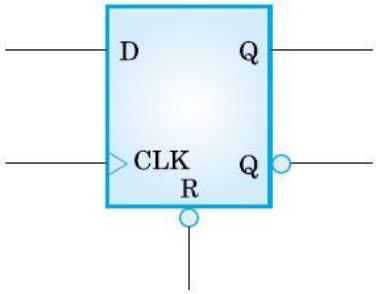


기값이 정해지지 않기 때문에 「Q」, 「Q_B」 = ”

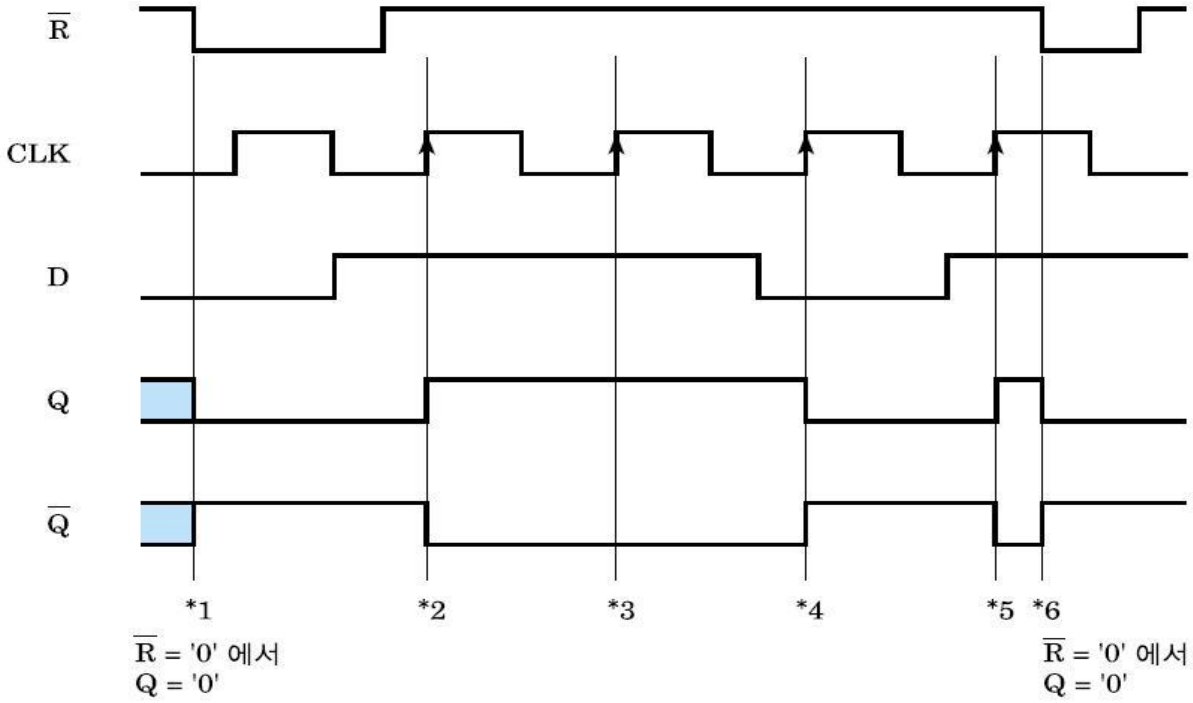
[그림 7-28] 동기형 D 플립플롭의 시뮬레이션 결과

❖ 실제 설계시 자주 사용되는 플립플롭

❖ 동작과 회로기호



[그림 7-37] 비동기 R+동기형 D 플립플롭의 회로기호



[그림 7-38] 비동기 R+동기형 D 플립플롭의 타임 차트

*1, *6 : 「$\overline{R}$」='0'에서 「Q」='0', 「$\overline{Q}$」='1'

*2~*5 : 「$\overline{R}$」='1'에서 동기형 D 플립플롭이 유효하게 되어 「CLK」가 상승할 때 「D」에 대응하여 「Q」, 「$\overline{Q}$」가 출력된다.

*4 : 「D」='0'에서 「Q」='0', 「$\overline{Q}$」='1'

*2,*3,*5 : 「D」='1'에서 「Q」='1', 「$\overline{Q}$」='0'

❖ 상태 천이표

- 표 안의 ' – '은 해당하는 신호 값이 출력에 영향을 주지 않는다는 것을 의미하며, [CLK]의 ' ↑ '는 상승을 의미

[표 7-2] 비동기 R+동기형 D 플립플롭의 상태 천이표

R	CLK	D	Q	Q(t+1)
0	–	–	–	0
1	1	–	1	1
1	1	–	0	0
1	0	–	1	1
1	0	–	0	0
1	↑	1	–	1
1	↑	0	–	0

❖ 비동기 R + 동기형 D 플립플롭의 VHDL 기술

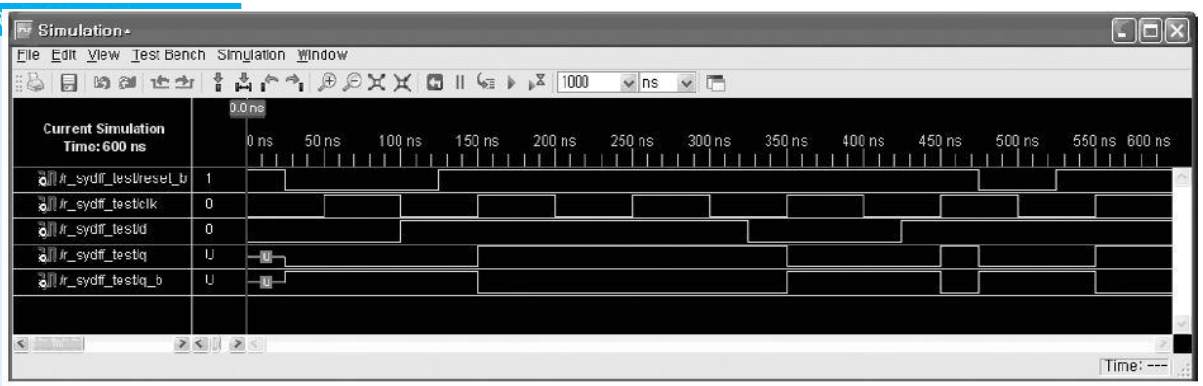
리스트 7-19 비동기 R+동기형 D 플립플롭의 VHDL 기술–동작 레벨(엔티티 「R_SYDFF」)

```
library IEEE;
use IEEE.std_logic_1164.all;

--      R_SYDFF
entity R_SYDFF is
        port ( RESET_B, CLK : in  std_logic;
               D            : in  std_logic;
               Q, Q_B       : out std_logic );
end R_SYDFF;

architecture R_SYDFF of R_SYDFF is
        signal FF_Q : std_logic;
begin
        Q <= FF_Q; Q_B <= not FF_Q;
        process ( RESET_B, CLK ) ←*1
        begin
               if ( RESET_B = '0' ) then ←*2
                      FF_Q <= '0';
               elsif ( CLK' event and CLK = '1' ) then
                      FF_Q <= D;
               end if;
        end process;
end R_SYDFF;
```

▪ 테스트 벤치와 시뮬레이션 결과

리스트 7-20 비동기 R+동기형 D 플립플롭의 테스트 벤치(엔티티 「R

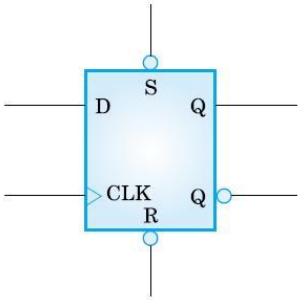
```
library IEEE;
use IEEE.std_logic_1164.all;

entity R_SYDFF_TEST is
end R_SYDFF_TEST;

architecture R_SYDFF_TEST of R_SYDFF_TEST is
       component R_SYDFF
              port ( RESET_B, CLK : in  std_logic;
                     D            : in  std_logic;
                     Q, Q_B       : out std_logic );
              end component;
       constant STEP : time := 100 ns;
       signal RESET_B, CLK, D : std_logic;
       signal Q, Q_B          : std_logic;
begin
       U0:R_SYDFF port map ( RESET_B => RESET_B, D => D,
                             CLK => CLK, Q => Q, Q_B => Q_B );
       process
       begin
                                CLK <= '0';
              wait for STEP / 2; CLK <= '1';
              wait for STEP / 2;
       end process;

       process
       begin
                                    RESET_B <= '1'; D <= '0';
              wait for STEP / 4;    RESET_B <= '0';
              wait for STEP * 3 / 4;                D <= '1';
              wait for STEP / 4;    RESET_B <= '1';
              wait for STEP * 2;                    D <= '0';
              wait for STEP;                        D <= '1';
              wait for STEP / 2;    RESET_B <= '0';
              wait for STEP / 2;    RESET_B <= '1';
              wait;
       end process;
end R_SYDFF_TEST;
```



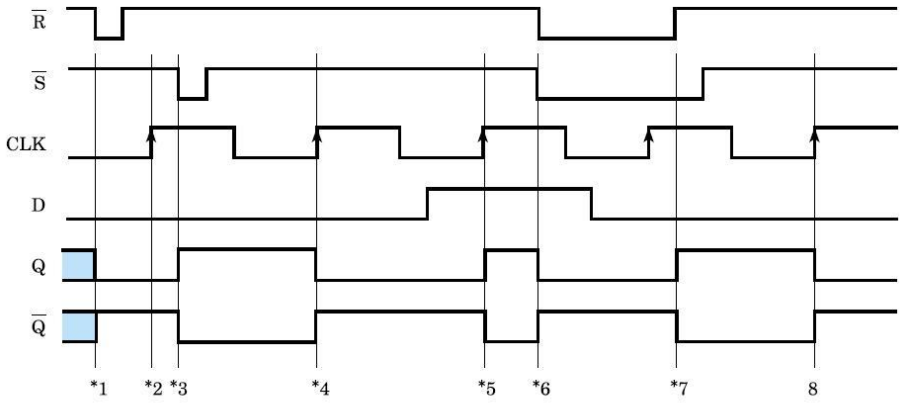
[그림 7-39] 비동기 R+동기형 D 플립플롭의 시뮬레이션 결과

❖ 동작과 회로기호

- 입력은 「$\overline{R}$」, 「$\overline{S}$」, 「D」, 출력은 [Q]와 「$\overline{Q}$」
- [R]은 부논리 입력이므로 '0'으로 플립플롭을 리셋
- [S]는 부논리 입력이므로 '0'으로 플립플롭을 세트
- [R]과 [S]가 동시에 '0'인 경우, 리셋 동작을 먼저 (금지 입력).
- [R]과 [S]가 '1'이라면 [CLK]가 상승하는 시점의 입력 [D]를 [Q]에, 반전한 값을 「$\overline{Q}$」에 출력



[그림 7-40] 비동기 RS + 동기형 D 플립플롭 회로 기호



[그림 7-41] 비동기 RS+동기형 D 플립플롭의 타임 차트

*1 : 「$\overline{R}$」='0', 「$\overline{S}$」='1'에서 「Q」='0', 「$\overline{Q}$」='1'

*3, *7 : 「$\overline{R}$」='1', 「$\overline{S}$」='0'에서 「Q」='1', 「$\overline{Q}$」='0'

*2, *4, *5, *8 : 「$\overline{R}$」='1', 「$\overline{S}$」='1'에서 동기형 D 플립플롭이 유효하게 되어 「CLK」가 상승할 때 「D」에 대응한 「Q」, 「$\overline{Q}$」가 출력된다.

*6 : 「$\overline{R}$」='0', 「$\overline{S}$」='0'에서 금지 입력이 되며 리셋 동작에 의해 「Q」='0', 「$\overline{Q}$」='1'

*2, *4, *8 : 「D」='0'에서 「Q」='0', 「$\overline{Q}$」='1'

*5 : 「D」='1'에서 「Q」='1', 「$\overline{Q}$」='0'

❖ 비동기 RS + 동기형 D 플립플롭의 VHDL 기술

리스트 7-21 비동기 RS+동기형 D 플립플롭의 VHDL 기술-동작 레벨(엔티티 [RS_SYDFF])

```
library IEEE;
use IEEE.std_logic_1164.all;

--      RS_SYDFF
entity RS_SYDFF is
       port ( RESET_B, SET_B : in  std_logic;
              CLK, D         : in  std_logic;
              Q, Q_B         : out std_logic );
end RS_SYDFF;

architecture RS_SYDFF of RS_SYDFF is
       signal REG_Q : std_logic;
begin
       Q <= REG_Q; Q_B <= not REG_Q;
       process ( RESET_B, SET_B, CLK )
       begin
              if ( RESET_B = '0' ) then
                     REG_Q <= '0';
              elsif ( SET_B = '0' ) then
                     REG_Q <= '1';
              elsif ( CLK' event and CLK = '1' ) then
                     REG_Q <= D;
              end if;
       end process;
end RS_SYDFF;
```

❖ 상태 천이표

[표 7-3] 비동기 RS+ 동기형 D 플립플롭의 상태 천이표

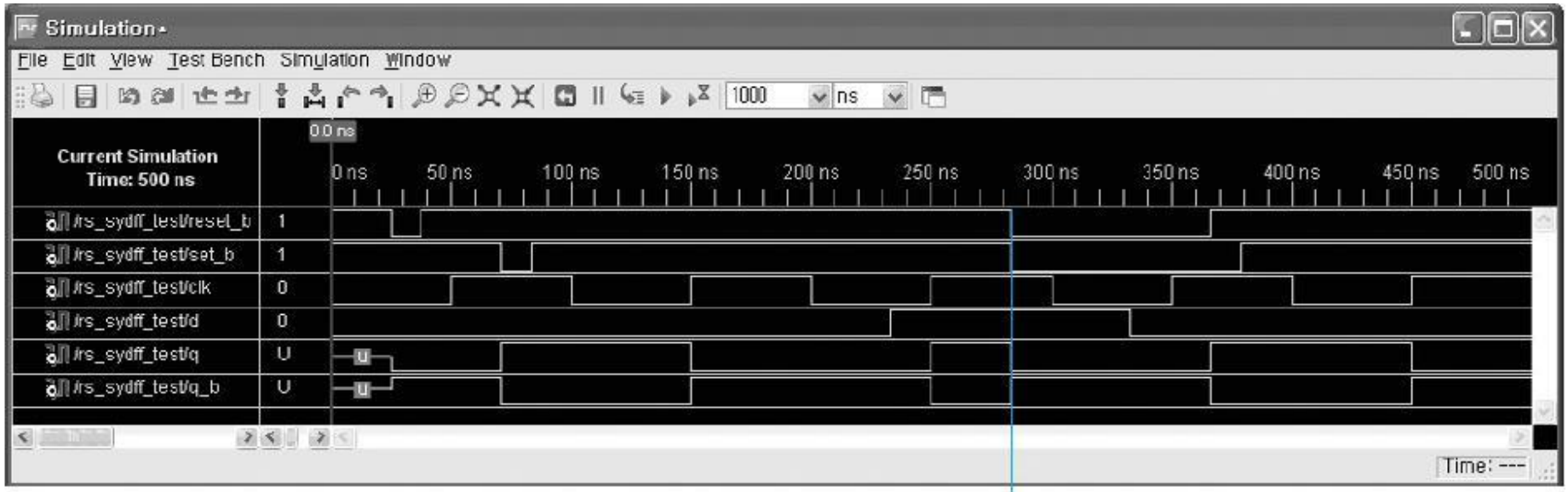
$\overline{R}$	$\overline{S}$	CLK	D	Q	Q(t+1)
0	–	–	–	–	0
0	0	–	–	–	0
1	0	–	–	–	1
1	1	1	–	1	1
1	1	1	–	0	0
1	1	0	–	1	1
1	1	0	–	0	0
1	1	↑	1	–	1
1	1	↑	0	–	0

- 테스트 벤치와 시뮬레이션 결과

리스트 7-22 비동기 RS+동기형 D 플립플롭의 테스트 벤치(엔티티 「RS_SYDFF_TEST」)

```
library IEEE;
use IEEE.std_logic_1164.all;

entity RS_SYDFF_TEST is
end RS_SYDFF_TEST;

architecture RS_SYDFF_TEST of RS_SYDFF_TEST is
       component RS_SYDFF
              port ( RESET_B, SET_B : in  std_logic;
                     CLK, D         : in  std_logic;
                     Q, Q_B         : out std_logic );
              end component;
       constant STEP : time := 100 ns;
       signal RESET_B, SET_B, CLK, D : std_logic;
       signal Q, Q_B                 : std_logic;
begin
       U0:RS_SYDFF port map ( RESET_B => RESET_B, SET_B => SET_B,
                              D => D, CLK => CLK, Q => Q, Q_B => Q_B );
       process
       begin
                                 CLK <= '0';
              wait for STEP / 2; CLK <= '1';
              wait for STEP / 2;
       end process;

       process
       begin
                                      RESET_B <= '1';  SET_B <= '1'; D <= '0';
              wait for STEP / 4;      RESET_B <= '0';
              wait for STEP / 8;      RESET_B <= '1';
              wait for STEP / 3;                       SET_B <= '0';
              wait for STEP / 8;                       SET_B <= '1';
              wait for STEP * 3 / 2;                                  D <= '1';
              wait for STEP / 2;      RESET_B <= '0'; SET_B <= '0';
              wait for STEP / 2;                                      D <= '0';
              wait for STEP / 3;      RESET_B <= '1';
              wait for STEP / 8;                       SET_B <= '1';
              wait;
       end process;
end RS_SYDFF_TEST;
```

- **테스트 벤치와 시뮬레이션 결과**



「RESET_B」와 「SET_B」가 동시에 '0'인 경우는, 리셋 동작이 우선한다.

[그림 7-42] 비동기 RS+동기형 D 플립플롭의 시뮬레이션 결과