CSU22022 Computer Architecture I

Sixth Lecture

Michael Manzke

2023-2024

Trinity College Dublin

Synchronous Sequential Logic

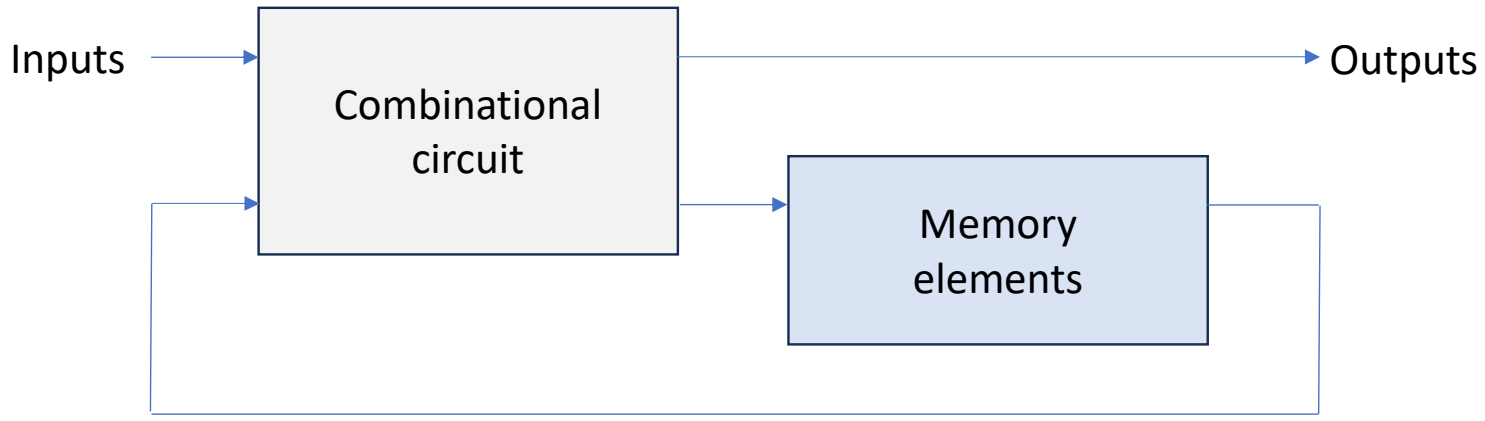

Figure 1: Block Diagram of Synchronous Sequential Logic

SR Latch

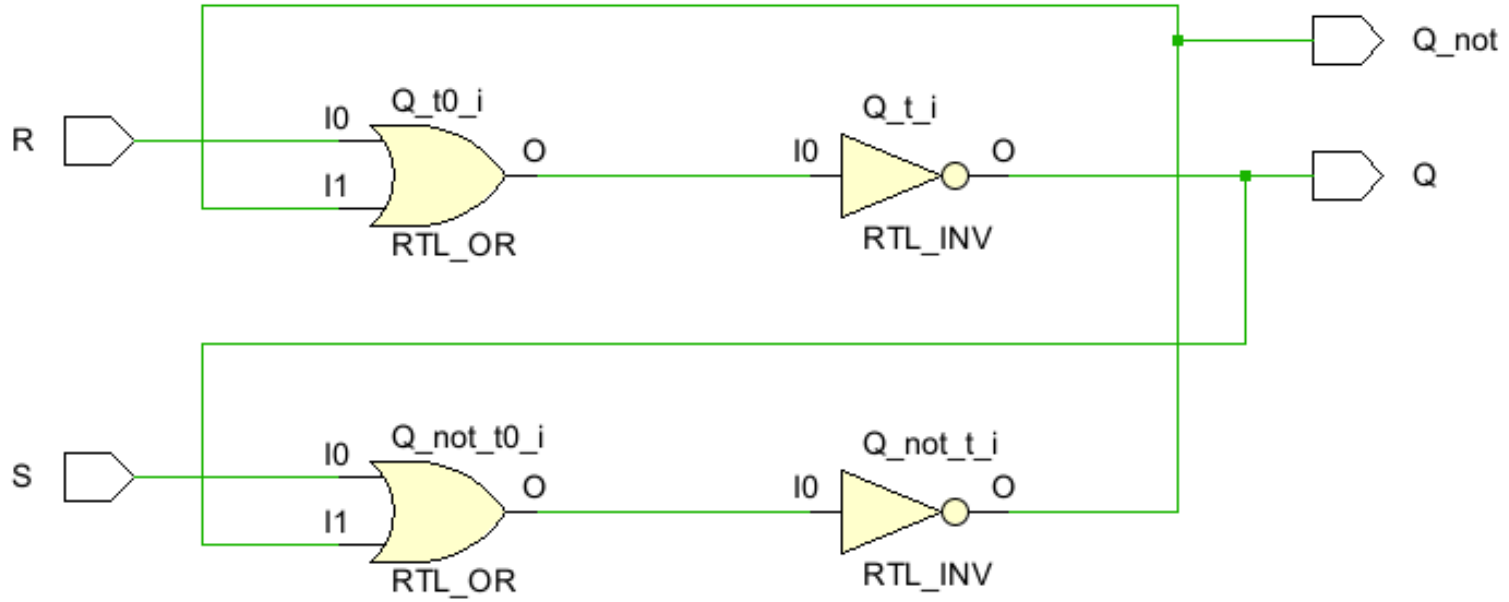

Figure 2: SR Latch Schematic

Latches are useful for asynchronous sequential circuits but not for synchronous sequential circuits.

SR Latch

	S	R	Q	Q'	**Comment**	Case
	0	0	0	0		A
Set	1	0	1	0		B
	0	0	1	0	after S=1, R=0	C
Reset	0	1	0	1		D
	0	0	0	1	after S=0, R=1	E
	1	1	0	0	undefined state	F
	0	0	x	x	undefined state	G

Table 1: SR Latch Function Table

SR Latch

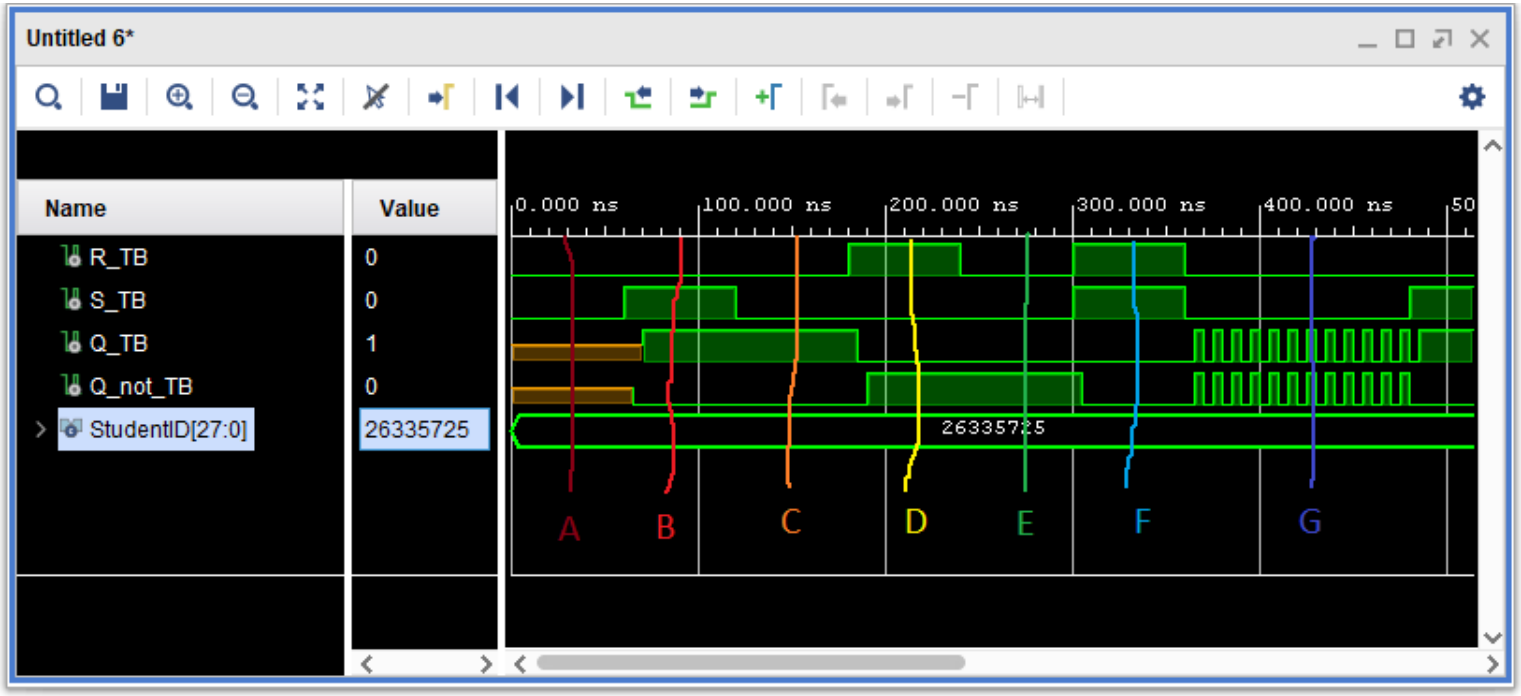

Figure 3: SR Latch Timing Diagram

Cases F and G lead to an undefined state.

SR Latch VHDL Code

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

entity SR_Latch is
    Port ( R, S : in STD_LOGIC;
           Q, Q_not : out STD_LOGIC);
end SR_Latch;

architecture Behavioral of SR_Latch is
    signal Q_t, Q_not_t : std_logic;
begin
    Q_t <= R nor Q_not_t after 5ns;
    Q_not_t <= S nor Q_t after 5ns;
    Q <= Q_t;
    Q_not <= Q_not_t;
end Behavioral;
```

Listing 1: signals Q_t and Q_not_t are needed because you cannot use an out port as input.

SR Latch Test Bench VHDL Code - One

```
1  entity SR_Latch_TB is
2  --  we don't need ports
3  end SR_Latch_TB;
4  architecture Simulation of SR_Latch_TB is
5     -- Component Declaration for the Unit Under Test (UUT)
6      COMPONENT SR_Latch
7      Port ( R, S : in STD_LOGIC;
8             Q, Q_not : out STD_LOGIC);
9      END COMPONENT;
10    --Inputs Signals
11    signal R_TB, S_TB : STD_LOGIC := '0';
12    --Output Signal
13    signal Q_TB, Q_not_TB : STD_LOGIC := '0';
14    -- StudentID e.g. 26 33 57 25(DEC) = 1 91 D9 ED(HEX)
15    constant StudentID : STD_LOGIC_VECTOR (27 downto 0) := x"
       191D9ED";
```

Listing 2: SR Latch entity

SR Latch Test Bench VHDL Code - Two

```
1 begin
2
3    -- Instantiate the Unit Under Test (UUT)
4    uut: SR_Latch PORT MAP (
5           R => R_TB,
6           S => S_TB,
7           Q => Q_TB,
8           Q_not => Q_not_TB
9         );
```

Listing 3: Instantiate the Unit Under Test (UUT)

SR Latch Test Bench VHDL Code - Three

```
1   stim_proc: process
2    begin
3       S_TB <= '0'; R_TB <= '0';-- case A
4       wait for 60 ns;
5       S_TB <= '1'; R_TB <= '0';-- case B
6       wait for 60 ns;
7       S_TB <= '0'; R_TB <= '0';-- case C
8       wait for 60 ns;
9       S_TB <= '0'; R_TB <= '1';-- case D
10      wait for 60 ns;
11      S_TB <= '0'; R_TB <= '0';-- case E
12      wait for 60 ns;
13      S_TB <= '1'; R_TB <= '1';-- case F
14      wait for 60 ns;
15      S_TB <= '0'; R_TB <= '0';-- case G
16      wait for 60 ns;
17   end process;
18 end Simulation;
```

Listing 4: Input signals to the Unit Under Test (UUT)

D Latch

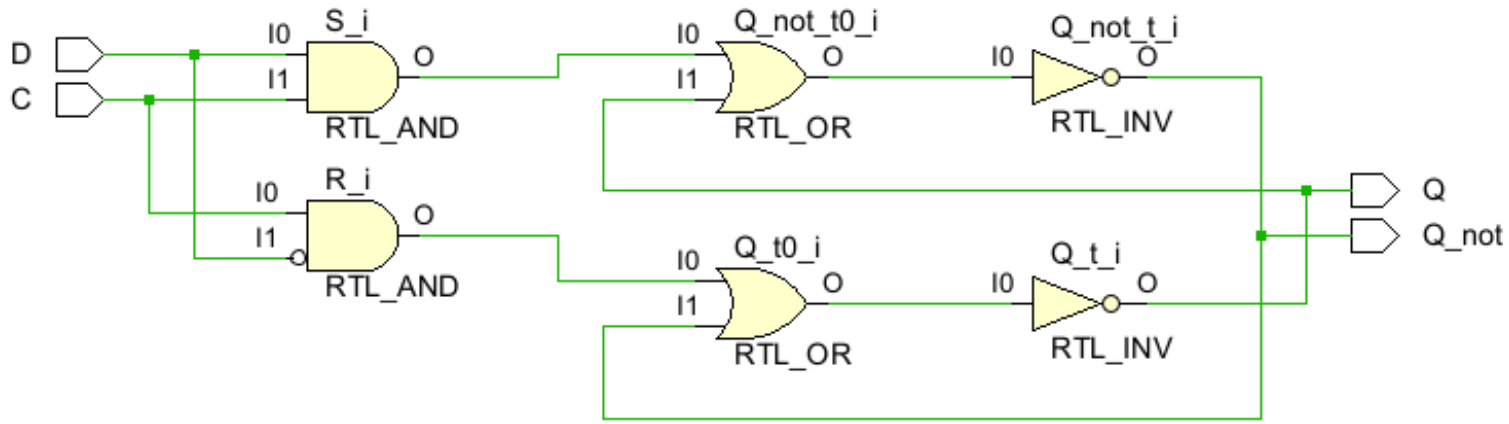

Figure 4: SR Latch Schematic

A **D Latche** can solve the problem with the undefined output.

D Latch

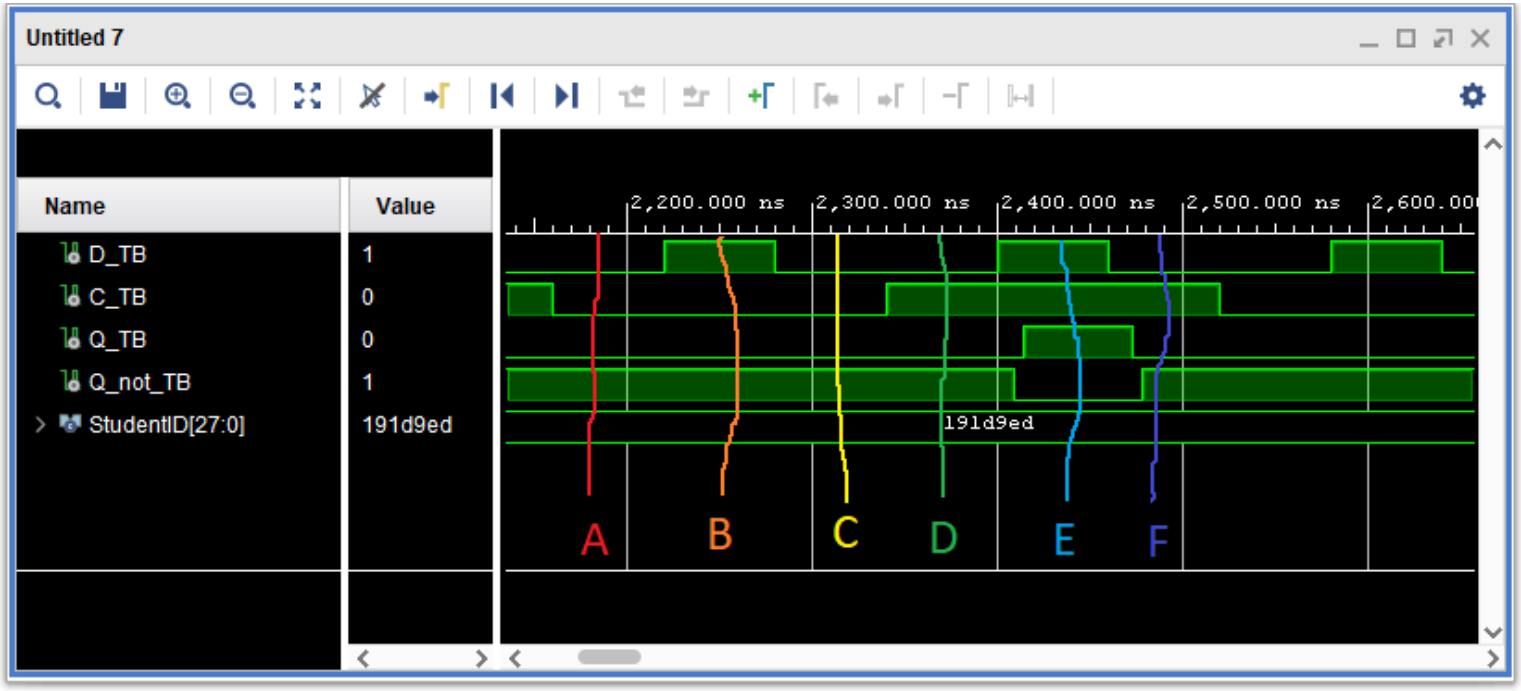

Figure 5: D Latch Timing Diagram

For case definitions see simulation code.

D Latch VHDL Code

```
1  entity D_Latch is
2      Port ( D, C : in STD_LOGIC;
3             Q, Q_not : out STD_LOGIC);
4  end D_Latch;
5
6  architecture Behavioral of D_Latch is
7      signal Q_t, Q_not_t, S, R, D_not : std_logic;
8  begin
9      S <= D and C after 4ns;
10     D_not <= not D after 3ns;
11     R <= C and D_not after 5ns;
12     Q_t <= R nor Q_not_t after 5ns;
13     Q_not_t <= S nor Q_t after 5ns;
14     Q <= Q_t;
15     Q_not <= Q_not_t;
16 end Behavioral;
```

Listing 5: signals Q_t and Q_not_t are needed because you cannot use an out port as input.

D Latch Test Bench VHDL Code - One

```
1  entity D_Latch_TB is
2  --  we don't need ports
3  end D_Latch_TB;
4
5  architecture Simulation of D_Latch_TB is
6     -- Component Declaration for the Unit Under Test (UUT)
7      COMPONENT D_Latch
8      Port ( D, C : in STD_LOGIC;
9             Q, Q_not : out STD_LOGIC);
10     END COMPONENT;
11    --Inputs Signals
12    signal D_TB, C_TB : STD_LOGIC := '0';
13    --Output Signal
14    signal Q_TB, Q_not_TB : STD_LOGIC := '0';
15    -- StudentID e.g. 26 33 57 25(DEC) = 1 91 D9 ED(HEX)
16    constant StudentID : STD_LOGIC_VECTOR (27 downto 0) := x"
       191D9ED";
```

Listing 6: SR Latch entity

D Latch Test Bench VHDL Code - Two

```
1 begin
2     -- Instantiate the Unit Under Test (UUT)
3     uut: D_Latch PORT MAP (
4             D => D_TB,
5             C => C_TB,
6             Q => Q_TB,
7             Q_not => Q_not_TB
8           );
```

Listing 7: Instantiate the Unit Under Test (UUT)

D Latch Test Bench VHDL Code - Three

```
1   stim_proc: process
2    begin
3       C_TB <= '0'; D_TB <= '0';-- case A
4       wait for 60 ns;
5       C_TB <= '0'; D_TB <= '1';-- case B
6       wait for 60 ns;
7       C_TB <= '0'; D_TB <= '0';-- case C
8       wait for 60 ns;
9       C_TB <= '1'; D_TB <= '0';-- case D
10      wait for 60 ns;
11      C_TB <= '1'; D_TB <= '1';-- case E
12      wait for 60 ns;
13      C_TB <= '1'; D_TB <= '0';-- case F
14      wait for 60 ns;
15
16   end process;
17 end Simulation;
```

Listing 8: Input signals to the Unit Under Test (UUT)

D-Type Positive-Edge-Triggered Flip-Flop

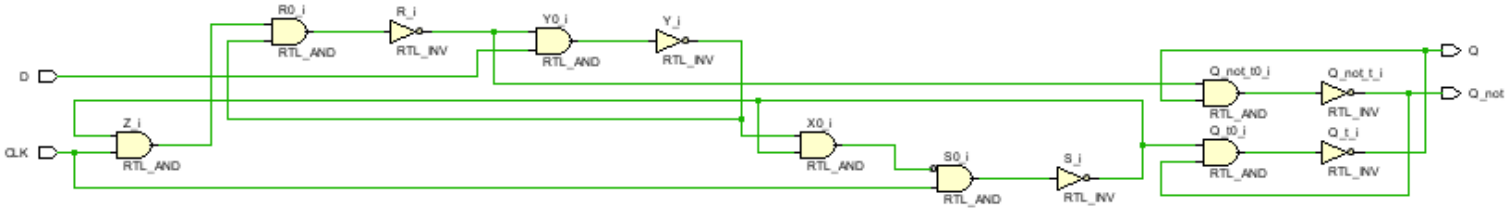

Figure 6: D-Type Positive-Edge-Triggered Flip-Flop Schematic

D-Type Positive-Edge-Triggered Flip-Flop

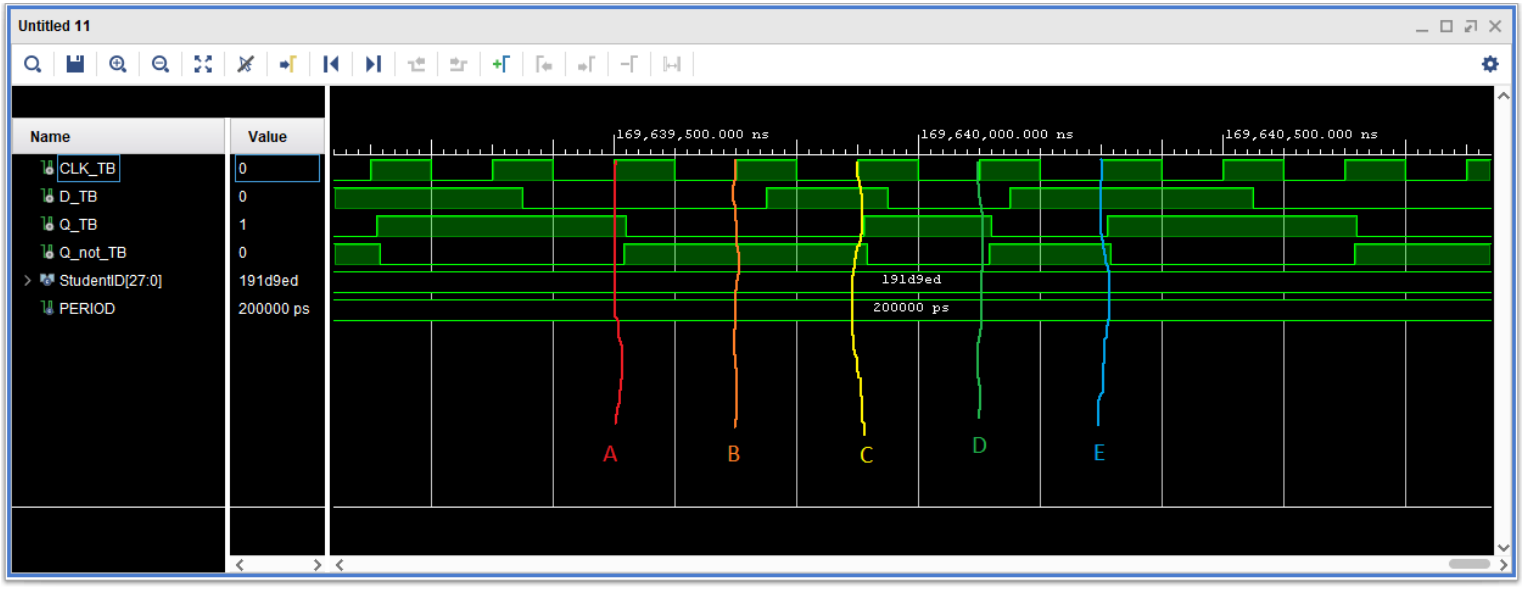

Figure 7: D-Type Positive-Edge-Triggered Flip-Flop Timing Diagram

For case definitions see simulation code.

D-Type Positive-Edge-Triggered Flip-Flop VHDL Code

```
1  entity D_Flip_Flop_PEdge is
2      Port ( CLK, D : in STD_LOGIC;
3             Q, Q_not : out STD_LOGIC);
4  end D_Flip_Flop_PEdge;
5  architecture Behavioral of D_Flip_Flop_PEdge is
6     signal Q_t, Q_not_t, X, S, R, Y, Z : std_logic;
7  begin
8      X <= Y nand S after 5ns;
9      S <= X nand CLK after 5ns;
10     Z <= S and CLK after 5ns;
11     R <= Z nand Y after 5ns;
12     Y <= R nand D after 5ns;
13     Q_t <= S nand Q_not_t after 5ns;
14     Q_not_t <= R nand Q_t after 5ns;
15     Q <= Q_t;
16     Q_not <= Q_not_t;
17 end Behavioral;
```

Listing 9: signals Q_t and Q_not_t are needed because you cannot use an out port as input.

D-Type Positive-Edge-Triggered Flip-Flop Test Bench VHDL Code - One

```
1  entity D_Flip_Flop_PEdge_TB is
2  --  we don't need ports
3  end D_Flip_Flop_PEdge_TB;
4  architecture Simulation of D_Flip_Flop_PEdge_TB is
5     -- Component Declaration for the Unit Under Test (UUT)
6      COMPONENT D_Flip_Flop_PEdge
7      Port ( CLK, D : in STD_LOGIC;
8             Q, Q_not : out STD_LOGIC);
9      END COMPONENT;
10    --Inputs Signals
11    signal CLK_TB, D_TB : STD_LOGIC := '0';
12    --Output Signal
13    signal Q_TB, Q_not_TB : STD_LOGIC := '0';
14    -- StudentID e.g. 26 33 57 25(DEC) = 1 91 D9 ED(HEX)
15    constant StudentID : STD_LOGIC_VECTOR (27 downto 0) := x"
       191D9ED";
16    constant PERIOD : time := 200ns;
```

Listing 10: D-Type Positive-Edge-Triggered Flip-Flop entity

D-Type Positive-Edge-Triggered Flip-Flop Test Bench VHDL Code - Two

```
1  begin
2
3     -- Instantiate the Unit Under Test (UUT)
4     uut: D_Flip_Flop_PEdge PORT MAP (
5             CLK => CLK_TB,
6             D => D_TB,
7             Q => Q_TB,
8             Q_not => Q_not_TB
9           );
10
11    Clk_TB <= not CLK_TB after PERIOD/2;
```

Listing 11: Instantiate the Unit Under Test (UUT)

D-Type Positive-Edge-Triggered Flip-Flop Test Bench VHDL Code - Three

```
1    stim_proc: process
2    begin
3       wait until CLK_TB'event and CLK_TB='1';
4       D_TB <= '0' after PERIOD/4;                    -- Case A
5       wait until CLK_TB'event and CLK_TB='1';
6       D_TB <= '0' after PERIOD/4;                    -- Case B
7       wait until CLK_TB'event and CLK_TB='1';
8       D_TB <= '1' after PERIOD/4;                    -- Case C
9       wait until CLK_TB'event and CLK_TB='1';
10      D_TB <= '0' after PERIOD/4;                    -- Case D
11      wait until CLK_TB'event and CLK_TB='1';
12      D_TB <= '1' after PERIOD/4;                    -- Case E
13      wait until CLK_TB'event and CLK_TB='1';
14   end process;
15 end Simulation;
```

Listing 12: Input signals to the Unit Under Test (UUT)