CSU22022 Computer Architecture I

Tenth Lecture - Register Transfer

Michael Manzke

2023-2024

Trinity College Dublin

Register Transfer

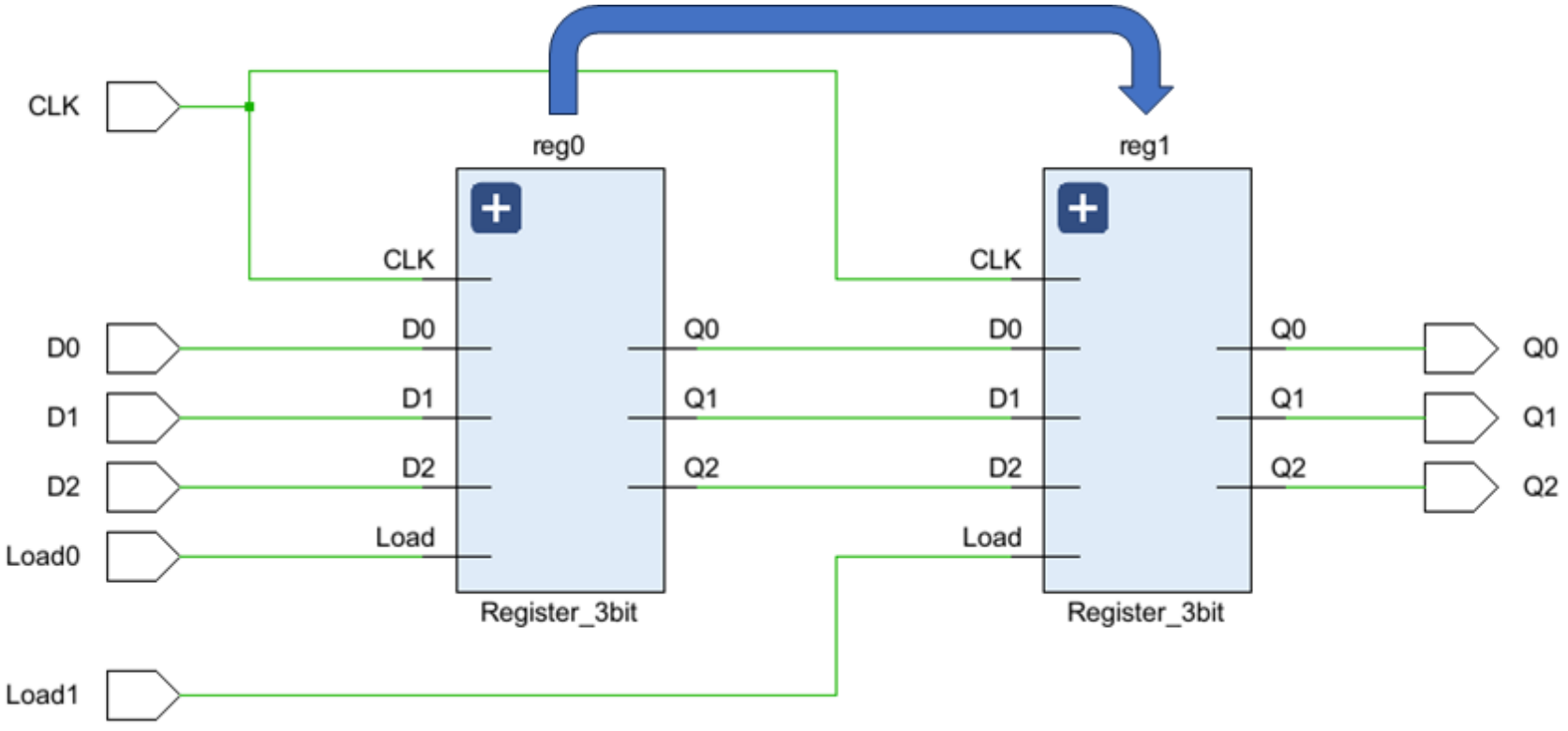

Figure 1: The schematic depicts a register transfer from reg0 to reg1

Two Register Transfer - One

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity Two_Register_3bit is
5      Port ( D0, D1, D2 : in STD_LOGIC;
6             Load0, Load1, CLK : in STD_LOGIC;
7             Q0, Q1, Q2 : out STD_LOGIC);
8  end Two_Register_3bit;
9
10 architecture Behavioral of Two_Register_3bit is
11
12     COMPONENT Register_3bit
13     Port ( D0, D1, D2 : in STD_LOGIC;
14            CLK, Load : in STD_LOGIC;
15            Q0, Q1, Q2 : out STD_LOGIC);
16     END COMPONENT;
```

Listing 1: The Two_Register_3bit entity

Two Register Transfer - Two

```
1   signal reg0toreg1_bit0, reg0toreg1_bit1, reg0toreg1_bit2
    : std_logic;
2    --  Propagation Delay according to StdentID e.g. 26 33
    57 25(DEC)
3    constant AND_gate_delay : Time := 6ns; --  6=5+1
4    constant NAND_gate_delay : Time := 3ns;--  3=2+1
5    constant OR_gate_delay : Time := 8ns;  --  8=7+1
6    constant NOR_gate_delay : Time := 6ns; --  6=5+1
7    constant XOR_gate_delay : Time := 4ns; --  4=3+1
8    constant XNOR_gate_delay : Time := 4ns;--  4=3+1
9    constant NOT_gate_delay : Time := 7ns; --  7=6+1
```

Listing 2: The signals reg0toreg1_bit0 reg0toreg1_bit1 and reg0toreg1_bit2 connect reg0 with reg1

Two Register Transfer - Three

```
1  begin
2
3     reg0: Register_3bit PORT MAP (
4          D0 => D0, D1 => D1, D2 => D2,
5          CLK => CLK,
6          Load => Load0,
7          Q0 => reg0toreg1_bit0, Q1 => reg0toreg1_bit1,
8          Q2 => reg0toreg1_bit2);
9
10    reg1: Register_3bit PORT MAP (
11         D0 => reg0toreg1_bit0, D1 => reg0toreg1_bit1,
12         D2 => reg0toreg1_bit2,
13         CLK => CLK,
14         Load => Load1,
15         Q0 => Q0, Q1 => Q1, Q2 => Q2);
16
17 end Behavioral;
```

Listing 3: The Register_3bit entity is instantiates twice (reg0 and reg1)

Register with three D Flip Flops

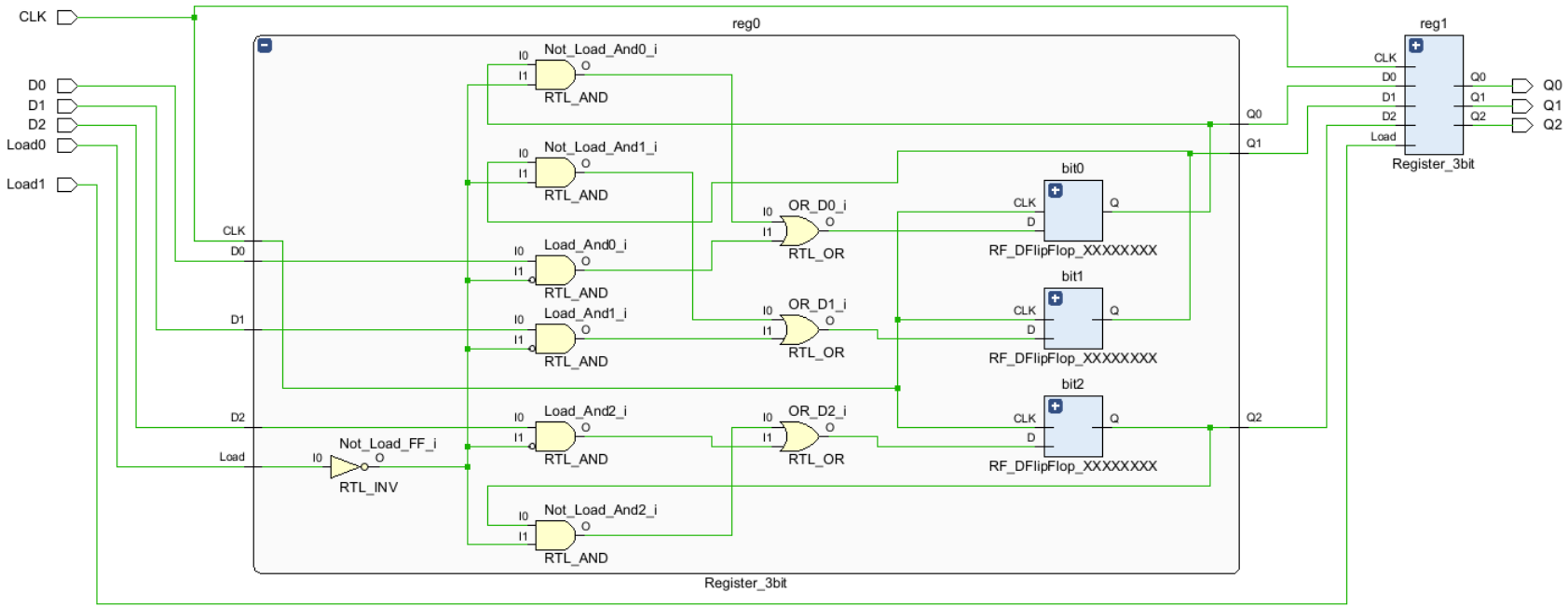

Figure 2: The schematic depicts registers reg0 and reg1.The register instantiates three D Flip Flops including **Load logic**

Register Implementation - One

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity Register_3bit is
5      Port ( D0, D1, D2 : in STD_LOGIC;
6             CLK, Load : in STD_LOGIC;
7             Q0, Q1, Q2 : out STD_LOGIC);
8  end Register_3bit;
9
10 architecture Behavioral of Register_3bit is
11
12     COMPONENT RF_DFlipFlop_XXXXXXXX
13     Port ( CLK, D, Load : in STD_LOGIC;
14            Q : out STD_LOGIC);
15     END COMPONENT;
```

Listing 4: Register_3bit entity

Register Implementation - Two

```
1     signal Q_bit0, Q_bit1, Q_bit2 : std_logic;
2     signal OR_D0, OR_D1, OR_D2 : std_logic;
3     signal Not_Load_FF, Load_FF : std_logic;
4     signal Not_Load_And0, Not_Load_And1, Not_Load_And2:
      std_logic;
5     signal Load_And0, Load_And1, Load_And2: std_logic;
6
7     --  Propagation Delay according to StdentID e.g. 26 33
      57 25(DEC)
8     constant AND_gate_delay : Time := 6ns; -- 6 =5+1
9     constant NAND_gate_delay : Time := 3ns;-- 3=2+1
10    constant OR_gate_delay : Time := 8ns;  -- 8=7+1
11    constant NOR_gate_delay : Time := 6ns; -- 6=5+1
12    constant XOR_gate_delay : Time := 4ns; -- 4=3+1
13    constant XNOR_gate_delay : Time := 4ns;-- 4=3+1
14    constant NOT_gate_delay : Time := 7ns; -- 7=6+1
```

Listing 5: signals and constants

Register Implementation - Three

```
1  begin
2
3     Not_Load_FF <= not Load after NOT_gate_delay;
4     Load_FF <= not Not_Load_FF after NOT_gate_delay;
5
6     -- Instantiate the least significant bit
7     bit0: RF_DFlipFlop_XXXXXXXX PORT MAP (
8           CLK => CLK, D => OR_D0, Q => Q_bit0 );
9
10    Not_Load_And0 <= Q_bit0 and Not_Load_FF after
       AND_gate_delay;
11    OR_D0 <= Not_Load_And0 or Load_And0 after OR_gate_delay;
12    Load_And0 <= D0 and Load_FF after AND_gate_delay;
13    Q0 <= Q_bit0;
14
15      bit1: RF_DFlipFlop_XXXXXXXX PORT MAP (
16          CLK => CLK, D => OR_D1, Q => Q_bit1 );
```

Listing 6: The Register_3bit entity instantiates three D Flip Flops

Register Implementation - Four

```
1     Not_Load_And1 <= Q_bit1 and Not_Load_FF after
       AND_gate_delay;
2     OR_D1 <= Not_Load_And1 or Load_And1 after OR_gate_delay;
3     Load_And1 <= D1 and Load_FF after AND_gate_delay;
4     Q1 <= Q_bit1;
5
6     -- Instantiate the most significant bit
7     bit2: RF_DFlipFlop_XXXXXXXX PORT MAP (
8             CLK => CLK, D => OR_D2, Q => Q_bit2);
9
10    Not_Load_And2 <= Q_bit2 and Not_Load_FF after
       AND_gate_delay;
11    OR_D2 <= Not_Load_And2 or Load_And2 after OR_gate_delay;
12    Load_And2 <= D2 and Load_FF after AND_gate_delay;
13    Q2 <= Q_bit2;
14
15 end Behavioral;
```

Listing 7: The Register_3bit entity instantiates three D Flip Flops

Register with three D Flip Flops and Load signal

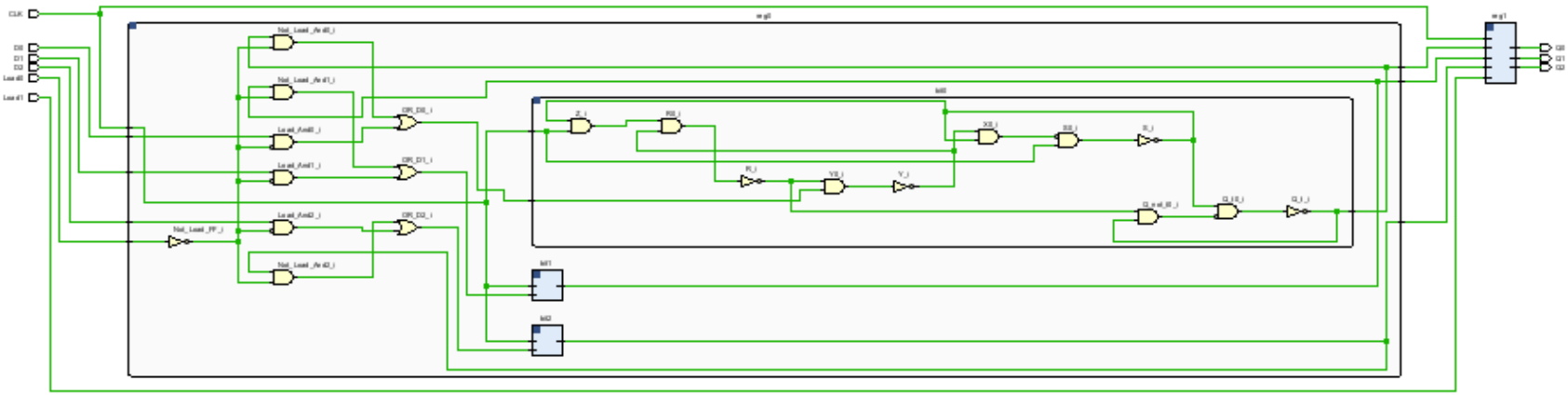

Figure 3: The schematic depicts the gate logic of one D Flip Flops

New D Flip Flop Version

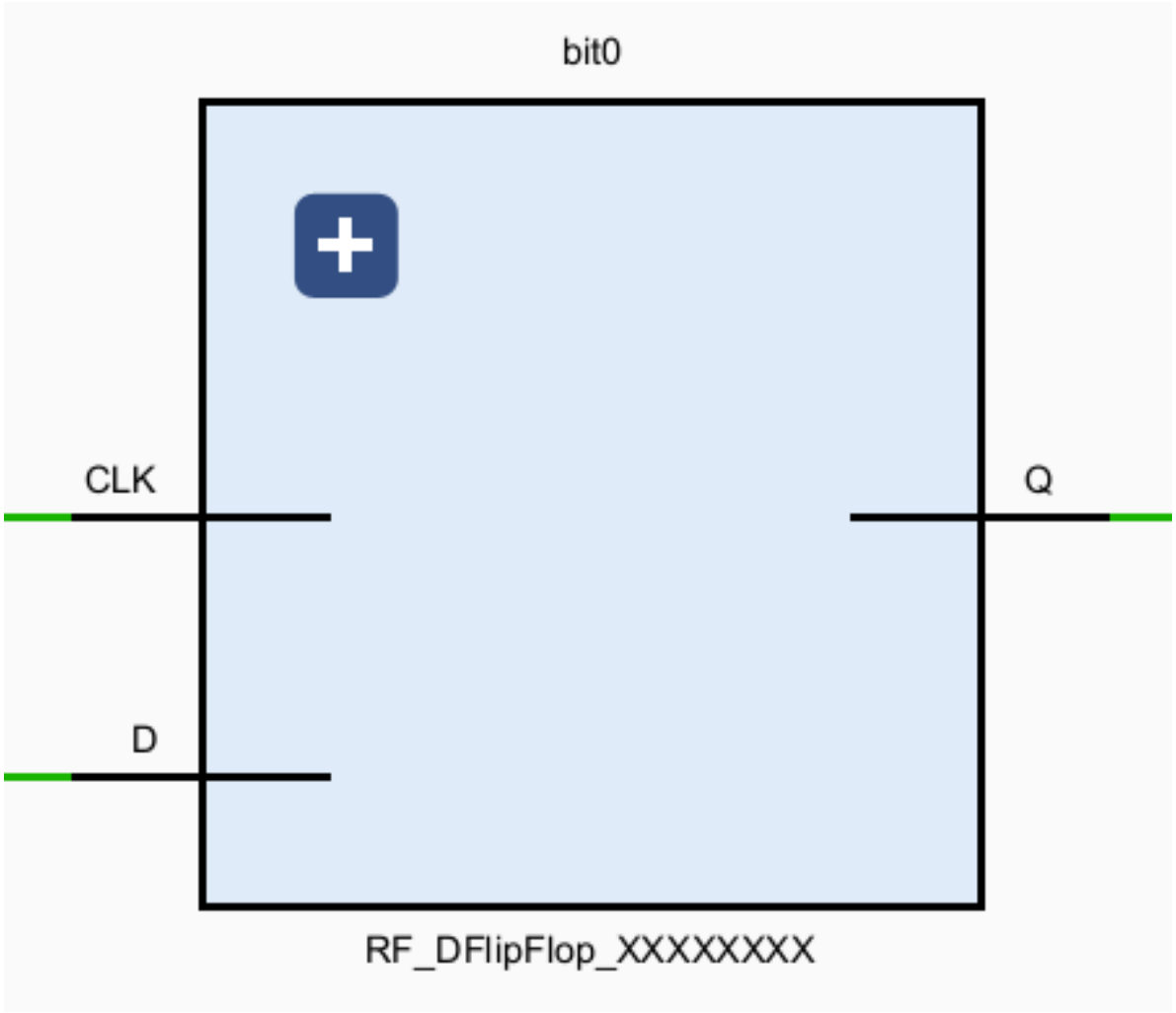

Figure 4: This D Flip Flops has no Q_not port

D Flip Flop Implementation - One

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity RF_DFlipFlop_XXXXXXXX is
5     Port ( CLK, D : in STD_LOGIC;
6            Q : out STD_LOGIC);
7 end RF_DFlipFlop_XXXXXXXX;
```

Listing 8: RF_DFlipFlop_XXXXXXXX entity

D Flip Flop Implementation - Two

```
1  architecture Behavioral of RF_DFlipFlop_XXXXXXXX is
2
3      signal Q_t, Q_not_t, X, S, R, Y, Z : std_logic;
4
5      --  Propagation Delay according to StdentID e.g. 26 33 57
         25(DEC)
6      constant AND_gate_delay : Time := 6ns;       --  6 = 5 + 1
7      constant NAND_gate_delay : Time := 3ns;      --  3 = 2 + 1
8      constant OR_gate_delay : Time := 8ns;        --  8 = 7 + 1
9      constant NOR_gate_delay : Time := 6ns;       --  6 = 5 + 1
10     constant XOR_gate_delay : Time := 4ns;       --  4 = 3 + 1
11     constant XNOR_gate_delay : Time := 4ns;      --  4 = 3 + 1
12     constant NOT_gate_delay : Time := 7ns;       --  7 = 6 + 1
```

Listing 9: signals and constants

D Flip Flop Implementation - Three

```
1  begin
2
3      X <= Y nand S after NAND_gate_delay;
4      S <= X nand CLK after NAND_gate_delay;
5      Z <= S and CLK after AND_gate_delay;
6      R <= Z nand Y after NAND_gate_delay;
7      Y <= R nand D after NAND_gate_delay;
8      Q_t <= S nand Q_not_t after NAND_gate_delay;
9      Q_not_t <= R nand Q_t after NAND_gate_delay;
10     Q <= Q_t;
11
12 end Behavioral;
```

Listing 10: The architecture with Load port and Q_not port removed

Register Load Logic

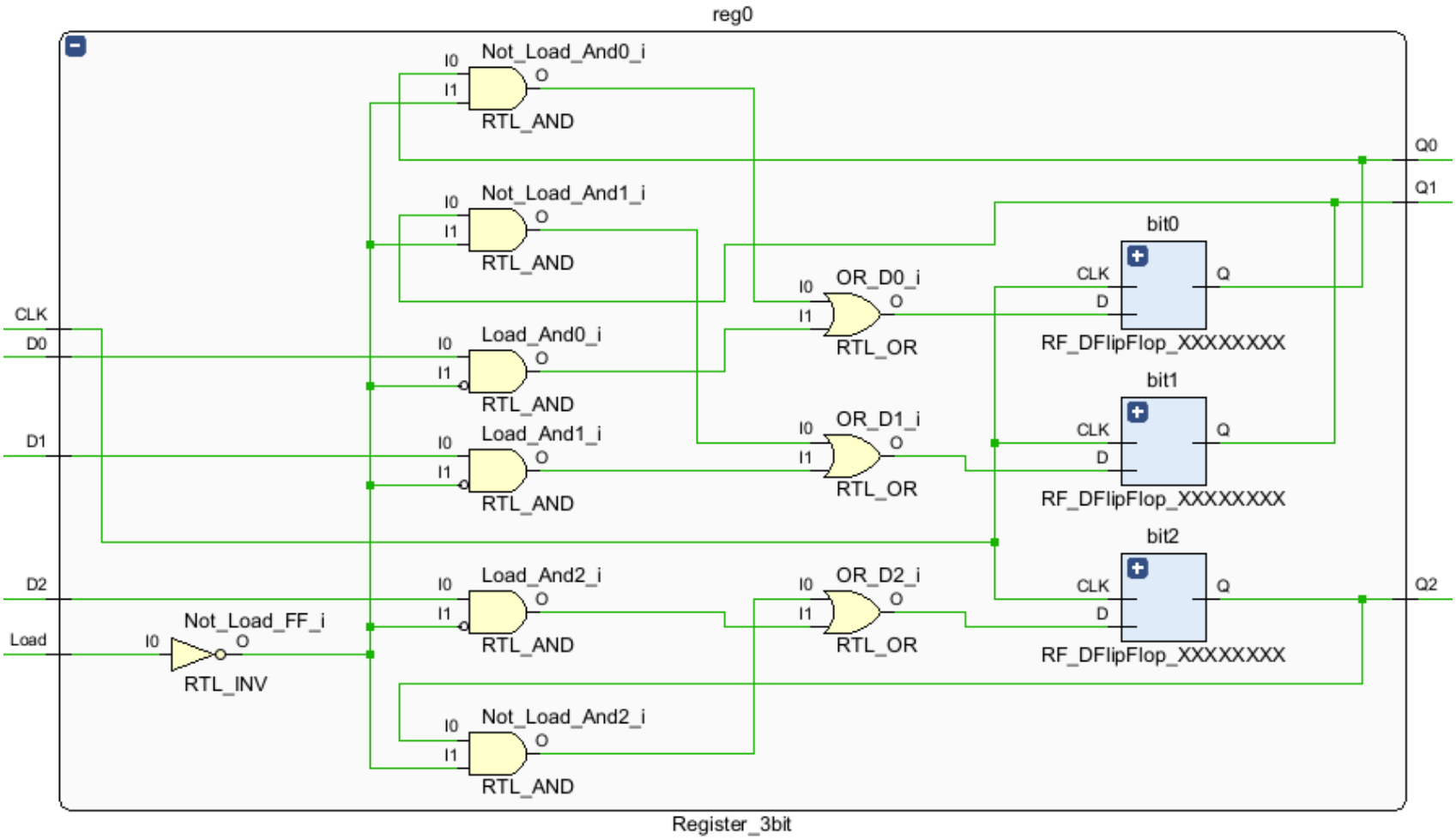

Figure 5: The logic puts the flip flop output Q back to its input D if the load signal is not set.

Two Register Transfer Timing Diagram - One

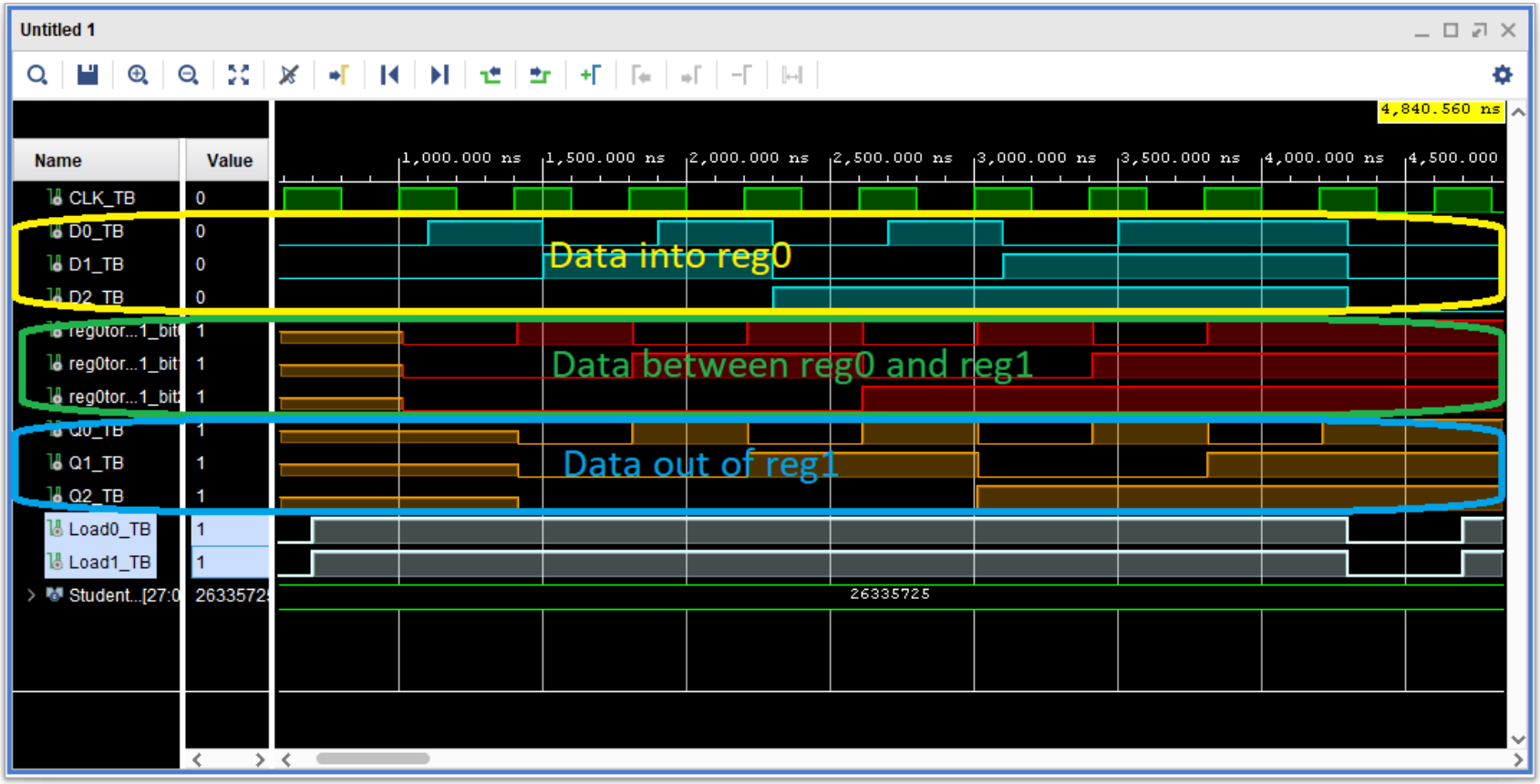

Figure 6: This timing diagram depicts 3-bit values going into reg0, 3-bit values between reg0 and reg1, and 3-bit values at the output of reg1.

Two Register Transfer Timing Diagram - Two

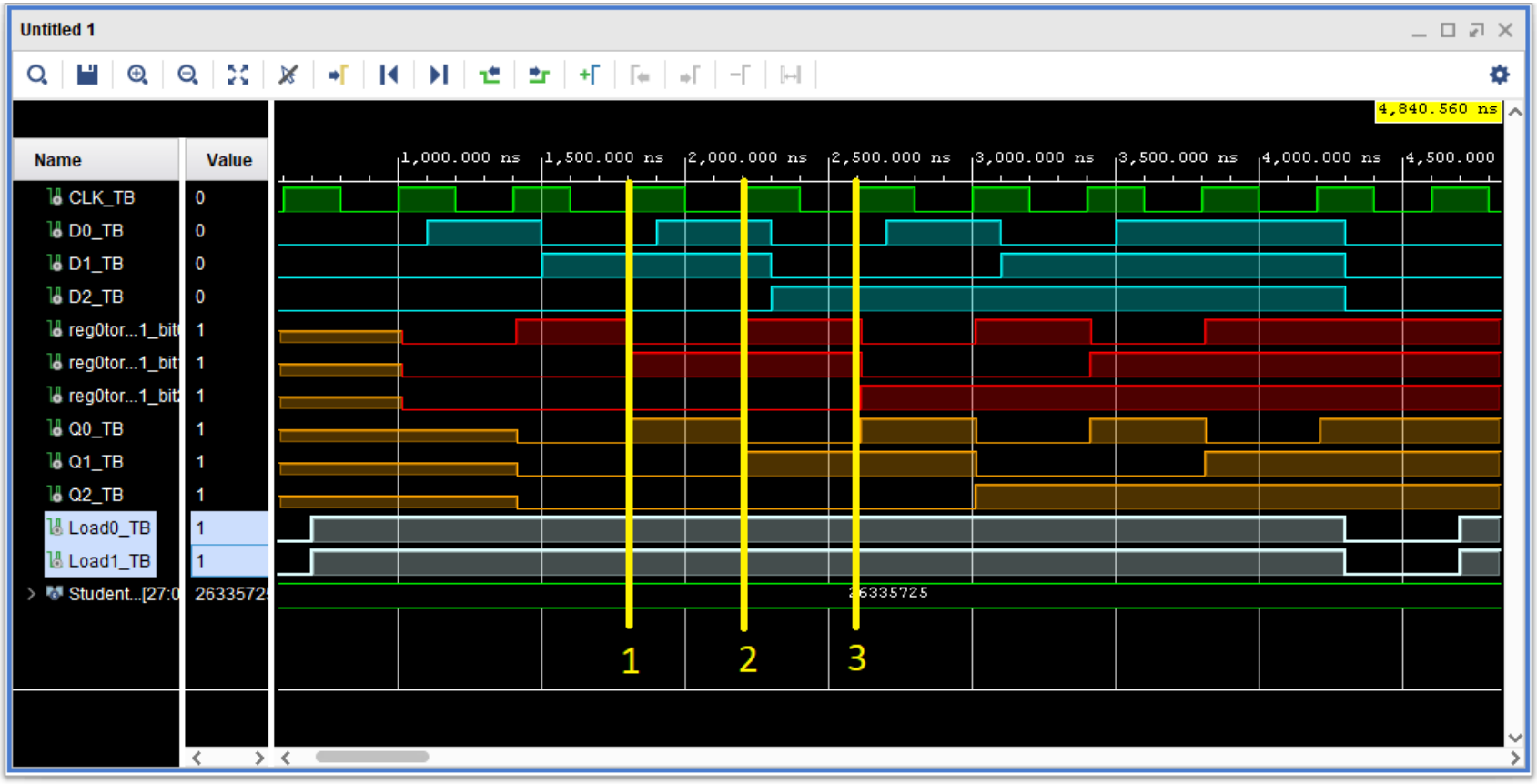

Figure 7: At every rising edge of the clock, we can observe how 3-bit values move from the input of reg0 to its output and at the same time the 3-bit values between the registers move to the output of reg1.