LAB 05 – COUNTER – 210495G – Nipun Viraj

- Task

We were assigned to do build a counter to represent a set of LEDs which take turns switching on and off either clockwise or anti-clockwise according to a button. We should use a D flip flop to determine the output of the next data set and also use a clock.

2. Completed Table

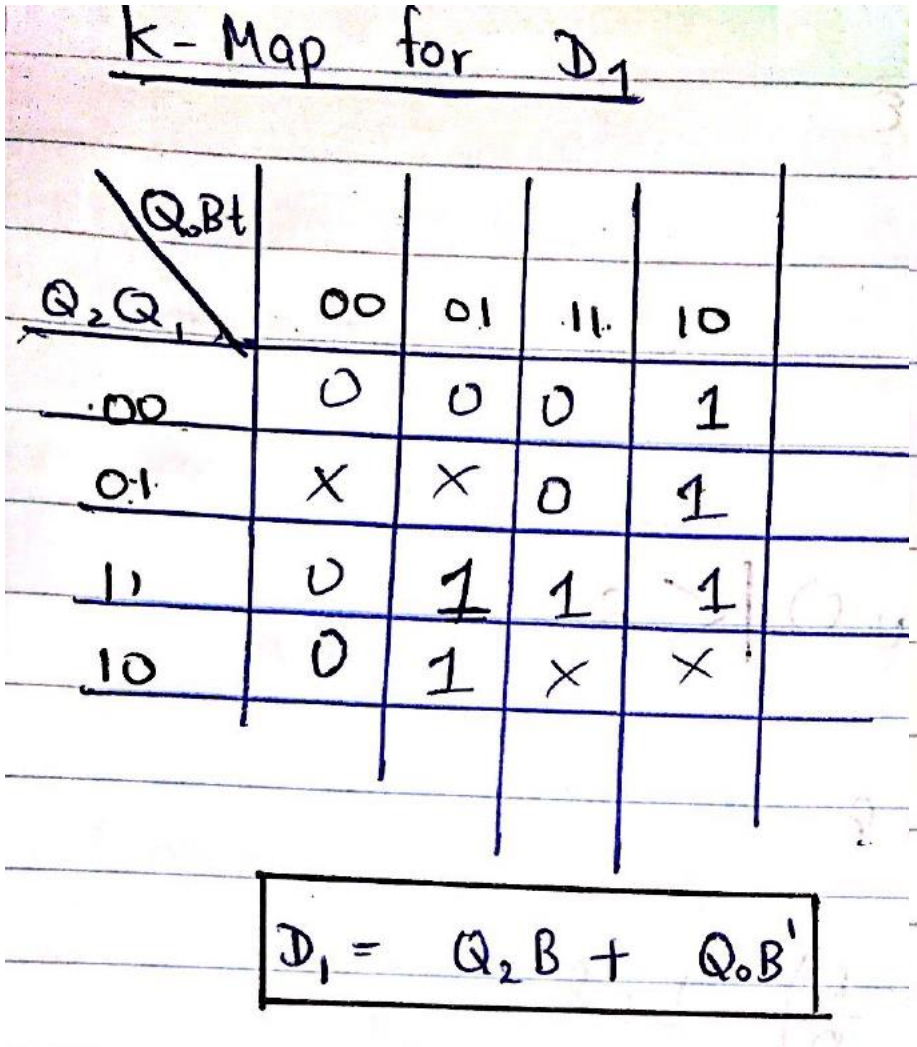
Q_t			Button	Q_{t+1}			D_2	D_1	D_0
Q_2	Q_1	Q_0		Q_2	Q_1	Q_0			
0	0	0	0	0	0	1	0	0	1
0	0	0	1	1	0	0	1	0	0
0	0	1	0	0	1	1	0	1	1
0	0	1	1	0	0	0	0	0	0
0	1	1	0	1	1	1	1	1	1
0	1	1	1	0	0	1	0	0	1
1	1	1	0	1	1	0	1	1	0
1	1	1	1	0	1	1	0	1	1
1	1	0	0	1	0	0	1	0	0
1	1	0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0	0	0
1	0	0	1	1	1	0	1	1	0

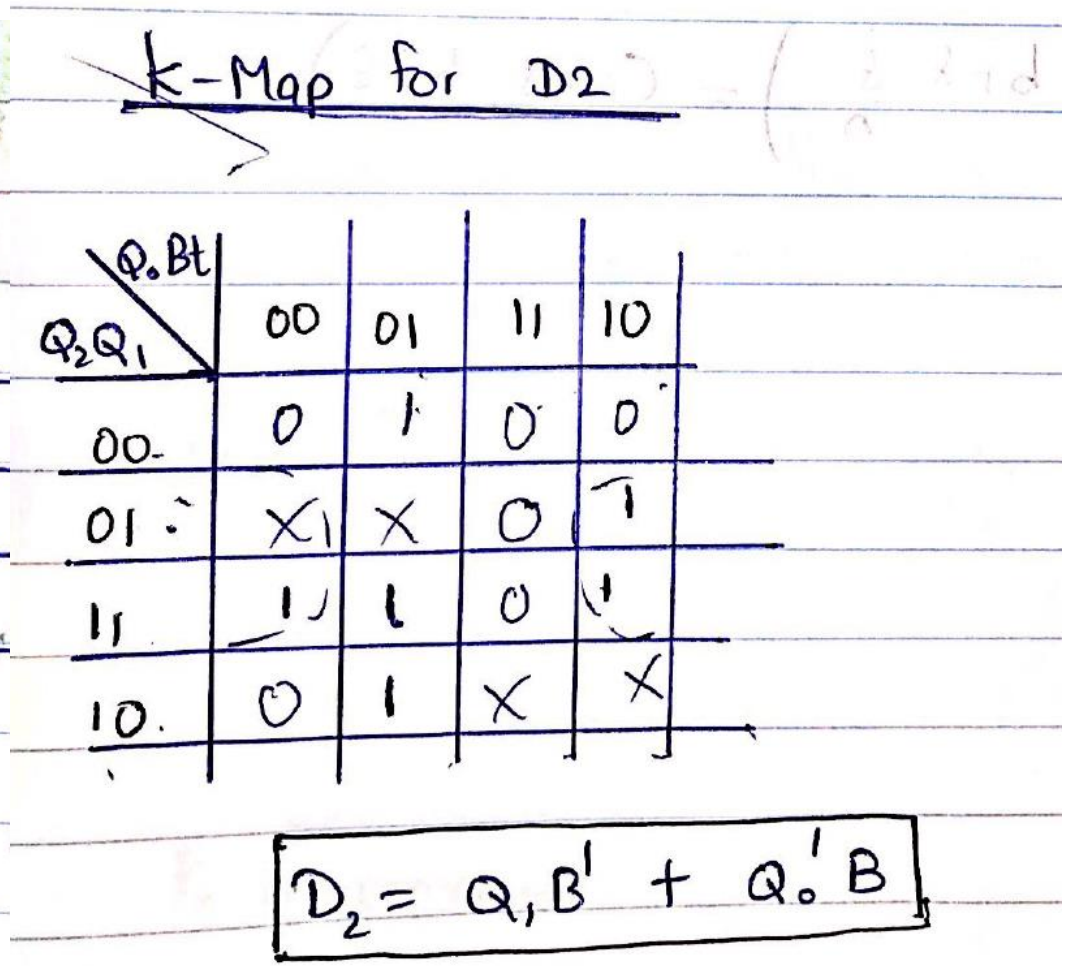
k-Map for D0

Q_0 Button \ Q_2Q_1	Q_2Q_1 00	Q_2Q_1 01	Q_2Q_1 11	Q_2Q_1 10
00	1	X	0	0
01	0	X	1	0
11	0	1	1	X
10	1	1	0	X

$$D_0 = Q_1B + Q_2'B'$$

2. Completed Table





3. Design Sources

- 1. D Flipflop

```
entity D_FF is
    Port ( D : in STD_LOGIC;
           Res : in STD_LOGIC;
           Clk : in STD_LOGIC;
           Q : out STD_LOGIC;
           Qbar : out STD_LOGIC);
end D_FF;

architecture Behavioral of D_FF is

begin
    process(Clk) begin
        if (rising_edge(Clk)) then
            if Res='1' then
                Q<='0';
                Qbar <= '1';
            else
                Q <= D;
                Qbar <= not D;
            end if;
         end if;
    end process;

end Behavioral;
```

3. Design Sources

- 2. Slowdown Counter

```
entity Slow_Clk is
    Port ( Clk_in : in STD_LOGIC;
           Clk_out : out STD_LOGIC);
end Slow_Clk;

architecture Behavioral of Slow_Clk is

signal count: integer :=1;
signal clk_status : std_logic :='0';

begin
process (Clk_in) begin
    if (rising_edge(Clk_in)) then
        count <= count+1;
        if (count=5) then
            clk_status <= not clk_status;
            Clk_out <= clk_status;
            count <= 1;
        end if;
    end if;
end process;
end Behavioral;
```

3. Design Sources

- 3. Counter

```
34  entity Counter is
35      Port ( Dir : in STD_LOGIC;
36             Res : in STD_LOGIC;
37             Clk : in STD_LOGIC;
38             Q : out STD_LOGIC_VECTOR (2 downto 0));
39  end Counter;
40
41  architecture Behavioral of Counter is
42
43      COMPONENT D_FF
44      PORT (
45          D : in STD_LOGIC;
46          Res : in STD_LOGIC;
47          Clk : in STD_LOGIC;
48          Q : out STD_LOGIC;
49          Qbar : out STD_LOGIC);
50      END COMPONENT;
51
52      COMPONENT Slow_Clk
53      PORT (
54          Clk_in : in STD_LOGIC;
55          Clk_out : out STD_LOGIC);
56      END COMPONENT;
57
58      signal D0,D1,D2 : std_logic;
59      signal Q0,Q1,Q2 : std_logic;
60      signal Clk_slow : std_logic;
61
62  begin
63
64      Slow_Clk0 : Slow_Clk
65          PORT MAP (
66          Clk_in => Clk,
67          Clk_out => Clk_slow);
```

```
66          Clk_in => Clk,
67          Clk_out => Clk_slow);
68
69      D0 <= (Q1 AND Dir) OR (NOT(Q2) AND NOT(Dir));
70      D1 <= (Q0 AND NOT(Dir)) OR (Q2 AND Dir);
71      D2 <= (Q1 AND NOT(Dir)) OR (NOT(Q0) AND Dir);
72
73      D_FF0 : D_FF
74          PORT MAP (
75          D => D0,
76          Res => Res,
77          Clk => Clk_slow,
78          Q => Q0);
79
80      D_FF1 : D_FF
81                  PORT MAP (
82                  D => D1,
83                  Res => Res,
84                  Clk => Clk_slow,
85                  Q => Q1);
86
87
88      D_FF2 : D_FF
89                  PORT MAP (
90                  D => D2,
91                  Res => Res,
92                  Clk => Clk_slow,
93                  Q => Q2);
94
95          Q(0) <= Q0;
96          Q(1) <= Q1;
97          Q(2) <= Q2;
98  end Behavioral;
99
```

4. Test Bench Codes

- 1. D Flipflop

```
entity D_FF_Sim is
--  Port ( );
end D_FF_Sim;

architecture Behavioral of D_FF_Sim is

COMPONENT D_FF
    PORT (D, Res, Clk : IN STD_LOGIC;
          Q, Qbar : OUT STD_LOGIC);
END COMPONENT;

SIGNAL D,Res,Clk : std_logic;
SIGNAL Q,Qbar : std_logic;

begin

UUT : D_FF PORT MAP(
    D => D,
    Res => Res,
    Clk => Clk,
    Q => Q,
    Qbar => Qbar);

process
begin

Res <= '0';
D <= '0';
Clk <= '0';

WAIT FOR 100ns;

Res <= '1';
Clk <= '0';

WAIT FOR 100ns;

Res <= '1';
Clk <= '1';

WAIT FOR 100ns;

Res <= '0';
D <= '1';
Clk <= '0';

WAIT FOR 100ns;

Clk <= '1';

WAIT FOR 100ns;

Res <= '1';
Clk <= '0';

WAIT FOR 100ns;

Res <= '1';
Clk <= '1';

WAIT;
end process;
end Behavioral;
```

4. Test Bench Codes

- 2. Slow Down Counter

```
34  entity Slow_Clk_Sim is
35  --  Port ( );
36  end Slow_Clk_Sim;
37
38  architecture Behavioral of Slow_Clk_Sim is
39
40  COMPONENT Slow_Clk
41      PORT (Clk_in : IN STD_LOGIC;
42            Clk_out : OUT STD_LOGIC);
43  END COMPONENT;
44
45      SIGNAL Clk_in : std_logic;
46      SIGNAL Clk_out : std_logic;
47
48  begin
49
50  UUT: Slow_Clk PORT MAP (
51      Clk_in => Clk_in,
52      Clk_out => Clk_out);
53
54   process
55   begin
56   Clk_in <= '0';
57   WAIT FOR 20ns;
58   Clk_in <= '1';
59   WAIT FOR 20ns;
60    Clk_in <= '0';
61   WAIT FOR 20ns;
62   Clk_in <= '1';
63   WAIT FOR 20ns;
64    Clk_in <= '0';
65   WAIT FOR 20ns;
66   Clk_in <= '1';
67   WAIT FOR 20ns;
```

```
67   WAIT FOR 20ns;
68    Clk_in <= '0';
69   WAIT FOR 20ns;
70   Clk_in <= '1';
71   WAIT FOR 20ns;
72    Clk_in <= '0';
73   WAIT FOR 20ns;
74   Clk_in <= '1';
75   WAIT FOR 20ns;
76    Clk_in <= '0';
77   WAIT FOR 20ns;
78   Clk_in <= '1';
79   WAIT FOR 20ns;
80    Clk_in <= '0';
81   WAIT FOR 20ns;
82   Clk_in <= '1';
83   WAIT FOR 20ns;
84    Clk_in <= '0';
85   WAIT FOR 20ns;
86   Clk_in <= '1';
87   WAIT FOR 20ns;
88    Clk_in <= '0';
89   WAIT FOR 20ns;
90   Clk_in <= '1';
91   WAIT FOR 20ns;
92    Clk_in <= '0';
93   WAIT FOR 20ns;
94   Clk_in <= '1';
95   WAIT FOR 20ns;
96    Clk_in <= '0';
97   WAIT FOR 20ns;
98   Clk_in <= '1';
99   WAIT FOR 20ns;
100   Clk_in <= '0';
```

4. Test Bench Codes

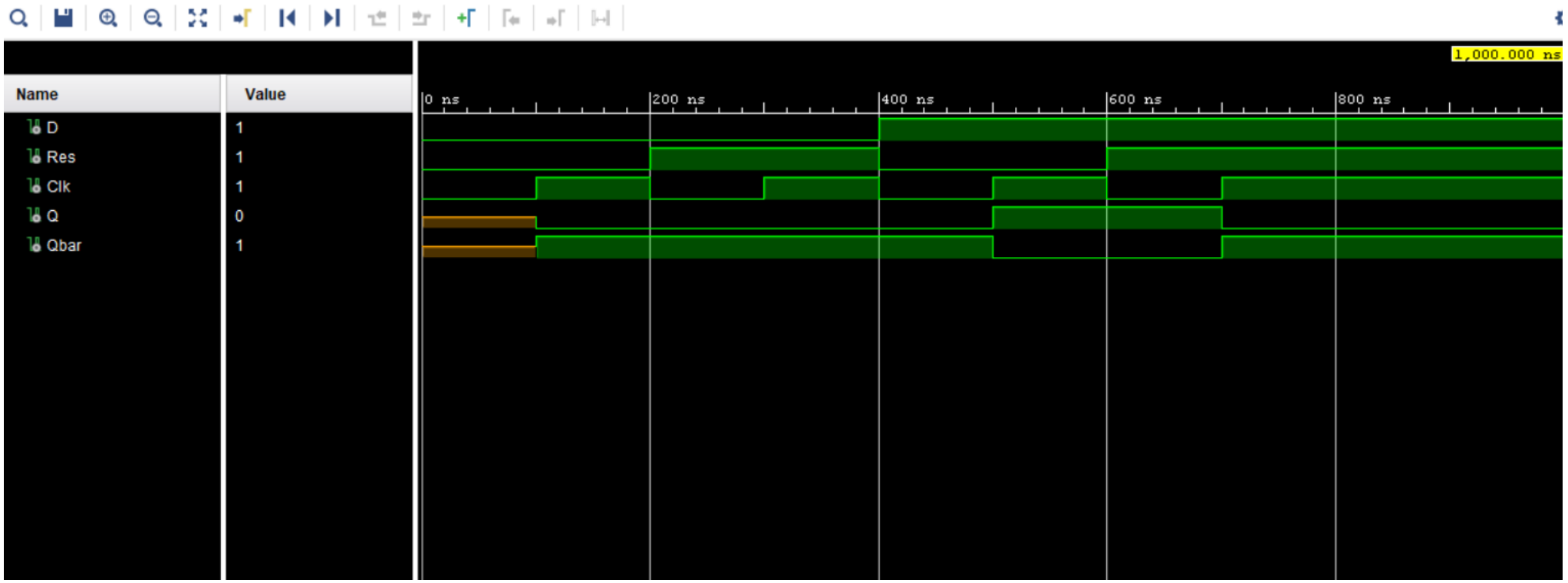
- 3. Counter

```
34  entity Counter_sim is
35  --  Port ( );
36  end Counter_sim;
37
38  architecture Behavioral of Counter_sim is
39  COMPONENT Counter
40      PORT( Dir,Res,Clk : IN STD_LOGIC;
41      Q : OUT STD_LOGIC_VECTOR(2 downto 0));
42  END COMPONENT;
43
44  SIGNAL Dir,Res,Clk : std_logic;
45  SIGNAL Q : std_logic_vector(2 downto 0);
46
47  begin
48  UUT: Counter PORT MAP(
49  Dir => Dir,
50  Res => Res,
51  Clk => Clk,
52  Q=>Q
53  );
54
55  process
56      begin
57      Dir <= '0';
58      Res <= '1';
59      for i in 1 to 20 loop
60          Clk <= '0';
61          WAIT FOR 2 ns;
62          Clk <= '1';
63          WAIT FOR 2 ns;
64          end loop;
65          WAIT FOR 50 ns;
```

```
55  process
56      begin
57      Dir <= '0';
58      Res <= '1';
59      for i in 1 to 20 loop
60          Clk <= '0';
61          WAIT FOR 2 ns;
62          Clk <= '1';
63          WAIT FOR 2 ns;
64          end loop;
65          WAIT FOR 50 ns;
66          Res <= '0';
67      for i in 1 to 200 loop
68          Clk <= '0';
69          WAIT FOR 2 ns;
70          Clk <= '1';
71          WAIT FOR 2 ns;
72          end loop;
73          Dir <= '1';
74          WAIT FOR 50 ns;
75      for i in 1 to 200 loop
76          Clk <= '0';
77          WAIT FOR 2 ns;
78          Clk <= '1';
79          WAIT FOR 2 ns;
80          end loop;
81          WAIT;
82
83  end process;
84  end Behavioral;
```

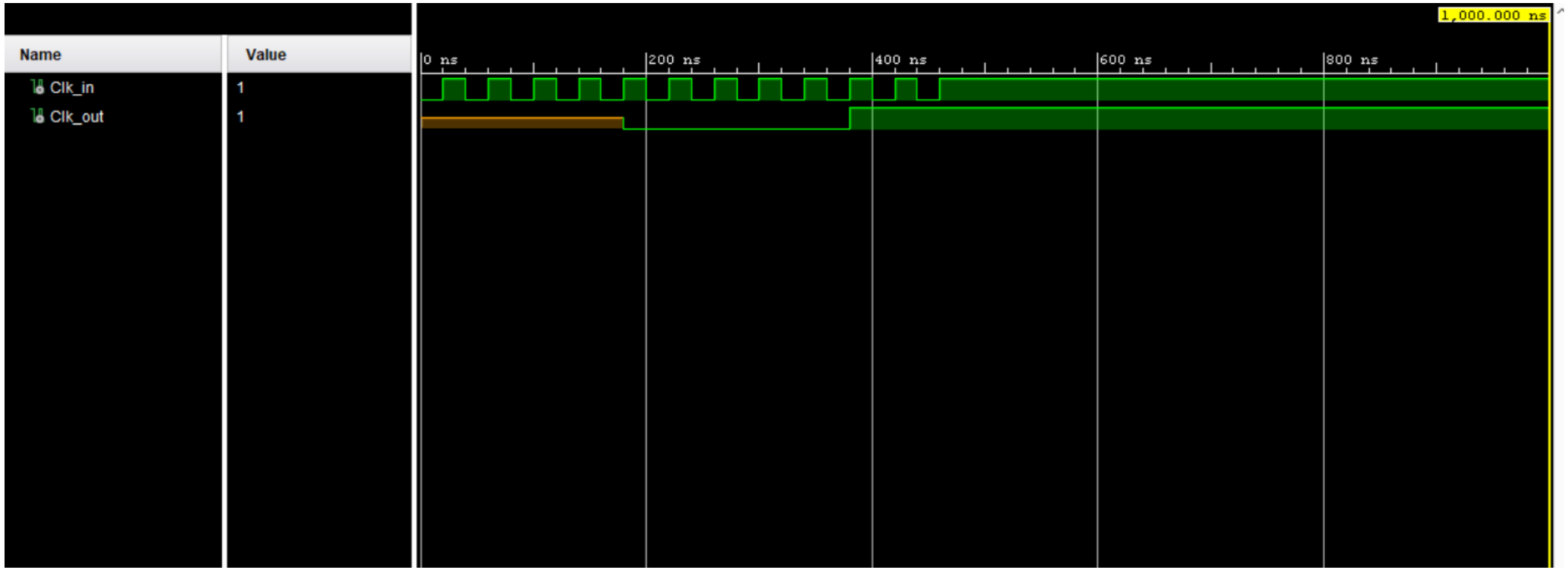
4. Timing Diagrams

- 1. D Flipflop



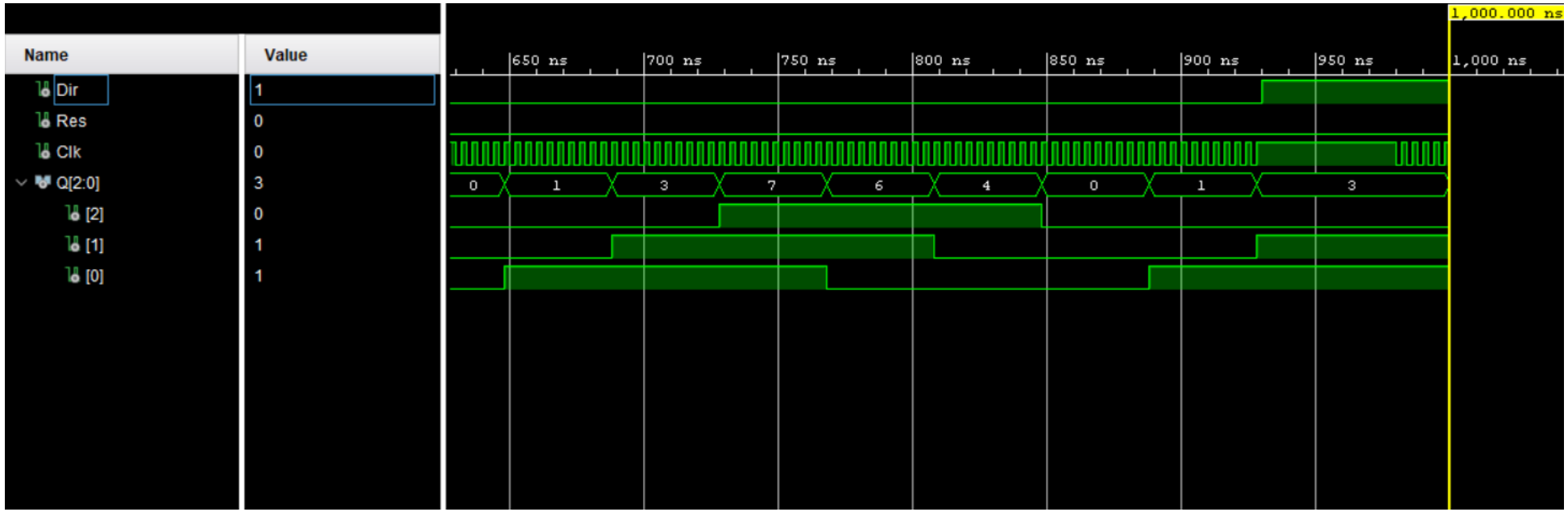
4. Timing Diagrams

- 2. Slowdown Counter



4. Timing Diagrams

- 3. Counter



5. Conclusions

We can build a counter circuit using VHDL behavioral modelling and high-level programming commands and the hierarchical design available. We also can even slow down the initial count rate according to our requirements using commands. Even the direction of counting can be controlled using a button in the Basys board.

Overall, we can say that VHDL behavioral modelling can be used to design complex sequential logic circuits.