D-FlipFlop:

VHDL Module:

```
library ieee;
use ieee.std_logic_1164.all;
entity d is
port(
D : in std_logic;
CLK : in std_logic;
RESET : in std_logic;
Q : out std_logic);
end d;
architecture behavioral of d is
begin
process (CLK, RESET)
begin
if (RESET = '1') then
Q <= '0';
elsif (CLK'event and CLK = '1') then
Q <= D;
end if;
end process;
end;
```

VHDL TestBench:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity DFF_tb is
```

```
end DFF_tb;
architecture behavior of DFF_tb is
--Signal declarations
component DFF
port(CLK, RESET, D : in std_logic;
Q : out std_logic);
end component;
signal clk,reset, d, q: std_logic := '0';
-- Clock period definitions
constant clk_period : time := 100 ns;
shared variable simend : boolean := false;

begin
-- Instantiate the Unit Under Test (UUT)
UUT :DFF port map (clk => CLK,reset  => RESET, d => D, q  => Q);
-- Clock process definitions
clk_process :process
begin
while simend=false loop
clk <= not clk;
wait for clk_period/2;
end loop;
wait;
end process;
-- Stimulus process
stim_proc: process
begin
D <='0';
```

```
wait for clk_period*2;


D <='1';
wait for clk_period*2;
reset <='1';
D <='0';
wait for clk_period*2;
D <='1';
wait for clk_period*2;
simend := true;
wait;
end process;
end;
```

Output(D-flipflop):

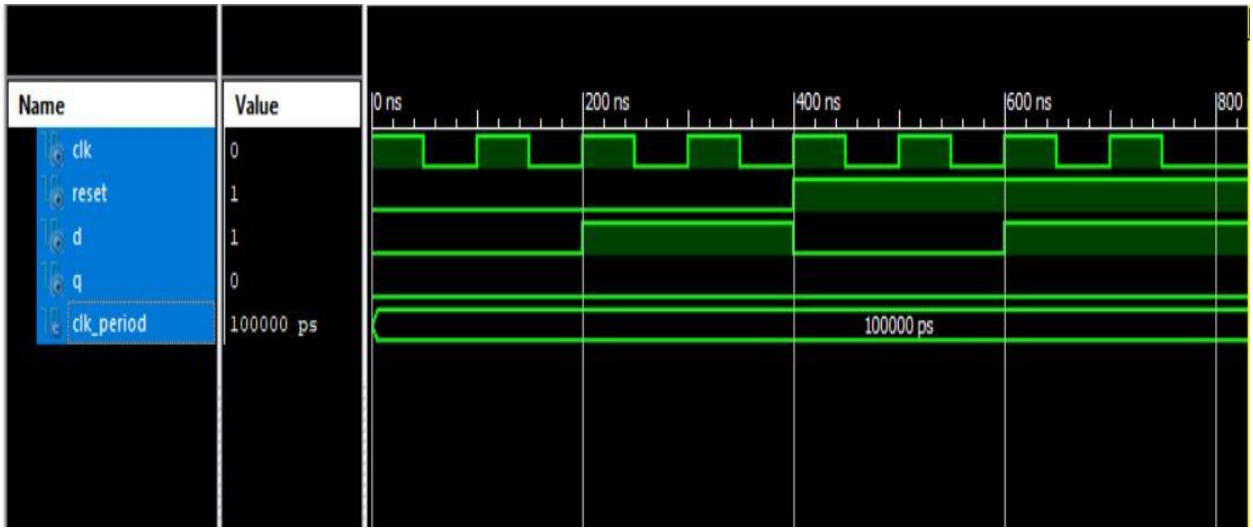

Figure : Simulation output of D-flipflop

JK-Flipflop:

VHDL Module:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity JK_Flipflop is
port ( clk:        in std_logic;
J, K:        in std_logic;
Q, Qbar:     out std_logic;
reset:       in std_logic
);
end JK_Flipflop;
--architecture of entity
architecture Behavioral of JK_Flipflop is
--signal declaration.
signal qtemp: std_logic :='0';
signal qbartemp: std_logic :='1';
begin
Q <= qtemp;
Qbar <= qbartemp;
process(clk,reset)
begin
if(reset = '1') then          --Reset the output.
qtemp <= '0';
qbartemp <= '1';
elsif( rising_edge(clk) ) then
if(J='0' and K='0') then       --No change in the output
NULL;
elsif(J='0' and K='1') then    --Reset the output.
```

```
qtemp <= '0';
qbartemp <= '1';
elsif(J='1' and K='0') then    --Set the output.
qtemp <= '1';
qbartemp <= '0';
else                           --Toggle the output.
qtemp <= not qtemp;
qbartemp <= not qbartemp;
end if;
end if;
end process;
end Behavioral;
```

VHDL Testbench:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity JK_Flipflop_tb is
end JK_Flipflop_tb;
architecture behavior of JK_Flipflop_tb is
--Signal declarations
component JK_Flipflop
port ( clk:        in std_logic;
J, K:        in std_logic;
Q, Qbar:     out std_logic;
reset:       in std_logic);
end component;
signal clk,J,K,reset,Q,Qbar : std_logic := '0';
-- Clock period definitions
constant clk_period : time := 100 ns;
```

```
shared variable simend : boolean := false;
begin
-- Instantiate the Unit Under Test (UUT)
UUT : JK_Flipflop port map (clk,J,K,Q,Qbar,reset);
-- Clock process definitions
clk_process :process
begin
while simend=false loop
clk <= not clk;
wait for clk_period/2;
end loop;
wait;
end process;
stim_proc: process
begin
J<='0';K<='0';
wait for clk_period*2;
J<='1';
K<='0';
wait for clk_period*2;
J<='1';
K<='1';
wait for clk_period*2;
J<='0';
K<='1';
wait for clk_period*2;

J<='0';
```

```
K<='0';
wait for clk_period*2;
J<='1';
K<='0';
wait for clk_period*2;
reset <='1';
J<='1';
K<='1';
wait for clk_period*2;
J<='0';
K<='1';
wait for clk_period*2;
reset <='0';
J<='1';
K<='1';
simend := true;
wait;
end process;
end;
```

Output(JK-flipflop)

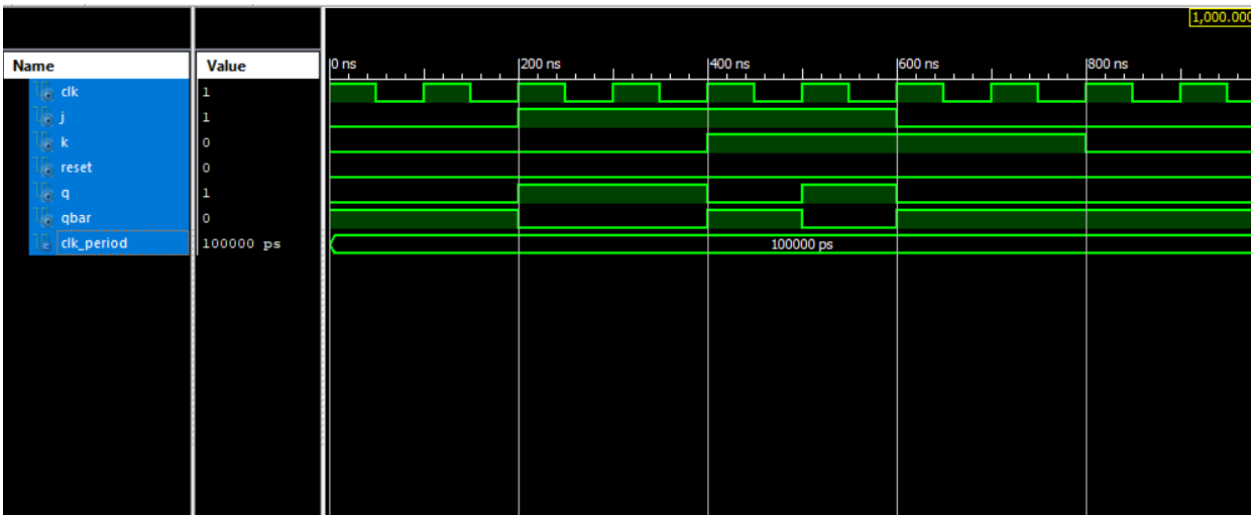

Figure : Simulation output of JK flipflop

4-Bit Register:

VHDL Module:

```
library ieee;
use ieee.std_logic_1164.all;
entity dff is
port(
clk, reset, d : in std_logic;
q : out std_logic          );
end dff;
architecture behavioral_dff of dff is
begin
process (clk, reset)
begin
if (reset = '1') then
q <= '0';
elsif (clk'event and clk = '1') then
q <= d;
end if;
end process;
end behavioral_dff;
library ieee;
use ieee.std_logic_1164.all;
entity reg is
port (
d: in std_logic_vector(3 downto 0);
clk, clear: in std_logic;
q: out std_logic_vector(3 downto 0)
);
```

```
end reg;
architecture behavioral_reg of reg is
component dff
port(
clk, reset, d : in std_logic;
q : out std_logic
);
end component;
signal reset: std_logic:= '0';
begin
dff0: dff
port map (clk, reset, d(0), q(0));
dff1: dff
port map (clk, reset, d(1), q(1));
dff2: dff
port map (clk, reset, d(2), q(2));
dff3: dff
port map (clk, reset, d(3), q(3));

end behavioral_reg;
```

VHDL Testbench:

```
library ieee;
use ieee.std_logic_1164.all;
entity reg_tb is
end reg_tb;
architecture behavioral_reg_tb of reg_tb is
component reg
port(
```

```
d: in std_logic_vector(3 downto 0);
clk, clear: in std_logic;
q: out std_logic_vector(3 downto 0)
);
end component;

signal d, q: std_logic_vector(3 downto 0);
signal clk, clear: std_logic:= '0';

constant clk_period : time := 100 ns;
shared variable simend : boolean := false;

begin
uut : reg port map (clk => clk,clear  => clear, d => d, q  => q);

clk_process :process
begin
while simend=false loop
clk <= not clk;
wait for clk_period/2;
end loop;
wait;
end process;

stim_proc: process
begin
d(0) <= '0'; d(1) <= '0'; d(2) <= '0'; d(3) <= '0';
wait for clk_period*2;
```

```
d(0) <= '0'; d(1) <= '1'; d(2) <= '0'; d(3) <= '1';
wait for clk_period*2;
clear <= '1';
d(0) <= '1'; d(1) <= '0'; d(2) <= '1'; d(3) <= '1';
wait for clk_period*2;
d(0) <= '0'; d(1) <= '0'; d(2) <= '1'; d(3) <= '1';
wait for clk_period*2;
simend := true;
wait;
end process;
end behavioral_reg_tb;
```

Output(4 bit Register):

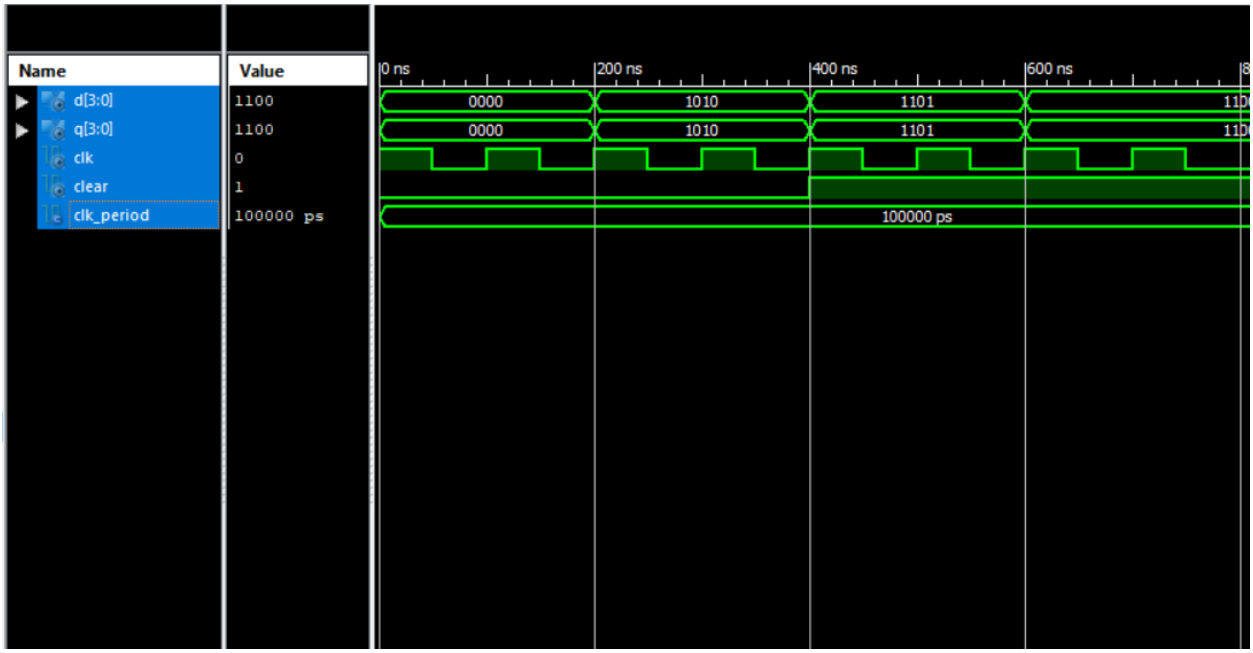

Figure : Simulation output of 4-bit Register

T Flipflop:

VHDL Module:

```
library ieee;
use ieee.std_logic_1164.all;
entity TFF is
port( din: in std_logic;
clk: in std_logic;
rst: in std_logic;
dout: out std_logic);
end TFF;
architecture behavioral of TFF is
begin
process(rst,clk,din)
begin
if (rst='1') then
dout<='0';
elsif(rising_edge(clk)) then
dout<=not din;
end if;
end process;
end behavioral;
```

VHDL Testbench:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY TFF_tb IS
END TFF_tb;
COMPONENT TFF
PORT(
din : IN std_logic;
clk : IN std_logic;
rst : IN std_logic;
dout : OUT std_logic
);
END COMPONENT;
signal din : std_logic := '0';
signal clk : std_logic := '0';
signal rst : std_logic := '0';
signal dout : std_logic;
constant clk_period : time := 10 ns;
BEGIN
uut: TFF PORT MAP (
din => din,
clk => clk,
```

```
rst => rst,
dout => dout
);
clk_process :process
begin
clk <= '0';
wait for clk_period/2;
clk <= '1';
wait for clk_period/2;
end process;
stim_proc: process
begin
rst <= '1';
wait for 50 ns;
rst <= '0';
din <= '0';
wait for 50 ns;
rst <= '0';
din <= '1';
wait;
end process;
END;
```

Output(T flipflop):

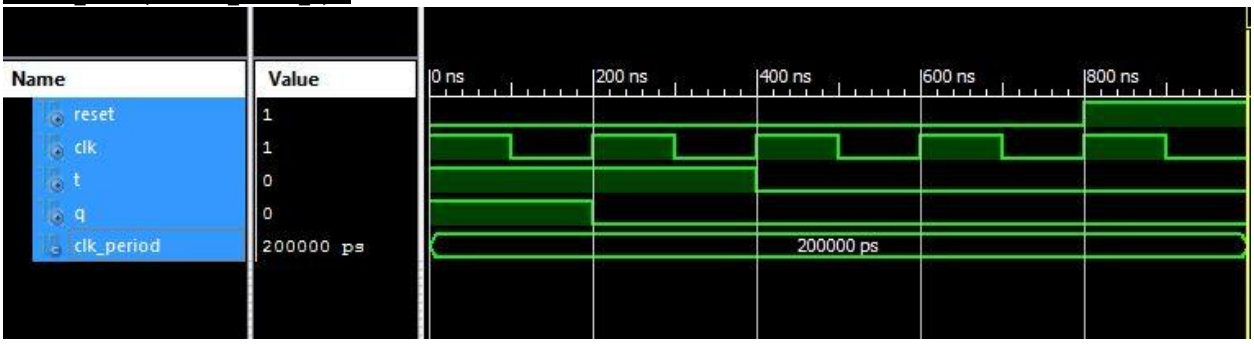

Figure : Simulation Output of T flipflop

4 bit Shift Register:

VHDL Module:

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

entity Dflipflop is
Port ( clk : in  STD_LOGIC;
reset : in  STD_LOGIC;
D : in  STD_LOGIC;
Q : out  STD_LOGIC);
end Dflipflop;

architecture struc of Dflipflop is
signal qtemp: std_logic:='0';
begin
Q<=qtemp;

process(clk,reset)
begin

if(reset ='1')then
qtemp<='0';

elsif(rising_edge(clk))then
if(D ='0')then

qtemp<='0';
else
qtemp<= '1';
end if;

else
qtemp<=qtemp;
end if;
end process;
end struc;
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity shiftregister is
Port ( clk : in  STD_LOGIC;
```

```
reset : in  STD_LOGIC;
input: in std_logic;
Q : out  STD_LOGIC_VECTOR(3 downto 0));
end shiftregister;

architecture Behavioral of shiftregister is

component Dflipflop
Port ( clk : in  STD_LOGIC;
reset : in  STD_LOGIC;
D : in  STD_LOGIC;
Q : out  STD_LOGIC);
end component;

signal C : std_logic_vector(2 downto 0);
begin
proc1:Dflipflop
port map(clk,reset,input,C(0));
Q(0)<= C(0);
proc2: Dflipflop
port map(clk,reset,C(0), C(1));
Q(1)<= C(1);

proc3:Dflipflop
port map(clk,reset,C(1),C(2));
Q(2)<= C(2);

proc4:Dflipflop
port map(clk,reset,C(2),Q(3));

end Behavioral;
```

VHDL Testbench:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY shiftregister_tb IS
END shiftregister_tb;

ARCHITECTURE behavior OF shiftregister_tb IS
COMPONENT shiftregister
PORT(
```

```
clk : IN  std_logic;
reset : IN  std_logic;
input : IN  std_logic;
Q : OUT  std_logic_vector(3 downto 0)
);
END COMPONENT;

signal clk : std_logic := '0';
signal reset : std_logic := '0';
signal input : std_logic := '0';

signal Q : std_logic_vector(3 downto 0);

constant clk_period : time := 100 ns;
shared variable simend: boolean:= false;

BEGIN

uut: shiftregister PORT MAP (
clk => clk,
reset => reset,
input => input,
Q => Q
);

clk_process :process
begin
while simend= false loop
clk<=not clk;
wait for clk_period/2;
end loop;
wait;
end process;

stim_proc: process
begin

input <= '0';
wait for clk_period;
input<='1';
wait for clk_period;
input <= '0';
wait for clk_period;
```

```
input <= '1';
wait for clk_period;
input <= '1';
wait for clk_period;
input <= '0';
wait for clk_period;
simend :=true;


wait;
end process;

END;
```

Output(4 bit Shift Register):

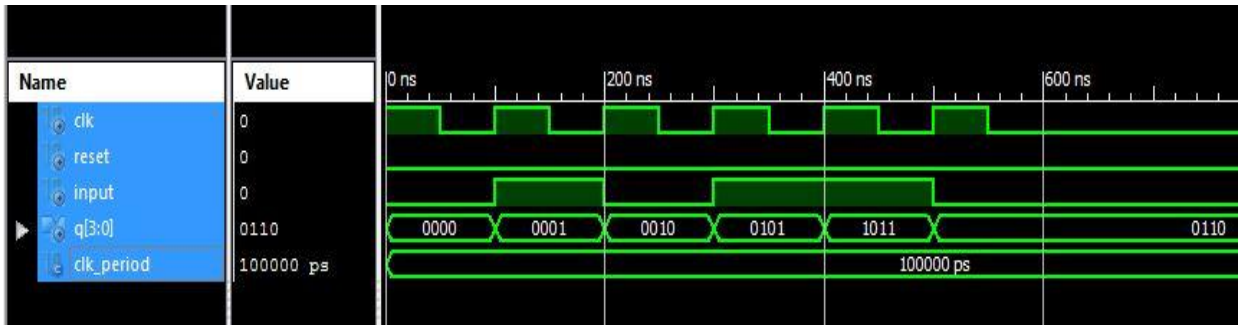

Figure : Simulation output of 4-bit Shift Register

5. 4-Bit Counter :

VHDL module :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity UPDOWN_COUNTER is
Port ( clk: in std_logic; -- clock input
reset: in std_logic; -- reset input
up_down: in std_logic; -- up or down
counter: out std_logic_vector(3 downto 0) -- output 4-bit counter
);
end UPDOWN_COUNTER;
architecture Behavioral of UPDOWN_COUNTER is
signal counter_updown: std_logic_vector(3 downto 0);
begin
process(clk,reset)
begin
if(rising_edge(clk)) then
if(reset='1') then
counter_updown <= x"0";
elsif(up_down='1') then
counter_updown <= counter_updown - x"1"; -- count down
else
counter_updown <= counter_updown + x"1"; -- count up
end if;
end if;
end process;
counter <= counter_updown;
end Behavioral;
```

VHDLTestbench :

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity tb_counters is
end tb_counters;
architecture Behavioral of tb_counters is
component UPDOWN_COUNTER
Port ( clk: in std_logic; -- clock input
reset: in std_logic; -- reset input
up_down: in std_logic;
counter: out std_logic_vector(3 downto 0) -- output 4-bit counter
);
end component;
signal reset,clk,up_down: std_logic;
signal counter:std_logic_vector(3 downto 0);
```

```
begin
dut: UPDOWN_COUNTER port map (clk => clk, reset=>reset, up_down => up_down, counter
=> counter);
clock_process :process
begin
clk <= '0';
wait for 10 ns;
clk <= '1';
wait for 10 ns;
end process;
stim_proc: process
begin
reset <= '1';
up_down <= '0';
wait for 20 ns;
reset <= '0';
wait for 300 ns;
up_down <= '1';
wait;
end process;
end Behavioral;
```

Output(4 bit counter):

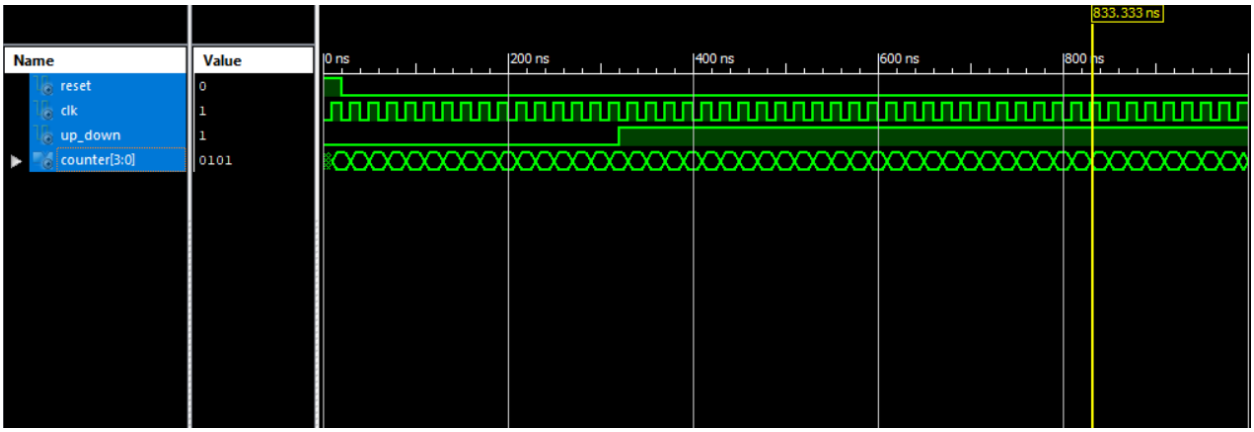

Figure : Simulation output of 4-bit counter