**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)**

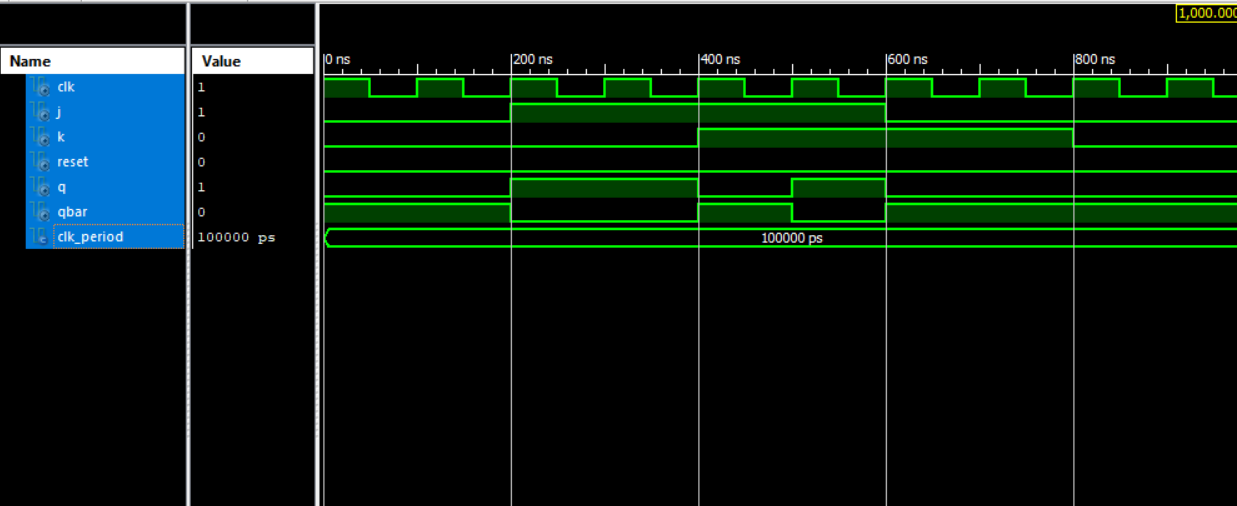


Figure : Simulation output of JK flipflop

**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):**

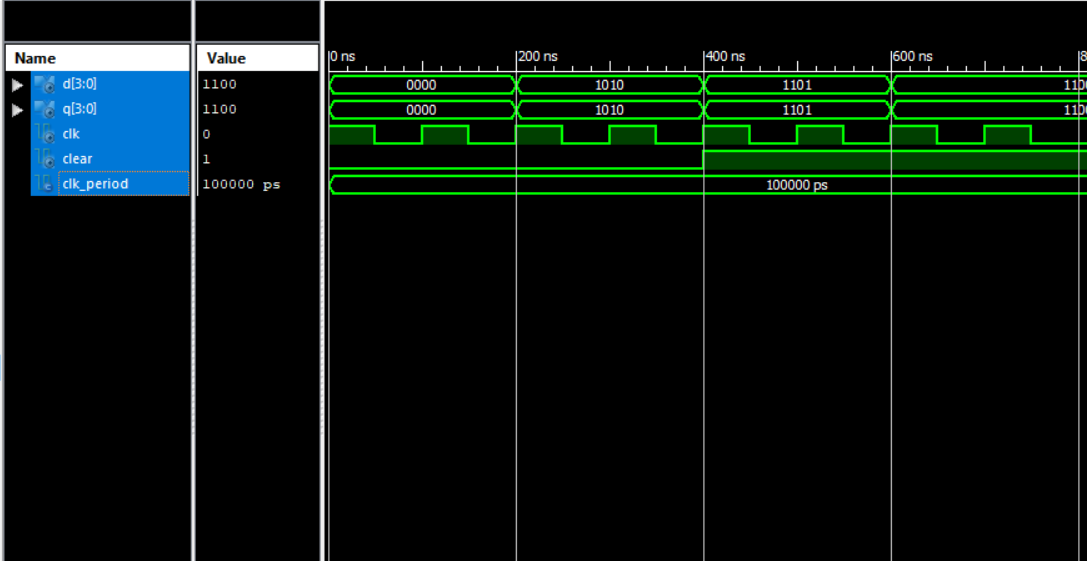


Figure : Simulation output of 4-bit Register

**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):**

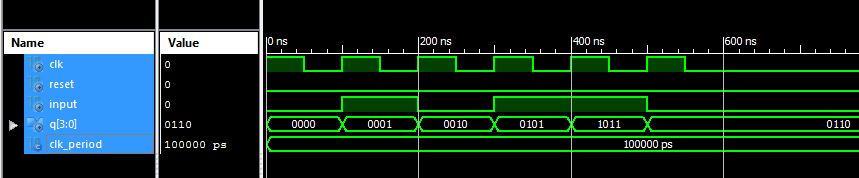
****

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):**

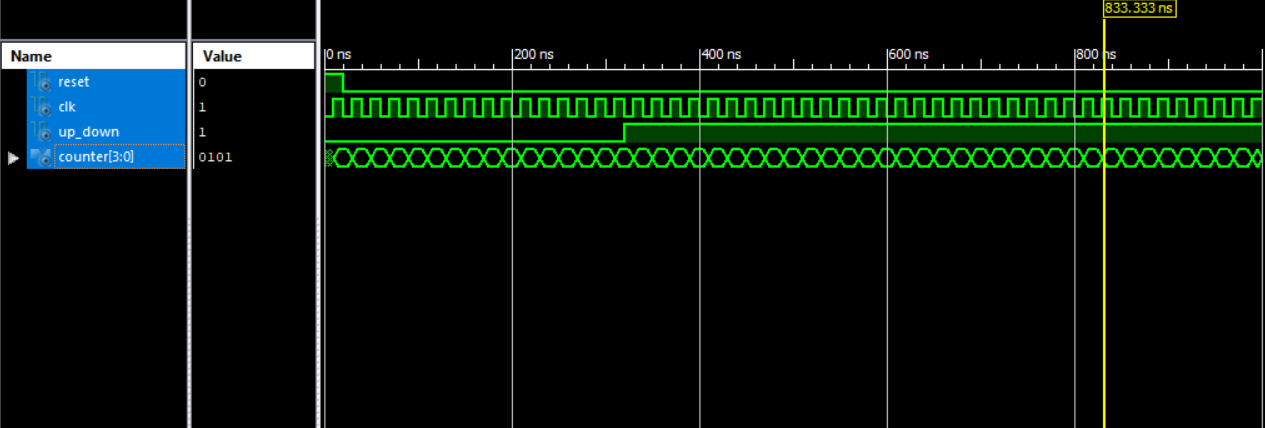


Figure : Simulation output of 4-bit counter