1. **JK Flip-flop :**

VHDL module :

```
--libraries to be used are specified here
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
 qtemp <= '0';
 qbartemp <= '1';
elsif( rising_edge(clk) ) then
if(J='0' and K='0') then
 NULL;
elsif(J='0' and K='1') then
 qtemp <= '0';
 qbartemp <= '1';
elsif(J='1' and K='0') then
 qtemp <= '1';
 qbartemp <= '0';
else
qtemp <= not qtemp;
qbartemp <= not qbartemp;
end if;
end if;
end process;
end Behavioral;
```

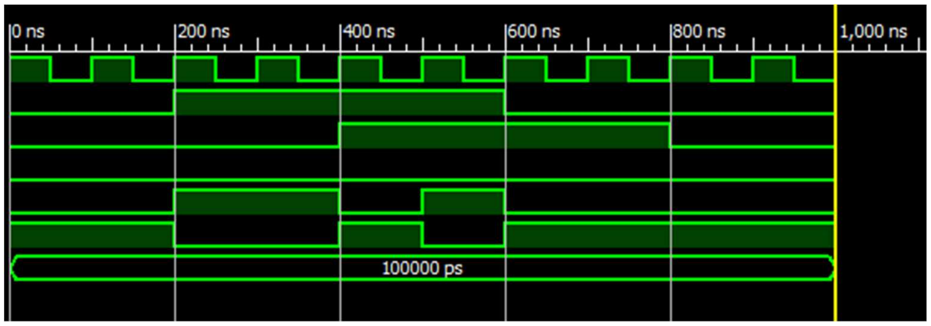
Testbench :

```
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;
-- Stimulus 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;
```

WAVEFROM :



2. T Flip-flop :

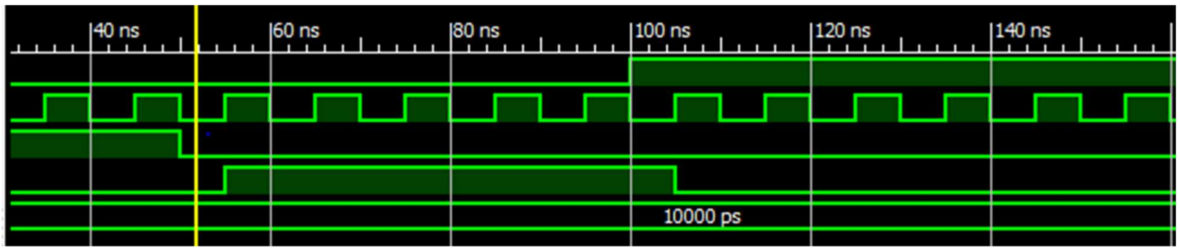
VHDL module :

```
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;
```

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;
```

WAVEFROM :



3. D-Flip-flop :

<u>VHDL module :</u>

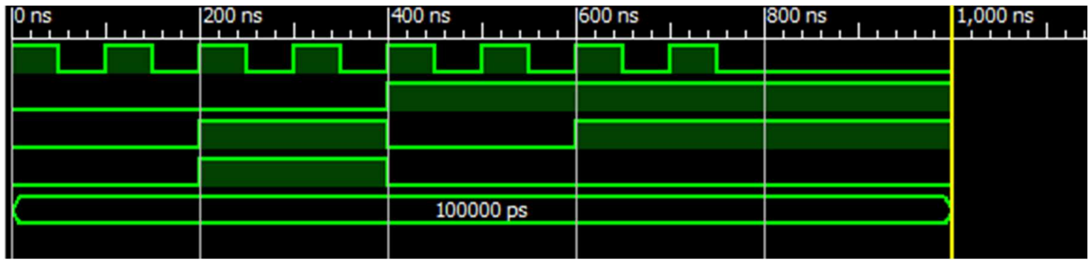
```
library ieee;
use ieee.std_logic_1164.all;
entity dff is
            port(CLK, RESET, D : in std_logic;
                        Q : out std_logic);
            end dff;
architecture pet_pr of dff is

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;
```

<u>Testbench :</u>

```
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';
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;
```

WAVEFROM :



4. 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
            );
```

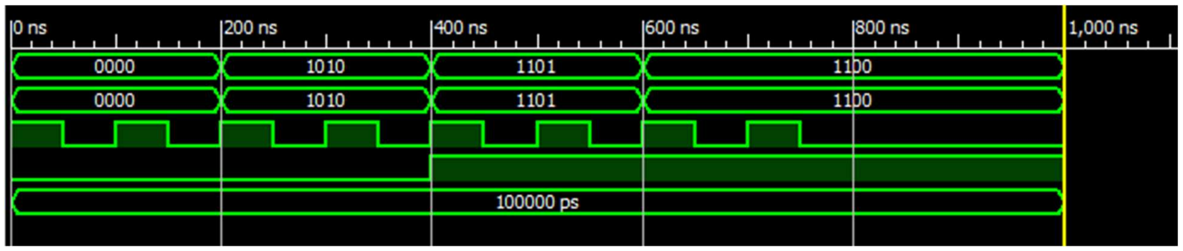
```
end dff;

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;
```

<u>Testbench:</u>

```
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;
```

WAVEFROM :



5. 4-Bit Counter :

VHDL module :

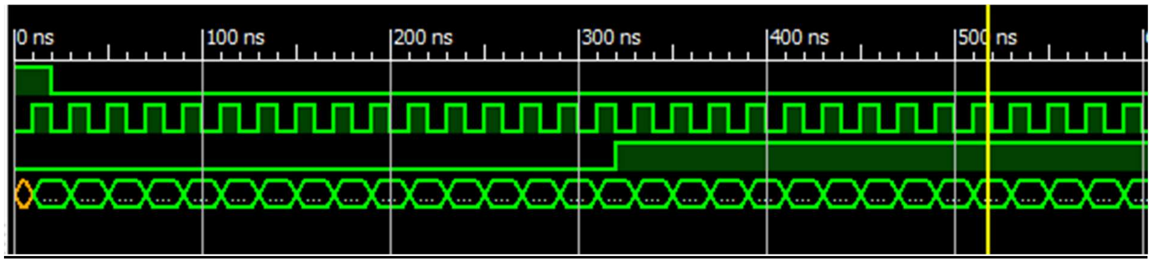
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity UPDOWN_COUNTER is
    Port ( clk: in std_logic; -- clock input
           reset: in std_logic; -- reset input
     up_down: in std_logic; -- up or down
           counter: out std_logic_vector(3 downto 0) -- output 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;
```

Testbench :

```
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;
```

Wavefrom :



6. 4-Bit Shift Register :

VHDL module :

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

entity shiftreg_4bit is
    Port ( Din : in  STD_LOGIC;
           CLK : in  STD_LOGIC;
```

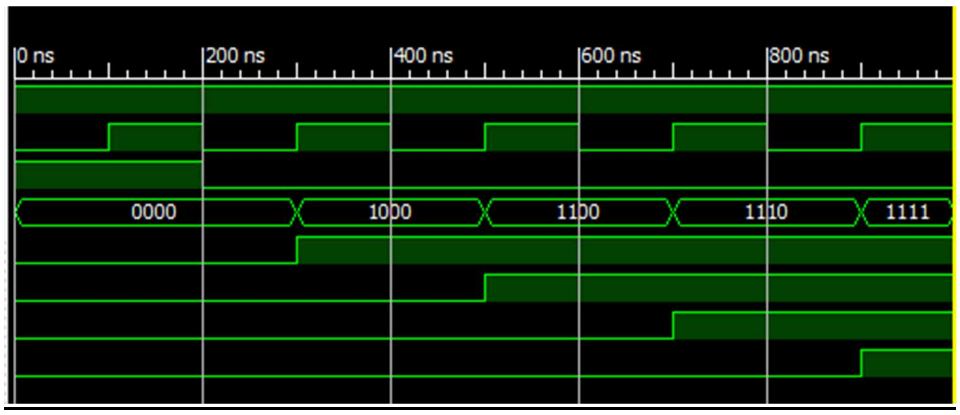
```
          RESET : in  STD_LOGIC;
          Q : out  STD_LOGIC_VECTOR (3 downto 0));
end shiftreg_4bit;

architecture Structural of shiftreg_4bit is component DF is
              Port ( D : in  STD_LOGIC;
                                      CLK : in  STD_LOGIC;
          RESET : in  STD_LOGIC;
          Q : out  STD_LOGIC);
              end component;
signal Qtemp : STD_LOGIC_VECTOR (3 downto 1);
begin
DF3 : DF port map (Din, CLK, RESET, Qtemp(3));
DF2 : DF port map (Qtemp(3), CLK, RESET, Qtemp(2));
DF1 : DF port map (Qtemp(2), CLK, RESET, Qtemp(1));
DF0 : DF port map (Qtemp(1), CLK, RESET, Q(0));

Q(3 downto 1) <= Qtemp;
end Structural;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity DF is
    Port ( D : in STD_LOGIC;
                                      CLK : in STD_LOGIC;
                                      RESET: in STD_LOGIC;
                                      Q : out STD_LOGIC);
end DF;
architecture Behavioral of DF is

begin
             process (CLK, RESET)
             begin
                          if(RESET = '1') then
                                       Q <= '0';
                          elsif(rising_edge(CLK)) then
                                       Q <= D;
                          end if;
             end process;
end Behavioral;
```

Test bench :

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY tb_shiftreg_4bit IS
END tb_shiftreg_4bit;

ARCHITECTURE behavior OF tb_shiftreg_4bit IS

    COMPONENT shiftreg_4bit
```

```
 PORT(
      Din : IN  std_logic;
      CLK : IN  std_logic;
      RESET : IN  std_logic;
      Q : OUT  std_logic_vector(3 downto 0)
     );
 END COMPONENT;
signal Din : std_logic := '0';
signal CLK : std_logic := '0';
signal RESET : std_logic := '0';
signal Q : std_logic_vector(3 downto 0);
BEGIN
uut: shiftreg_4bit PORT MAP (
       Din => Din,
       CLK => CLK,
       RESET => RESET,
       Q => Q
     );
CLK_process :process
begin
               CLK <= '0';
               wait for 100 ns;
               CLK <= '1';
               wait for 100 ns;
end process;
stim_proc: process
begin

               RESET <= '1';
               Din <= '1';
               wait for 200 ns;
               Reset <= '0';
end process;
END
```

Wavefrom :


0 ns
200 ns
400 ns
600 ns
800 ns