1. Реализовать синхронный JK триггер с инверсным асинхронным входом сброса. Реализовать 8-ми разрядный параллельный регистр с инверсным асинхронным входом сброса (используя JK триггер).

```
-- JK триггер:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity JKTrigger is
    Port (
        CLK: IN STD_LOGIC;
        J: IN STD_LOGIC;
        K: IN STD_LOGIC;
        R: IN STD_LOGIC;
        Q: OUT STD_LOGIC;
        nQ: OUT STD_LOGIC
    );
end JKTrigger;

architecture Behavioral of JKTrigger is

signal temp: STD_LOGIC := '0';

begin

    process (CLK, R)
    begin
        if (R = '1') then
            temp <= '0';
        end if;

        if (rising_edge(CLK)) then
            if (J = '0' and K = '0') then
                temp <= temp;
            elsif (J = '1' and K = '0') then
                temp <= '1';
            elsif (J = '0' and K = '1') then
                temp <= '0';
            elsif (J = '1' and K = '1') then
                temp <= not temp;
            end if;
        end if;
    end process;

    Q <= temp;
    nQ <= not temp;

end Behavioral;
```

-- Параллельный регистр с инверсным асинхронным входом сброса:

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

entity ParallelRegister is
    Port (
        CLK: IN STD_LOGIC;
        CLR: IN STD_LOGIC;
        D: IN STD_LOGIC_VECTOR (7 downto 0);
        Q: OUT STD_LOGIC_VECTOR (7 downto 0)
    );
end ParallelRegister;

architecture Behavioral of ParallelRegister is

component JKTrigger port (
    CLK: IN STD_LOGIC;
    J: IN STD_LOGIC;
    K: IN STD_LOGIC;
    R: IN STD_LOGIC;
    Q: OUT STD_LOGIC;
    nQ: OUT STD_LOGIC
);
end component;

signal nD: STD_LOGIC_VECTOR (7 downto 0) := "00000000";
signal outQ: STD_LOGIC_VECTOR (7 downto 0);
signal outnQ: STD_LOGIC_VECTOR (7 downto 0);

begin

    nD <= not D;
    JKTrigger0: JKTrigger port map(CLK, D(0), nD(0), CLR, outQ(0), outnQ(0));
    JKTrigger1: JKTrigger port map(CLK, D(1), nD(1), CLR, outQ(1), outnQ(1));
    JKTrigger2: JKTrigger port map(CLK, D(2), nD(2), CLR, outQ(2), outnQ(2));
    JKTrigger3: JKTrigger port map(CLK, D(3), nD(3), CLR, outQ(3), outnQ(3));
    JKTrigger4: JKTrigger port map(CLK, D(4), nD(4), CLR, outQ(4), outnQ(4));
    JKTrigger5: JKTrigger port map(CLK, D(5), nD(5), CLR, outQ(5), outnQ(5));
    JKTrigger6: JKTrigger port map(CLK, D(6), nD(6), CLR, outQ(6), outnQ(6));
    JKTrigger7: JKTrigger port map(CLK, D(7), nD(7), CLR, outQ(7), outnQ(7));

    Q <= outQ;

end Behavioral;
```

```
-- ТестБенч:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity TestParallelRegister is
--  Port ( );
end TestParallelRegister;

architecture Behavioral of TestParallelRegister is

component ParallelRegister port (
    CLK: IN STD_LOGIC;
    CLR: IN STD_LOGIC;
    D: IN STD_LOGIC_VECTOR (7 downto 0);
    Q: OUT STD_LOGIC_VECTOR (7 downto 0)
);
end component;

constant period: time := 10 ns;
signal CLK: STD_LOGIC := '0';
signal CLR: STD_LOGIC := '0';
signal D: STD_LOGIC_VECTOR (7 downto 0);
signal Q: STD_LOGIC_VECTOR (7 downto 0);

begin
    test: ParallelRegister port map (
        CLK => CLK,
        CLR => CLR,
        D => D,
        Q => Q
    );

    process begin
        D <= "00000000";
        CLR <= '1';
        wait for period;
        D <= "11111111";
        CLR <= '1';
        wait for period;
        CLR <= '0';
        wait for period;
        D <= "10010100";
        CLR <= '1';
        wait for period;
        wait;
    end process;

    process begin
        wait for period / 2;
        CLK <= not CLK;
    end process;

end Behavioral;
```

2. Реализовать синхронный D триггер с инверсным асинхронным входом сброса. Реализовать 8-ми разрядный параллельный регистр с инверсным асинхронным входом сброса (используя D триггер).

```
-- D триггер:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity DTrigger is
    Port (
        CLK: in STD_LOGIC;
        R: in STD_LOGIC;
        D: in STD_LOGIC;
        Q: out STD_LOGIC;
        nQ: out STD_LOGIC
    );
end DTrigger;

architecture Behavioral of DTrigger is

signal temp: STD_LOGIC := '0';

begin

    process (CLK, R) begin
        if (R = '0') then
            temp <= '0';
        elsif (CLK'event and CLK = '1') then
            temp <= D;
        end if;
    end process;

    Q <= temp;
    nQ <= not temp;

end Behavioral;
```

```
-- Параллельный регистр:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ParallelRegister is
    Port (
        CLK: IN STD_LOGIC;
        CLR: IN STD_LOGIC;
        D: IN STD_LOGIC_VECTOR (7 downto 0);
        Q: OUT STD_LOGIC_VECTOR (7 downto 0)
    );
end ParallelRegister;

architecture Behavioral of ParallelRegister is

component DTrigger port (
    CLK: IN STD_LOGIC;
    R: IN STD_LOGIC;
    D: IN STD_LOGIC;
    Q: OUT STD_LOGIC;
    nQ: OUT STD_LOGIC
);
end component;

signal nD: STD_LOGIC_VECTOR (7 downto 0);
signal outQ: STD_LOGIC_VECTOR (7 downto 0);
signal outnQ: STD_LOGIC_VECTOR (7 downto 0);

begin

    nD <= not D;
    DTrigger0: DTrigger port map(CLK, CLR, D(0), outQ(0), outnQ(0));
    DTrigger1: DTrigger port map(CLK, CLR, D(1), outQ(1), outnQ(1));
    DTrigger2: DTrigger port map(CLK, CLR, D(2), outQ(2), outnQ(2));
    DTrigger3: DTrigger port map(CLK, CLR, D(3), outQ(3), outnQ(3));
    DTrigger4: DTrigger port map(CLK, CLR, D(4), outQ(4), outnQ(4));
    DTrigger5: DTrigger port map(CLK, CLR, D(5), outQ(5), outnQ(5));
    DTrigger6: DTrigger port map(CLK, CLR, D(6), outQ(6), outnQ(6));
    DTrigger7: DTrigger port map(CLK, CLR, D(7), outQ(7), outnQ(7));
    Q <= outQ;

end Behavioral;
```

```
-- ТестБенч:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity TestParallelRegister is
--  Port ( );
end TestParallelRegister;

architecture Behavioral of TestParallelRegister is

component ParallelRegister port (
    CLK: IN STD_LOGIC;
    CLR: IN STD_LOGIC;
    D: IN STD_LOGIC_VECTOR (7 downto 0);
    Q: OUT STD_LOGIC_VECTOR (7 downto 0)
);
end component;

constant period: time := 10 ns;
signal CLK: STD_LOGIC := '0';
signal CLR: STD_LOGIC := '0';
signal D: STD_LOGIC_VECTOR (7 downto 0);
signal Q: STD_LOGIC_VECTOR (7 downto 0);

begin
    test: ParallelRegister port map (
        CLK => CLK,
        CLR => CLR,
        D => D,
        Q => Q
    );

    process begin
        D <= "00000000";
        CLR <= '1';
        wait for period;
        D <= "11111111";
        CLR <= '1';
        wait for period;
        CLR <= '0';
        wait for period;
        D <= "10010100";
        CLR <= '1';
        wait for period;
        wait;
    end process;

    process begin
        wait for period / 2;
        CLK <= not CLK;
    end process;

end Behavioral;
```

3. Реализовать синхронный JK триггер с инверсным асинхронным входом сброса. Реализовать 8-ми разрядный последовательный регистр с инверсным асинхронным входом сброса (используя JK триггер).

```
-- JK-триггер:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity JKTrigger is
    Port (
        CLK : in STD_LOGIC;
        R : in STD_LOGIC;
        J : in STD_LOGIC;
        K : in STD_LOGIC;
        Q : out STD_LOGIC;
        nQ : out STD_LOGIC
    );
end JKTrigger;

architecture Behavioral of JKTrigger is

signal temp: STD_LOGIC := '0';

begin

    process (CLK, R)
    begin
        if (R = '0') then
            temp <= '0';
        elsif (rising_edge(CLK)) then
            if (J = '0' and K = '0') then
                temp <= temp;
            elsif (J = '1' and K = '0') then
                temp <= '1';
            elsif (J = '0' and K = '1') then
                temp <= '0';
            elsif (J = '1' and K = '1') then
                temp <= not temp;
            end if;
        end if;
    end process;

    Q <= temp;
    nQ <= not temp;

end Behavioral;
```

```
-- Последовательный регистр:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity SerialRegister is
    Port (
        CLK: IN STD_LOGIC;
        CLR: IN STD_LOGIC;
        D: IN STD_LOGIC;
        Q: OUT STD_LOGIC_VECTOR (7 downto 0)
    );
end SerialRegister;

architecture Behavioral of SerialRegister is

component JKTrigger port (
    CLK : in STD_LOGIC;
    R : in STD_LOGIC;
    J : in STD_LOGIC;
    K : in STD_LOGIC;
    Q : out STD_LOGIC;
    nQ : out STD_LOGIC
);
end component;

signal nD: STD_LOGIC;
signal outQ: STD_LOGIC_VECTOR (7 downto 0);
signal outnQ: STD_LOGIC_VECTOR (7 downto 0);

begin
    nD <= not D;
    JKTrigger0: JKTrigger port map (CLK, CLR, D, nD, outQ(0), outnQ(0));
    JKTrigger1: JKTrigger port map (CLK, CLR, outQ(0), outnQ(0), outQ(1), outnQ(1));
    JKTrigger2: JKTrigger port map (CLK, CLR, outQ(1), outnQ(1), outQ(2), outnQ(2));
    JKTrigger3: JKTrigger port map (CLK, CLR, outQ(2), outnQ(2), outQ(3), outnQ(3));
    JKTrigger4: JKTrigger port map (CLK, CLR, outQ(3), outnQ(3), outQ(4), outnQ(4));
    JKTrigger5: JKTrigger port map (CLK, CLR, outQ(4), outnQ(4), outQ(5), outnQ(5));
    JKTrigger6: JKTrigger port map (CLK, CLR, outQ(5), outnQ(5), outQ(6), outnQ(6));
    JKTrigger7: JKTrigger port map (CLK, CLR, outQ(6), outnQ(6), outQ(7), outnQ(7));
    Q <= outQ;

end Behavioral;
```

```
-- ТестБенч:

entity TestSerialRegister is
--  Port ( );
end TestSerialRegister;

architecture Behavioral of TestSerialRegister is

component SerialRegister port(
    CLK: IN STD_LOGIC;
    CLR: IN STD_LOGIC;
    D: IN STD_LOGIC;
    Q: OUT STD_LOGIC_VECTOR (7 downto 0)
);
end component;

signal CLK: STD_LOGIC := '0';
constant period: time := 10 ns;
signal CLR: STD_LOGIC := '1';
signal D: STD_LOGIC := '0';
signal outQ: STD_LOGIC_VECTOR (7 downto 0);

begin
    test: SerialRegister port map(
        CLK => CLK,
        CLR => CLR,
        D => D,
        Q => outQ
    );

    process begin
        D <= '1';
        CLR <= '1';
        wait for period*8;
        D <= '0';
        CLR <= '1';
        wait for period*8;
        D <= '1';
        CLR <= '1';
        wait for period*8;
        CLR <= '0';
        wait for period;
        CLR <= '1';
        D <= '1';
        wait for period;
        D <= '0';
        wait for period;
        D <= '0';
        wait for period;
        D <= '1';
        wait for period;
        D <= '0';
        wait for period;
        D <= '1';
        wait for period;
        wait;
    end process;

    process begin
        wait for period;
        CLK <= not CLK;
    end process;
end Behavioral;
```

4. Реализовать синхронный D триггер с инверсным асинхронным входом сброса. Реализовать 8-ми разрядный последовательный регистр с инверсным асинхронным входом сброса (используя D триггер).

```
-- D-триггер:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity DTrigger is
    Port (
        CLK: in STD_LOGIC;
        R: in STD_LOGIC;
        D: in STD_LOGIC;
        Q: out STD_LOGIC;
        nQ: out STD_LOGIC
    );
end DTrigger;

architecture Behavioral of DTrigger is

signal temp: STD_LOGIC := '0';

begin

    process (CLK, R) begin
        if (R = '0') then
            temp <= '0';
        elsif (rising_edge(CLK)) then
            temp <= D;
        end if;
    end process;

    Q <= temp;
    nQ <= not temp;

end Behavioral;
```

```
-- Последовательный регистр:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity SerialRegister is
    Port (
        CLK: IN STD_LOGIC;
        CLR: IN STD_LOGIC;
        D: IN STD_LOGIC;
        Q: OUT STD_LOGIC_VECTOR (7 downto 0)
    );
end SerialRegister;

architecture Behavioral of SerialRegister is

component DTrigger port (
    CLK : in STD_LOGIC;
    R : in STD_LOGIC;
    D : in STD_LOGIC;
    Q : out STD_LOGIC;
    nQ : out STD_LOGIC
);
end component;

signal nD: STD_LOGIC;
signal outQ: STD_LOGIC_VECTOR (7 downto 0);
signal outnQ: STD_LOGIC_VECTOR (7 downto 0);

begin
    nD <= not D;
    DTrigger0: DTrigger port map (CLK, CLR, D, outQ(0), outnQ(0));
    DTrigger1: DTrigger port map (CLK, CLR, outQ(0), outQ(1), outnQ(1));
    DTrigger2: DTrigger port map (CLK, CLR, outQ(1), outQ(2), outnQ(2));
    DTrigger3: DTrigger port map (CLK, CLR, outQ(2), outQ(3), outnQ(3));
    DTrigger4: DTrigger port map (CLK, CLR, outQ(3), outQ(4), outnQ(4));
    DTrigger5: DTrigger port map (CLK, CLR, outQ(4), outQ(5), outnQ(5));
    DTrigger6: DTrigger port map (CLK, CLR, outQ(5), outQ(6), outnQ(6));
    DTrigger7: DTrigger port map (CLK, CLR, outQ(6), outQ(7), outnQ(7));
    Q <= outQ;

end Behavioral;
```

```
-- ТестБенч:

entity TestSerialRegister is
--  Port ( );
end TestSerialRegister;

architecture Behavioral of TestSerialRegister is

component SerialRegister port(
    CLK: IN STD_LOGIC;
    CLR: IN STD_LOGIC;
    D: IN STD_LOGIC;
    Q: OUT STD_LOGIC_VECTOR (7 downto 0)
);
end component;

signal CLK: STD_LOGIC := '0';
constant period: time := 10 ns;
signal CLR: STD_LOGIC := '1';
signal D: STD_LOGIC := '0';
signal outQ: STD_LOGIC_VECTOR (7 downto 0);

begin
    test: SerialRegister port map(
        CLK => CLK,
        CLR => CLR,
        D => D,
        Q => outQ
    );

    process begin
        D <= '1';
        CLR <= '1';
        wait for period*8;
        D <= '0';
        CLR <= '1';
        wait for period*8;
        D <= '1';
        CLR <= '1';
        wait for period*8;
        CLR <= '0';
        wait for period;
        CLR <= '1';
        D <= '1';
        wait for period;
        D <= '0';
        wait for period;
        D <= '0';
        wait for period;
        D <= '1';
        wait for period;
        D <= '0';
        wait for period;
        D <= '1';
        wait for period;
        wait;
    end process;

    process begin
        wait for period;
        CLK <= not CLK;
    end process;
end Behavioral;
```

5. Реализовать синхронный JK триггер с инверсным асинхронным входом сброса. Реализовать 8-ми разрядный вычитающий счетчик с инверсным асинхронным входом сброса (используя JK триггер).

```
-- JK-триггер:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity JKTrigger is
    Port (
        CLK : in STD_LOGIC;
        R : in STD_LOGIC;
        J : in STD_LOGIC;
        K : in STD_LOGIC;
        Q : out STD_LOGIC;
        nQ : out STD_LOGIC
    );
end JKTrigger;

architecture Behavioral of JKTrigger is

signal temp: STD_LOGIC := '0';

begin

    process (CLK, R)
    begin
        if (R = '0') then
            temp <= '0';
        elsif (rising_edge(CLK)) then
            if (J = '0' and K = '0') then
                temp <= temp;
            elsif (J = '1' and K = '0') then
                temp <= '1';
            elsif (J = '0' and K = '1') then
                temp <= '0';
            elsif (J = '1' and K = '1') then
                temp <= not temp;
            end if;
        end if;
    end process;

    Q <= temp;
    nQ <= not temp;

end Behavioral;
```

```
-- Вычитающий счетчик:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Counter is
    Port (
        CLK: IN STD_LOGIC;
        CLR: IN STD_LOGIC;
        Q: OUT STD_LOGIC_VECTOR (7 downto 0)
    );
end Counter;

architecture Behavioral of Counter is

component JKTrigger port (
    CLK : in STD_LOGIC;
    R : in STD_LOGIC;
    J : in STD_LOGIC;
    K : in STD_LOGIC;
    Q : out STD_LOGIC;
    nQ : out STD_LOGIC
);
end component;

signal outQ: STD_LOGIC_VECTOR(7 downto 0);
signal outnQ: STD_LOGIC_VECTOR(7 downto 0);

begin
    JKTrigger0: JKTrigger port map(CLK, CLR, '1', '1', outQ(0), outnQ(0));
    JKTrigger1: JKTrigger port map(outQ(0), CLR, '1', '1', outQ(1), outnQ(1));
    JKTrigger2: JKTrigger port map(outQ(1), CLR, '1', '1', outQ(2), outnQ(2));
    JKTrigger3: JKTrigger port map(outQ(2), CLR, '1', '1', outQ(3), outnQ(3));
    JKTrigger4: JKTrigger port map(outQ(3), CLR, '1', '1', outQ(4), outnQ(4));
    JKTrigger5: JKTrigger port map(outQ(4), CLR, '1', '1', outQ(5), outnQ(5));
    JKTrigger6: JKTrigger port map(outQ(5), CLR, '1', '1', outQ(6), outnQ(6));
    JKTrigger7: JKTrigger port map(outQ(6), CLR, '1', '1', outQ(7), outnQ(7));
    Q <= outQ;

end Behavioral;
```

```
-- ТестБенч:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity TestCounter is
--  Port ( );
end TestCounter;

architecture Behavioral of TestCounter is

component Counter port (
    CLK: IN STD_LOGIC;
    CLR: IN STD_LOGIC;
    Q: OUT STD_LOGIC_VECTOR (7 downto 0)
);
end component;

signal CLK: STD_LOGIC := '0';
constant period: time := 10 ns;
signal CLR: STD_LOGIC := '1';
signal Q: STD_LOGIC_VECTOR (7 downto 0);

begin
    test: Counter port map(CLK, CLR, Q);
    process begin
        wait for period*2;
        CLR <= '0';
        wait for period;
        CLR <= '1';
        wait for period*16;
        wait;
    end process;

    process begin
        wait for period;
        CLK <= not CLK;
    end process;

end Behavioral;
```

6. Реализовать синхронный D триггер с инверсным асинхронным входом сброса. Реализовать 8-ми разрядный вычитающий счетчик с инверсным асинхронным входом сброса (используя D триггер).

```
-- D-триггер:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity DTrigger is
    Port (
        CLK: in STD_LOGIC;
        R: in STD_LOGIC;
        D: in STD_LOGIC;
        Q: out STD_LOGIC;
        nQ: out STD_LOGIC
    );
end DTrigger;

architecture Behavioral of DTrigger is

signal temp: STD_LOGIC := '0';

begin

    process (CLK, R) begin
        if (R = '0') then
            temp <= '0';
        elsif (rising_edge(CLK)) then
            temp <= D;
        end if;
    end process;

    Q <= temp;
    nQ <= not temp;

end Behavioral;
```

```
-- Вычитающий счетчик:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Counter is
    Port (
        CLK: IN STD_LOGIC;
        CLR: IN STD_LOGIC;
        Q: OUT STD_LOGIC_VECTOR (7 downto 0)
    );
end Counter;

architecture Behavioral of Counter is

component DTrigger port (
    CLK: in STD_LOGIC;
    R: in STD_LOGIC;
    D: in STD_LOGIC;
    Q: out STD_LOGIC;
    nQ: out STD_LOGIC
);
end component;

signal outQ: STD_LOGIC_VECTOR(7 downto 0);
signal outnQ: STD_LOGIC_VECTOR(7 downto 0);

begin
    DTrigger0: DTrigger port map(CLK, CLR, outnQ(0), outQ(0), outnQ(0));
    DTrigger1: DTrigger port map(outQ(0), CLR, outnQ(1), outQ(1), outnQ(1));
    DTrigger2: DTrigger port map(outQ(1), CLR, outnQ(2), outQ(2), outnQ(2));
    DTrigger3: DTrigger port map(outQ(2), CLR, outnQ(3), outQ(3), outnQ(3));
    DTrigger4: DTrigger port map(outQ(3), CLR, outnQ(4), outQ(4), outnQ(4));
    DTrigger5: DTrigger port map(outQ(4), CLR, outnQ(5), outQ(5), outnQ(5));
    DTrigger6: DTrigger port map(outQ(5), CLR, outnQ(6), outQ(6), outnQ(6));
    DTrigger7: DTrigger port map(outQ(6), CLR, outnQ(7), outQ(7), outnQ(7));
    Q <= outQ;

end Behavioral;
```

```
-- ТестБенч:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity TestCounter is
--  Port ( );
end TestCounter;

architecture Behavioral of TestCounter is

component Counter port (
    CLK: IN STD_LOGIC;
    CLR: IN STD_LOGIC;
    Q: OUT STD_LOGIC_VECTOR (7 downto 0)
);
end component;

signal CLK: STD_LOGIC := '0';
constant period: time := 10 ns;
signal CLR: STD_LOGIC := '1';
signal Q: STD_LOGIC_VECTOR (7 downto 0);

begin
    test: Counter port map(CLK, CLR, Q);
    process begin
        wait for period*2;
        CLR <= '0';
        wait for period;
        CLR <= '1';
        wait for period*16;
        wait;
    end process;

    process begin
        wait for period;
        CLK <= not CLK;
    end process;

end Behavioral;
```

7. Реализовать синхронный JK триггер с инверсным асинхронным входом сброса. Реализовать 8-ми разрядный суммирующий счетчик с инверсным асинхронным входом сброса (используя JK триггер).

```
-- JK-триггер:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity JKTrigger is
    Port (
        CLK : in STD_LOGIC;
        R : in STD_LOGIC;
        J : in STD_LOGIC;
        K : in STD_LOGIC;
        Q : out STD_LOGIC;
        nQ : out STD_LOGIC
    );
end JKTrigger;

architecture Behavioral of JKTrigger is

signal temp: STD_LOGIC := '0';

begin

    process (CLK, R)
    begin
        if (R = '0') then
            temp <= '0';
        elsif (rising_edge(CLK)) then
            if (J = '0' and K = '0') then
                temp <= temp;
            elsif (J = '1' and K = '0') then
                temp <= '1';
            elsif (J = '0' and K = '1') then
                temp <= '0';
            elsif (J = '1' and K = '1') then
                temp <= not temp;
            end if;
        end if;
    end process;

    Q <= temp;
    nQ <= not temp;

end Behavioral;
```

```
-- Суммирующий счетчик:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Counter is
    Port (
        CLK: IN STD_LOGIC;
        CLR: IN STD_LOGIC;
        Q: OUT STD_LOGIC_VECTOR (7 downto 0)
    );
end Counter;

architecture Behavioral of Counter is

component JKTrigger port (
    CLK : in STD_LOGIC;
    R : in STD_LOGIC;
    J : in STD_LOGIC;
    K : in STD_LOGIC;
    Q : out STD_LOGIC;
    nQ : out STD_LOGIC
);
end component;

signal outQ: STD_LOGIC_VECTOR(7 downto 0);
signal outnQ: STD_LOGIC_VECTOR(7 downto 0);

begin
    JKTrigger0: JKTrigger port map(CLK, CLR, '1', '1', outQ(0), outnQ(0));
    JKTrigger1: JKTrigger port map(outnQ(0), CLR, '1', '1', outQ(1),
outnQ(1));
    JKTrigger2: JKTrigger port map(outnQ(1), CLR, '1', '1', outQ(2),
outnQ(2));
    JKTrigger3: JKTrigger port map(outnQ(2), CLR, '1', '1', outQ(3),
outnQ(3));
    JKTrigger4: JKTrigger port map(outnQ(3), CLR, '1', '1', outQ(4),
outnQ(4));
    JKTrigger5: JKTrigger port map(outnQ(4), CLR, '1', '1', outQ(5),
outnQ(5));
    JKTrigger6: JKTrigger port map(outnQ(5), CLR, '1', '1', outQ(6),
outnQ(6));
    JKTrigger7: JKTrigger port map(outnQ(6), CLR, '1', '1', outQ(7),
outnQ(7));
    Q <= outQ;

end Behavioral;
```

```
-- ТестБенч:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity TestCounter is
--  Port ( );
end TestCounter;

architecture Behavioral of TestCounter is

component Counter port (
    CLK: IN STD_LOGIC;
    CLR: IN STD_LOGIC;
    Q: OUT STD_LOGIC_VECTOR (7 downto 0)
);
end component;

signal CLK: STD_LOGIC := '0';
constant period: time := 10 ns;
signal CLR: STD_LOGIC := '1';
signal Q: STD_LOGIC_VECTOR (7 downto 0);

begin
    test: Counter port map(CLK, CLR, Q);
    process begin
        wait for period*2;
        CLR <= '0';
        wait for period;
        CLR <= '1';
        wait for period*16;
        wait;
    end process;

    process begin
        wait for period;
        CLK <= not CLK;
    end process;

end Behavioral;
```

8. Реализовать синхронный D триггер с инверсным асинхронным входом сброса. Реализовать 8-ми разрядный суммирующий счетчик с инверсным асинхронным входом сброса (используя D триггер).

```
-- D-триггер:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity DTrigger is
    Port (
        CLK: in STD_LOGIC;
        R: in STD_LOGIC;
        D: in STD_LOGIC;
        Q: out STD_LOGIC;
        nQ: out STD_LOGIC
    );
end DTrigger;

architecture Behavioral of DTrigger is

signal temp: STD_LOGIC := '0';

begin
    process (CLK, R) begin
        if (R = '0') then
            temp <= '0';
        elsif (rising_edge(CLK)) then
            temp <= D;
        end if;
    end process;

    Q <= temp;
    nQ <= not temp;

end Behavioral;
```

```
-- Суммирующий счетчик:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Counter is
    Port (
        CLK: in STD_LOGIC;
        CLR: in STD_LOGIC;
        Q: out STD_LOGIC_VECTOR (7 downto 0)
    );
end Counter;

architecture Behavioral of Counter is

component DTrigger port (
    CLK: in STD_LOGIC;
    R: in STD_LOGIC;
    D: in STD_LOGIC;
    Q: out STD_LOGIC;
    nQ: out STD_LOGIC
);
end component;

signal outQ: STD_LOGIC_VECTOR(7 downto 0);
signal outnQ: STD_LOGIC_VECTOR(7 downto 0);

begin
    DTrigger0: DTrigger port map (CLK, CLR, outnQ(0), outQ(0), outnQ(0));
    DTrigger1: DTrigger port map (outnQ(0), CLR, outnQ(1), outQ(1),
outnQ(1));
    DTrigger2: DTrigger port map (outnQ(1), CLR, outnQ(2), outQ(2),
outnQ(2));
    DTrigger3: DTrigger port map (outnQ(2), CLR, outnQ(3), outQ(3),
outnQ(3));
    DTrigger4: DTrigger port map (outnQ(3), CLR, outnQ(4), outQ(4),
outnQ(4));
    DTrigger5: DTrigger port map (outnQ(4), CLR, outnQ(5), outQ(5),
outnQ(5));
    DTrigger6: DTrigger port map (outnQ(5), CLR, outnQ(6), outQ(6),
outnQ(6));
    DTrigger7: DTrigger port map (outnQ(6), CLR, outnQ(7), outQ(7),
outnQ(7));
    Q <= outQ;

end Behavioral;
```

```
-- ТестБенч:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity TestCounter is
--  Port ( );
end TestCounter;

architecture Behavioral of TestCounter is

component Counter port (
    CLK: IN STD_LOGIC;
    CLR: IN STD_LOGIC;
    Q: OUT STD_LOGIC_VECTOR (7 downto 0)
);
end component;

signal CLK: STD_LOGIC := '0';
constant period: time := 10 ns;
signal CLR: STD_LOGIC := '1';
signal Q: STD_LOGIC_VECTOR (7 downto 0);

begin
    test: Counter port map(CLK, CLR, Q);
    process begin
        wait for period*2;
        CLR <= '0';
        wait for period;
        CLR <= '1';
        wait for period*16;
        wait;
    end process;

    process begin
        wait for period;
        CLK <= not CLK;
    end process;

end Behavioral;
```

9. Реализовать синхронный JK триггер. Реализовать 8-ми разрядный параллельный регистр с асинхронным входом разрешения чтения OUT_EN (используя JK триггер).

```
-- JK-триггер:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity JKTrigger is
    Port (
        CLK: in STD_LOGIC;
        J: in STD_LOGIC;
        K: in STD_LOGIC;
        Q: out STD_LOGIC;
        nQ: out STD_LOGIC
    );
end JKTrigger;

architecture Behavioral of JKTrigger is

signal temp: STD_LOGIC := '0';

begin
    process (CLK) begin
        if (rising_edge(CLK)) then
            if (J = '0' and K = '0') then
                temp <= temp;
            elsif (J = '1' and K = '0') then
                temp <= '1';
            elsif (J = '0' and K = '1') then
                temp <= '0';
            elsif (J = '1' and K = '1') then
                temp <= not temp;
            end if;
        end if;
    end process;

    Q <= temp;
    nQ <= not temp;

end Behavioral;
```

```
-- Параллельный регистр:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ParallelRegister is
    Port (
        CLK: in STD_LOGIC;
        OE: in STD_LOGIC;
        D: in STD_LOGIC_VECTOR(7 downto 0);
        Q: out STD_LOGIC_VECTOR (7 downto 0)
    );
end ParallelRegister;

architecture Behavioral of ParallelRegister is

component JKTrigger port (
    CLK: in STD_LOGIC;
    J: in STD_LOGIC;
    K: in STD_LOGIC;
    Q: out STD_LOGIC;
    nQ: out STD_LOGIC
);
end component;

signal outQ: STD_LOGIC_VECTOR (7 downto 0);
signal outnQ: STD_LOGIC_VECTOR (7 downto 0);
signal nD: STD_LOGIC_VECTOR (7 downto 0);

begin
    nD <= not D;
    JKTrigger0: JKTrigger port map (CLK, D(0), nD(0), outQ(0), outnQ(0));
    JKTrigger1: JKTrigger port map (CLK, D(1), nD(1), outQ(1), outnQ(1));
    JKTrigger2: JKTrigger port map (CLK, D(2), nD(2), outQ(2), outnQ(2));
    JKTrigger3: JKTrigger port map (CLK, D(3), nD(3), outQ(3), outnQ(3));
    JKTrigger4: JKTrigger port map (CLK, D(4), nD(4), outQ(4), outnQ(4));
    JKTrigger5: JKTrigger port map (CLK, D(5), nD(5), outQ(5), outnQ(5));
    JKTrigger6: JKTrigger port map (CLK, D(6), nD(6), outQ(6), outnQ(6));
    JKTrigger7: JKTrigger port map (CLK, D(7), nD(7), outQ(7), outnQ(7));
    Q <= outQ when OE = '1' else "ZZZZZZZZ";

end Behavioral;
```

```
-- ТестБенч:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity TestParallelRegister is
--  Port ( );
end TestParallelRegister;

architecture Behavioral of TestParallelRegister is

component ParallelRegister port (
    CLK: in STD_LOGIC;
    OE: in STD_LOGIC;
    D: in STD_LOGIC_VECTOR(7 downto 0);
    Q: out STD_LOGIC_VECTOR (7 downto 0)
);
end component;

constant period: time := 10 ns;
signal CLK: STD_LOGIC := '0';
signal OE: STD_LOGIC := '1';
signal D: STD_LOGIC_VECTOR(7 downto 0);
signal Q: STD_LOGIC_VECTOR (7 downto 0);

begin
    test: ParallelRegister port map (CLK, OE, D, Q);

    process begin
        OE <= '1';
        D <= "10000000";
        wait for period * 2;
        D <= "00000001";
        wait for period * 2;
        OE <= '0';
        wait;
    end process;

    process begin
        wait for period / 2;
        CLK <= not CLK;
    end process;

end Behavioral;
```

10. Реализовать синхронный D триггер. Реализовать 8-ми разрядный параллельный регистр с асинхронным входом разрешения чтения OUT_EN (используя D триггер).

```
-- D-триггер:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity DTrigger is
    Port (
        CLK: in STD_LOGIC;
        D: in STD_LOGIC;
        Q: out STD_LOGIC;
        nQ: out STD_LOGIC
    );
end DTrigger;

architecture Behavioral of DTrigger is

signal temp: STD_LOGIC := '0';

begin
    process (CLK) begin
        if (rising_edge(CLK)) then
            temp <= D;
        end if;
    end process;

    Q <= temp;
    nQ <= not temp;

end Behavioral;
```

```
-- Параллельный регистр:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ParallelRegister is
    Port (
        CLK: in STD_LOGIC;
        OE: in STD_LOGIC;
        D: in STD_LOGIC_VECTOR(7 downto 0);
        Q: out STD_LOGIC_VECTOR (7 downto 0)
    );
end ParallelRegister;

architecture Behavioral of ParallelRegister is

component DTrigger port (
    CLK: in STD_LOGIC;
    D: in STD_LOGIC;
    Q: out STD_LOGIC;
    nQ: out STD_LOGIC
);
end component;

signal outQ: STD_LOGIC_VECTOR (7 downto 0);
signal outnQ: STD_LOGIC_VECTOR (7 downto 0);
signal nD: STD_LOGIC_VECTOR (7 downto 0);

begin
    nD <= not D;
    DTrigger0: DTrigger port map (CLK, D(0), outQ(0), outnQ(0));
    DTrigger1: DTrigger port map (CLK, D(1), outQ(1), outnQ(1));
    DTrigger2: DTrigger port map (CLK, D(2), outQ(2), outnQ(2));
    DTrigger3: DTrigger port map (CLK, D(3), outQ(3), outnQ(3));
    DTrigger4: DTrigger port map (CLK, D(4), outQ(4), outnQ(4));
    DTrigger5: DTrigger port map (CLK, D(5), outQ(5), outnQ(5));
    DTrigger6: DTrigger port map (CLK, D(6), outQ(6), outnQ(6));
    DTrigger7: DTrigger port map (CLK, D(7), outQ(7), outnQ(7));
    Q <= outQ when OE = '1' else "ZZZZZZZZ";

end Behavioral;
```

```
-- ТестБенч:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity TestParallelRegister is
--  Port ( );
end TestParallelRegister;

architecture Behavioral of TestParallelRegister is

component ParallelRegister port (
    CLK: in STD_LOGIC;
    OE: in STD_LOGIC;
    D: in STD_LOGIC_VECTOR(7 downto 0);
    Q: out STD_LOGIC_VECTOR (7 downto 0)
);
end component;

constant period: time := 10 ns;
signal CLK: STD_LOGIC := '0';
signal OE: STD_LOGIC := '1';
signal D: STD_LOGIC_VECTOR(7 downto 0);
signal Q: STD_LOGIC_VECTOR (7 downto 0);

begin
    test: ParallelRegister port map (CLK, OE, D, Q);

    process begin
        OE <= '1';
        D <= "10000000";
        wait for period * 2;
        D <= "00000001";
        wait for period * 2;
        OE <= '0';
        wait;
    end process;

    process begin
        wait for period / 2;
        CLK <= not CLK;
    end process;

end Behavioral;
```

11. Реализовать синхронный JK триггер. Реализовать 8-ми разрядный суммирующий счетчик с выходами с третьим состоянием (используя JK триггер).

```
-- JK-триггер:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity JKTrigger is
    Port (
        CLK: in STD_LOGIC;
        J: in STD_LOGIC;
        K: in STD_LOGIC;
        Q: out STD_LOGIC;
        nQ: out STD_LOGIC
    );
end JKTrigger;

architecture Behavioral of JKTrigger is

signal temp: STD_LOGIC := '0';

begin
    process (CLK) begin
        if (rising_edge(CLK)) then
            if (J = '0' and K = '0') then
                temp <= temp;
            elsif (J = '1' and K = '0') then
                temp <= '1';
            elsif (J = '0' and K = '1') then
                temp <= '0';
            elsif (J = '1' and K = '1') then
                temp <= not temp;
            end if;
        end if;
    end process;

    Q <= temp;
    nQ <= not temp;

end Behavioral;
```

```
-- Суммирующий счетчик:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Counter is
    Port (
        CLK: in STD_LOGIC;
        OE: in STD_LOGIC;
        Q: out STD_LOGIC_VECTOR (7 downto 0)
    );
end Counter;

architecture Behavioral of Counter is

component JKTrigger port (
    CLK: in STD_LOGIC;
    J: in STD_LOGIC;
    K: in STD_LOGIC;
    Q: out STD_LOGIC;
    nQ: out STD_LOGIC
);
end component;

signal outQ: STD_LOGIC_VECTOR(7 downto 0);
signal outnQ: STD_LOGIC_VECTOR(7 downto 0);

begin
    JKTrigger0: JKTrigger port map (CLK, '1', '1', outQ(0), outnQ(0));
    JKTrigger1: JKTrigger port map (outnQ(0), '1', '1', outQ(1), outnQ(1));
    JKTrigger2: JKTrigger port map (outnQ(1), '1', '1', outQ(2), outnQ(2));
    JKTrigger3: JKTrigger port map (outnQ(2), '1', '1', outQ(3), outnQ(3));
    JKTrigger4: JKTrigger port map (outnQ(3), '1', '1', outQ(4), outnQ(4));
    JKTrigger5: JKTrigger port map (outnQ(4), '1', '1', outQ(5), outnQ(5));
    JKTrigger6: JKTrigger port map (outnQ(5), '1', '1', outQ(6), outnQ(6));
    JKTrigger7: JKTrigger port map (outnQ(6), '1', '1', outQ(7), outnQ(7));
    Q <= outQ when OE = '1' else "ZZZZZZZZ";

end Behavioral;
```

```
-- ТестБенч:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity TestCounter is
--  Port ( );
end TestCounter;

architecture Behavioral of TestCounter is

component Counter port (
    CLK: IN STD_LOGIC;
    OE: IN STD_LOGIC;
    Q: OUT STD_LOGIC_VECTOR (7 downto 0)
);
end component;

signal CLK: STD_LOGIC := '0';
constant period: time := 10 ns;
signal OE: STD_LOGIC := '1';
signal Q: STD_LOGIC_VECTOR (7 downto 0);

begin
    test: Counter port map(CLK, OE, Q);
    process begin
        wait for period*2;
        OE <= '0';
        wait for period;
        OE <= '1';
        wait for period*16;
        wait;
    end process;

    process begin
        wait for period;
        CLK <= not CLK;
    end process;

end Behavioral;
```

12. Реализовать синхронный D триггер. Реализовать 8-ми разрядный суммирующий счетчик с выходами с третьим состоянием (используя D триггер).

```
-- D-триггер:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity DTrigger is
    Port (
        CLK: in STD_LOGIC;
        D: in STD_LOGIC;
        Q: out STD_LOGIC;
        nQ: out STD_LOGIC
    );
end DTrigger;

architecture Behavioral of DTrigger is

signal temp: STD_LOGIC := '0';

begin
    process (CLK) begin
        if (rising_edge(CLK)) then
            temp <= D;
        end if;
    end process;

    Q <= temp;
    nQ <= not temp;

end Behavioral;
```

```
-- Суммирующий счетчик:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Counter is
    Port (
        CLK: in STD_LOGIC;
        OE: in STD_LOGIC;
        Q: out STD_LOGIC_VECTOR (7 downto 0)
    );
end Counter;

architecture Behavioral of Counter is

component DTrigger port (
    CLK: in STD_LOGIC;
    D: in STD_LOGIC;
    Q: out STD_LOGIC;
    nQ: out STD_LOGIC
);
end component;

signal outQ: STD_LOGIC_VECTOR(7 downto 0);
signal outnQ: STD_LOGIC_VECTOR(7 downto 0);

begin
    DTrigger0: DTrigger port map (CLK, outnQ(0), outQ(0), outnQ(0));
    DTrigger1: DTrigger port map (outnQ(0), outnQ(1), outQ(1), outnQ(1));
    DTrigger2: DTrigger port map (outnQ(1), outnQ(2), outQ(2), outnQ(2));
    DTrigger3: DTrigger port map (outnQ(2), outnQ(3), outQ(3), outnQ(3));
    DTrigger4: DTrigger port map (outnQ(3), outnQ(4), outQ(4), outnQ(4));
    DTrigger5: DTrigger port map (outnQ(4), outnQ(5), outQ(5), outnQ(5));
    DTrigger6: DTrigger port map (outnQ(5), outnQ(6), outQ(6), outnQ(6));
    DTrigger7: DTrigger port map (outnQ(6), outnQ(7), outQ(7), outnQ(7));
    Q <= outQ when OE = '1' else "ZZZZZZZZ";

end Behavioral;
```

```
-- ТестБенч:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity TestCounter is
--  Port ( );
end TestCounter;

architecture Behavioral of TestCounter is

component Counter port (
    CLK: IN STD_LOGIC;
    OE: IN STD_LOGIC;
    Q: OUT STD_LOGIC_VECTOR (7 downto 0)
);
end component;

signal CLK: STD_LOGIC := '0';
constant period: time := 10 ns;
signal OE: STD_LOGIC := '1';
signal Q: STD_LOGIC_VECTOR (7 downto 0);

begin
    test: Counter port map(CLK, OE, Q);
    process begin
        wait for period*2;
        OE <= '0';
        wait for period;
        OE <= '1';
        wait for period*16;
        wait;
    end process;

    process begin
        wait for period;
        CLK <= not CLK;
    end process;

end Behavioral;
```

13. Реализовать синхронный JK триггер. Реализовать 8-ми разрядный вычитающий счетчик с выходами с третьим состоянием (используя JK триггер).

```
-- JK-триггер:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity JKTrigger is
    Port (
        CLK: in STD_LOGIC;
        J: in STD_LOGIC;
        K: in STD_LOGIC;
        Q: out STD_LOGIC;
        nQ: out STD_LOGIC
    );
end JKTrigger;

architecture Behavioral of JKTrigger is

signal temp: STD_LOGIC := '0';

begin
    process (CLK) begin
        if (rising_edge(CLK)) then
            if (J = '0' and K = '0') then
                temp <= temp;
            elsif (J = '1' and K = '0') then
                temp <= '1';
            elsif (J = '0' and K = '1') then
                temp <= '0';
            elsif (J = '1' and K = '1') then
                temp <= not temp;
            end if;
        end if;
    end process;

    Q <= temp;
    nQ <= not temp;

end Behavioral;
```

```
-- Вычитающий счетчик:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Counter is
    Port (
        CLK: in STD_LOGIC;
        OE: in STD_LOGIC;
        Q: out STD_LOGIC_VECTOR (7 downto 0)
    );
end Counter;

architecture Behavioral of Counter is

component JKTrigger port (
    CLK: in STD_LOGIC;
    J: in STD_LOGIC;
    K: in STD_LOGIC;
    Q: out STD_LOGIC;
    nQ: out STD_LOGIC
);
end component;

signal outQ: STD_LOGIC_VECTOR(7 downto 0);
signal outnQ: STD_LOGIC_VECTOR(7 downto 0);

begin
    JKTrigger0: JKTrigger port map (CLK, '1', '1', outQ(0), outnQ(0));
    JKTrigger1: JKTrigger port map (outQ(0), '1', '1', outQ(1), outnQ(1));
    JKTrigger2: JKTrigger port map (outQ(1), '1', '1', outQ(2), outnQ(2));
    JKTrigger3: JKTrigger port map (outQ(2), '1', '1', outQ(3), outnQ(3));
    JKTrigger4: JKTrigger port map (outQ(3), '1', '1', outQ(4), outnQ(4));
    JKTrigger5: JKTrigger port map (outQ(4), '1', '1', outQ(5), outnQ(5));
    JKTrigger6: JKTrigger port map (outQ(5), '1', '1', outQ(6), outnQ(6));
    JKTrigger7: JKTrigger port map (outQ(6), '1', '1', outQ(7), outnQ(7));
    Q <= outQ when OE = '1' else "ZZZZZZZZ";

end Behavioral;
```

```
-- ТестБенч:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity TestCounter is
--  Port ( );
end TestCounter;

architecture Behavioral of TestCounter is

component Counter port (
    CLK: IN STD_LOGIC;
    OE: IN STD_LOGIC;
    Q: OUT STD_LOGIC_VECTOR (7 downto 0)
);
end component;

signal CLK: STD_LOGIC := '0';
constant period: time := 10 ns;
signal OE: STD_LOGIC := '1';
signal Q: STD_LOGIC_VECTOR (7 downto 0);

begin
    test: Counter port map(CLK, OE, Q);
    process begin
        wait for period*2;
        OE <= '0';
        wait for period;
        OE <= '1';
        wait for period*16;
        wait;
    end process;

    process begin
        wait for period;
        CLK <= not CLK;
    end process;

end Behavioral;
```

14. Реализовать синхронный D триггер. Реализовать 8-ми разрядный вычитающий счетчик с выходами с третьим состоянием (используя D триггер).

```
-- D-триггер:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity DTrigger is
    Port (
        CLK: in STD_LOGIC;
        D: in STD_LOGIC;
        Q: out STD_LOGIC;
        nQ: out STD_LOGIC
    );
end DTrigger;

architecture Behavioral of DTrigger is

signal temp: STD_LOGIC := '0';

begin
    process (CLK) begin
        if (rising_edge(CLK)) then
            temp <= D;
        end if;
    end process;

    Q <= temp;
    nQ <= not temp;

end Behavioral;
```

```
-- Вычитающий счетчик:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Counter is
    Port (
        CLK: in STD_LOGIC;
        OE: in STD_LOGIC;
        Q: out STD_LOGIC_VECTOR (7 downto 0)
    );
end Counter;

architecture Behavioral of Counter is

component DTrigger port (
    CLK: in STD_LOGIC;
    D: in STD_LOGIC;
    Q: out STD_LOGIC;
    nQ: out STD_LOGIC
);
end component;

signal outQ: STD_LOGIC_VECTOR(7 downto 0);
signal outnQ: STD_LOGIC_VECTOR(7 downto 0);

begin
    DTrigger0: DTrigger port map (CLK, outnQ(0), outQ(0), outnQ(0));
    DTrigger1: DTrigger port map (outQ(0), outnQ(1), outQ(1), outnQ(1));
    DTrigger2: DTrigger port map (outQ(1), outnQ(2), outQ(2), outnQ(2));
    DTrigger3: DTrigger port map (outQ(2), outnQ(3), outQ(3), outnQ(3));
    DTrigger4: DTrigger port map (outQ(3), outnQ(4), outQ(4), outnQ(4));
    DTrigger5: DTrigger port map (outQ(4), outnQ(5), outQ(5), outnQ(5));
    DTrigger6: DTrigger port map (outQ(5), outnQ(6), outQ(6), outnQ(6));
    DTrigger7: DTrigger port map (outQ(6), outnQ(7), outQ(7), outnQ(7));
    Q <= outQ when OE = '1' else "ZZZZZZZZ";

end Behavioral;
```

```
-- ТестБенч:

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity TestCounter is
--  Port ( );
end TestCounter;

architecture Behavioral of TestCounter is

component Counter port (
    CLK: IN STD_LOGIC;
    OE: IN STD_LOGIC;
    Q: OUT STD_LOGIC_VECTOR (7 downto 0)
);
end component;

signal CLK: STD_LOGIC := '0';
constant period: time := 10 ns;
signal OE: STD_LOGIC := '1';
signal Q: STD_LOGIC_VECTOR (7 downto 0);

begin
    test: Counter port map(CLK, OE, Q);
    process begin
        wait for period*2;
        OE <= '0';
        wait for period;
        OE <= '1';
        wait for period*16;
        wait;
    end process;

    process begin
        wait for period;
        CLK <= not CLK;
    end process;

end Behavioral;
```