зад 1

```
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 (CLK'event and CLK = '1') then
if (J = '0' and K = '1') then
temp <= '0';
elsif (J = '1' and K = '0') then
temp <= '1';
elsif (J = '1' and K = '1') then
temp <= not temp;
end if;
end if;
end process;
Q <= temp;
nQ <= not temp;
end Behavioral;
```

синх. jk, инверс. асинх. вход сброса,
8р., парал. рег., инверс. асинх. вход сброса

```
entity TestParalReg is
-- Port ( );
end TestParalReg;
architecture Behavioral of TestParalReg is
component ParalReg
Port (
CLK : in STD_LOGIC;
CLEAR : 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 CLEAR : STD_LOGIC := '1';
signal D : STD_LOGIC_VECTOR (7 downto 0);
signal Q : STD_LOGIC_VECTOR (7 downto 0);
begin
test: ParalReg port map(CLK, CLEAR, D, Q);
process
begin
D <= "00000000";
CLEAR <= '1';
wait for period;
D <= "11111111";
CLEAR <= '1';
wait for period;
CLEAR <= '0';
wait for period;
D <= "10010100";
CLEAR <= '1';
wait for period;
end process;
process
begin
wait for period / 2;
CLK <= not CLK;
end process;
end Behavioral;
```

zad 2

```
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;
```

синх. d, инверс. асинх. вход сброса,
8р., парал. рег., инверс. асинх. вход сброса

```
entity TestParalReg is
-- Port ( );
end TestParalReg;
architecture Behavioral of TestParalReg is
component ParalReg
Port (
CLK : in STD_LOGIC;
CLEAR : 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 CLEAR : STD_LOGIC := '1';
signal D : STD_LOGIC_VECTOR (7 downto 0);
signal Q : STD_LOGIC_VECTOR (7 downto 0);
begin
test: ParalReg port map(CLK, CLEAR, D, Q);
process
begin
D <= "00000000";
CLEAR <= '1';
wait for period;
D <= "11111111";
CLEAR <= '1';
wait for period;
CLEAR <= '0';
wait for period;
D <= "10010100";
CLEAR <= '1';
wait for period;
end process;
process
begin
wait for period / 2;
CLK <= not CLK;
end process;
end Behavioral;
```

zad 3

```
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 (CLK'event and CLK = '1') then
if (J = '0' and K = '1') then
temp <= '0';
elsif (J = '1' and K = '0') then
temp <= '1';
elsif (J = '1' and K = '1') then
temp <= not temp;
end if;
end if;
end process;
Q <= temp;
nQ <= not temp;
end Behavioral;
```

синх. jk, инверс. асинх. вход сброса,
8р., послед. рег., инверс. асинх. вход сброса

```
entity TestPosledReg is
-- Port ( );
end TestPosledReg;
architecture Behavioral of TestPosledReg is
component PosledReg
Port ( CLK : in STD_LOGIC;
CLEAR : in STD_LOGIC;
D : in STD_LOGIC;
Q : out STD_LOGIC_VECTOR (7 downto 0) );
end component;
CONSTANT period : time := 10 ns;
signal CLK : STD_LOGIC := '0';
signal CLEAR : STD_LOGIC := '1';
signal D : STD_LOGIC;
signal Q : STD_LOGIC_VECTOR (7 downto 0);
begin test: PosledReg port map(CLK, CLEAR, D, Q);
process begin
D <= '1';
CLEAR <= '1';
wait for period*8; D <= '0'; CLEAR <= '1';
wait for period*8;
D <= '1';
CLEAR <= '1';
wait for period*8;
CLEAR <= '0';
wait for period;
CLEAR <= '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; end process; process begin
wait for period / 2;
CLK <= not CLK;
end process;
end Behavioral;
```

zad 4

```
entity TestPosledReg is
-- Port ( );
end TestPosledReg;
architecture Behavioral of TestPosledReg is
component PosledReg
Port ( CLK : in STD_LOGIC;
CLEAR : in STD_LOGIC;
D : in STD_LOGIC;
Q : out STD_LOGIC_VECTOR (7 downto 0) );
end component;
CONSTANT period : time := 10 ns;
signal CLK : STD_LOGIC := '0';
signal CLEAR : STD_LOGIC := '1';
signal D : STD_LOGIC;
signal Q : STD_LOGIC_VECTOR (7 downto 0);
begin
test: PosledReg port map(CLK, CLEAR, D, Q);
process begin
D <= '1';
CLEAR <= '1';
wait for period*8; D <= '0';
CLEAR <= '1'; wait for period*8;
D <= '1'; CLEAR <= '1';
wait for period*8; CLEAR <= '0';
wait for period; CLEAR <= '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;
end process;
process
begin
wait for period / 2;
CLK <= not CLK;
end process;
end Behavioral;
```

```
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;
```

синх. d, инверс. асинх. вход сброса,
8р., послед. рег., инверс. асинх. вход сброса

zad 5

```
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 (CLK'event and CLK = '1') then
if (J = '0' and K = '1') then
temp <= '0';
elsif (J = '1' and K = '0') then
temp <= '1';
elsif (J = '1' and K = '1') then
temp <= not temp;
end if;
end if;
end process;
Q <= temp;
nQ <= not temp;
end Behavioral;
```

```
entity TestCounterDown is
-- Port ( );
end TestCounterDown;
architecture Behavioral of
TestCounterDown is
component CounterDown
Port (
CLK : in STD_LOGIC;
CLEAR : in STD_LOGIC;
Q : out STD_LOGIC_VECTOR (7
downto 0) );
end component;
CONSTANT period : time := 10 ns;
signal CLK : STD_LOGIC := '0';
signal CLEAR : STD_LOGIC := '1';
signal Q : STD_LOGIC_VECTOR (7
downto 0);
begin
test: CounterDown port map(CLK,
CLEAR, Q);
process begin
wait for period*2;
CLEAR <= '0';
wait for period;
CLEAR <= '1';
wait for period*16;
end process;
process begin
wait for period / 2;
CLK <= not CLK;
end process; end Behavioral;
```

синх. jk, инверс. асинх. вход сброса,
8р., вычит. счет., инверс. асинх. вход сброса

zad 6

```
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;
```

синх. d, инверс. асинх. вход сброса,
8р., вычит. счет., инверс. асинх. вход сброса

```
entity TestCounterDown is
-- Port ( );
end TestCounterDown;
architecture Behavioral of
TestCounterDown is
component CounterDown
Port (
CLK : in STD_LOGIC;
CLEAR : in STD_LOGIC;
Q : out STD_LOGIC_VECTOR (7 downto
0)
);
end component;
CONSTANT period : time := 10 ns;
signal CLK : STD_LOGIC := '0';
signal CLEAR : STD_LOGIC := '1';
signal Q : STD_LOGIC_VECTOR (7
downto 0);
begin
test: CounterDown port map(CLK,
CLEAR, Q);
process
begin
wait for period*2;
CLEAR <= '0';
wait for period;
CLEAR <= '1';
wait for period*16;
end process;
process
begin
wait for period / 2;
CLK <= not CLK;
end process;
end Behavioral;
```

```
entity PosledReg is 3
Port (
CLK : in STD_LOGIC;
CLEAR : in STD_LOGIC;
D : in STD_LOGIC;
Q : out STD_LOGIC_VECTOR (7 downto 0)
);
end PosledReg;
architecture Behavioral of PosledReg 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, CLEAR, D, nD, outQ(0), outnQ(0));
JKTrigger1: JKTrigger port map (CLK, CLEAR, outQ(0), outnQ(0), outQ(1), outnQ(1));
JKTrigger2: JKTrigger port map (CLK, CLEAR, outQ(1), outnQ(1), outQ(2), outnQ(2));
JKTrigger3: JKTrigger port map (CLK, CLEAR, outQ(2), outnQ(2), outQ(3), outnQ(3));
JKTrigger4: JKTrigger port map (CLK, CLEAR, outQ(3), outnQ(3), outQ(4), outnQ(4));
JKTrigger5: JKTrigger port map (CLK, CLEAR, outQ(4), outnQ(4), outQ(5), outnQ(5));
JKTrigger6: JKTrigger port map (CLK, CLEAR, outQ(5), outnQ(5), outQ(6), outnQ(6));
JKTrigger7: JKTrigger port map (CLK, CLEAR, outQ(6), outnQ(6), outQ(7), outnQ(7));
Q <= outQ;
end Behavioral;
```

```
entity ParalReg is 2
Port (
CLK : in STD_LOGIC;
CLEAR : in STD_LOGIC;
D : in STD_LOGIC_VECTOR (7 downto 0);
Q : out STD_LOGIC_VECTOR (7 downto 0)
);
end ParalReg;
architecture Behavioral of ParalReg 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, CLEAR, D(0), outQ(0), outnQ(0));
DTrigger1: DTrigger port map (CLK, CLEAR, D(1), outQ(1), outnQ(1));
DTrigger2: DTrigger port map (CLK, CLEAR, D(2), outQ(2), outnQ(2));
DTrigger3: DTrigger port map (CLK, CLEAR, D(3), outQ(3), outnQ(3));
DTrigger4: DTrigger port map (CLK, CLEAR, D(4), outQ(4), outnQ(4));
DTrigger5: DTrigger port map (CLK, CLEAR, D(5), outQ(5), outnQ(5));
DTrigger6: DTrigger port map (CLK, CLEAR, D(6), outQ(6), outnQ(6));
DTrigger7: DTrigger port map (CLK, CLEAR, D(7), outQ(7), outnQ(7));
Q <= outQ;
end Behavioral;
```

```
entity ParalReg is 1
Port (
CLK : in STD_LOGIC;
CLEAR : in STD_LOGIC;
D : in STD_LOGIC_VECTOR (7 downto 0);
Q : out STD_LOGIC_VECTOR (7 downto 0)
);
end ParalReg;
architecture Behavioral of ParalReg 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_VECTOR (7 downto 0);
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, CLEAR, D(0), nD(0), outQ(0), outnQ(0));
JKTrigger1: JKTrigger port map (CLK, CLEAR, D(1), nD(1), outQ(1), outnQ(1));
JKTrigger2: JKTrigger port map (CLK, CLEAR, D(2), nD(2), outQ(2), outnQ(2));
JKTrigger3: JKTrigger port map (CLK, CLEAR, D(3), nD(3), outQ(3), outnQ(3));
JKTrigger4: JKTrigger port map (CLK, CLEAR, D(4), nD(4), outQ(4), outnQ(4));
JKTrigger5: JKTrigger port map (CLK, CLEAR, D(5), nD(5), outQ(5), outnQ(5));
JKTrigger6: JKTrigger port map (CLK, CLEAR, D(6), nD(6), outQ(6), outnQ(6));
JKTrigger7: JKTrigger port map (CLK, CLEAR, D(7), nD(7), outQ(7), outnQ(7));
Q <= outQ;
end Behavioral;
```

```
entity CounterDown is 6
Port (
CLK : in STD_LOGIC;
CLEAR : in STD_LOGIC;
Q : out STD_LOGIC_VECTOR (7 downto 0)
);
end CounterDown;
architecture Behavioral of CounterDown 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, CLEAR, outnQ(0), outQ(0), outnQ(0));
DTrigger1: DTrigger port map (outQ(0), CLEAR, outnQ(1), outQ(1), outnQ(1));
DTrigger2: DTrigger port map (outQ(1), CLEAR, outnQ(2), outQ(2), outnQ(2));
DTrigger3: DTrigger port map (outQ(2), CLEAR, outnQ(3), outQ(3), outnQ(3));
DTrigger4: DTrigger port map (outQ(3), CLEAR, outnQ(4), outQ(4), outnQ(4));
DTrigger5: DTrigger port map (outQ(4), CLEAR, outnQ(5), outQ(5), outnQ(5));
DTrigger6: DTrigger port map (outQ(5), CLEAR, outnQ(6), outQ(6), outnQ(6));
DTrigger7: DTrigger port map (outQ(6), CLEAR, outnQ(7), outQ(7), outnQ(7));
Q <= outQ;
end Behavioral;
```

```
entity CounterDown is 5
Port (
CLK : in STD_LOGIC;
CLEAR : in STD_LOGIC;
Q : out STD_LOGIC_VECTOR (7 downto 0)
);
end CounterDown;
architecture Behavioral of CounterDown 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, CLEAR, '1', '1', outQ(0), outnQ(0));
JKTrigger1: JKTrigger port map (outQ(0), CLEAR, '1', '1', outQ(1), outnQ(1));
JKTrigger2: JKTrigger port map (outQ(1), CLEAR, '1', '1', outQ(2), outnQ(2));
JKTrigger3: JKTrigger port map (outQ(2), CLEAR, '1', '1', outQ(3), outnQ(3));
JKTrigger4: JKTrigger port map (outQ(3), CLEAR, '1', '1', outQ(4), outnQ(4));
JKTrigger5: JKTrigger port map (outQ(4), CLEAR, '1', '1', outQ(5), outnQ(5));
JKTrigger6: JKTrigger port map (outQ(5), CLEAR, '1', '1', outQ(6), outnQ(6));
JKTrigger7: JKTrigger port map (outQ(6), CLEAR, '1', '1', outQ(7), outnQ(7));
Q <= outQ;
end Behavioral;
```

```
entity PosledReg is 4
Port (
CLK : in STD_LOGIC;
CLEAR : in STD_LOGIC;
D : in STD_LOGIC;
Q : out STD_LOGIC_VECTOR (7 downto 0)
);
end PosledReg;
architecture Behavioral of PosledReg 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, CLEAR, D, outQ(0), outnQ(0));
DTrigger1: DTrigger port map (CLK, CLEAR, outQ(0), outQ(1), outnQ(1));
DTrigger2: DTrigger port map (CLK, CLEAR, outQ(1), outQ(2), outnQ(2));
DTrigger3: DTrigger port map (CLK, CLEAR, outQ(2), outQ(3), outnQ(3));
DTrigger4: DTrigger port map (CLK, CLEAR, outQ(3), outQ(4), outnQ(4));
DTrigger5: DTrigger port map (CLK, CLEAR, outQ(4), outQ(5), outnQ(5));
DTrigger6: DTrigger port map (CLK, CLEAR, outQ(5), outQ(6), outnQ(6));
DTrigger7: DTrigger port map (CLK, CLEAR, outQ(6), outQ(7), outnQ(7));
Q <= outQ; end Behavioral;
```

зад 7

```
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 (CLK'event and CLK = '1') then
if (J = '0' and K = '1') then
temp <= '0';
elsif (J = '1' and K = '0') then
temp <= '1';
elsif (J = '1' and K = '1') then
temp <= not temp;
end if;
end if;
end process;
Q <= temp;
nQ <= not temp;
end Behavioral;
```

синх. jk, инверс. асинх. вход сброса,
8р., сум. счет., инверс. асинх. вход сброса

```
entity TestCounterUp is
-- Port ( );
end TestCounterUp;
architecture Behavioral of TestCounterUp is
component CounterUp
Port (
CLK : in STD_LOGIC;
CLEAR : in STD_LOGIC;
Q : out STD_LOGIC_VECTOR (7 downto 0)
);
end component;
CONSTANT period : time := 10 ns;
signal CLK : STD_LOGIC := '0';
signal CLEAR : STD_LOGIC := '1';
signal Q : STD_LOGIC_VECTOR (7 downto 0);
begin
test: CounterUp port map(CLK, CLEAR, Q);
process
begin
wait for period;
CLEAR <= '0';
wait for period;
CLEAR <= '1';
wait for period*16;
end process;
process
begin
wait for period / 2;
CLK <= not CLK;
end process;
end Behavioral;
```

zad 8

```
entity TestCounterUp is
-- Port ( );
end TestCounterUp;
architecture Behavioral of
TestCounterUp is
component CounterUp
Port (
CLK : in STD_LOGIC;
CLEAR : in STD_LOGIC;
Q : out STD_LOGIC_VECTOR (7
downto 0)
);
end component;
CONSTANT period : time := 10 ns;
signal CLK : STD_LOGIC := '0';
signal CLEAR : STD_LOGIC := '1';
signal Q : STD_LOGIC_VECTOR (7
downto 0);
begin
test: CounterUp port map(CLK,
CLEAR, Q);
process begin
wait for period;
CLEAR <= '0';
wait for period;
CLEAR <= '1';
wait for period*16;
end process;
process begin
wait for period / 2;
CLK <= not CLK;
end process;
end Behavioral;
```

```
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;
```

синх. d, инверс. асинх. вход сброса,
8р., сум. счет., инверс. асинх. вход сброса

zad 9

```
entity TestParalReg is
-- Port ( );
end TestParalReg;
architecture Behavioral of TestParalReg is
component ParalReg
Port (
CLK : in STD_LOGIC;
OUTEN : 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 OUTEN : STD_LOGIC := '0';
signal D : STD_LOGIC_VECTOR (7
downto 0);
signal Q : STD_LOGIC_VECTOR (7
downto 0);
begin
test: ParalReg port map(CLK, OUTEN, D,
Q); process begin
D <= "00000000"; wait for period;
D <= "11111111"; wait for period;
OUTEN <= '1';
wait for period;
OUTEN <= '0';
D <= "10010100";
wait for period;
end process; process begin
wait for period / 2;
CLK <= not CLK;
end process;
end Behavioral;
```

```
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 (CLK'event and CLK = '1') then
if (J = '0' and K = '1') then
temp <= '0';
elsif (J = '1' and K = '0') then
temp <= '1';
elsif (J = '1' and K = '1') then
temp <= not temp;
end if;
end if;
end process;
Q <= temp;
nQ <= not temp;
end Behavioral;
```

синх. jk,
8р., парал. рег., инверс. асинх. вход разреш. чтен. out_en

zad 10

```
entity TestParalReg is
-- Port ( );
end TestParalReg;
architecture Behavioral of TestParalReg is
component ParalReg
Port (
CLK : in STD_LOGIC;
OUTEN : 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 OUTEN : STD_LOGIC := '0';
signal D : STD_LOGIC_VECTOR (7 downto 0);
signal Q : STD_LOGIC_VECTOR (7 downto 0);
begin
test: ParalReg port map(CLK, OUTEN, D, Q);
process
begin
D <= "00000000";
wait for period;
D <= "11111111";
wait for period;
OUTEN <= '1';
wait for period;
OUTEN <= '0';
D <= "10010100";
wait for period;
end process;
process
begin
wait for period / 2;
CLK <= not CLK;
end process;
end Behavioral;
```

```
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 (CLK'event and CLK = '1') then
temp <= D;
end if;
end process;
Q <= temp;
nQ <= not temp;
end Behavioral;
```

синх. d,
8р., парал. рег., инверс. асинх. вход разреш. чтен. out_en

zad 11

```
entity TestCounterUp is
-- Port ( );
end TestCounterUp;
architecture Behavioral of TestCounterUp is
component CounterUp
Port (
CLK : in STD_LOGIC;
OUTEN : in STD_LOGIC;
Q : out STD_LOGIC_VECTOR (7 downto 0)
);
end component;
CONSTANT period : time := 10 ns;
signal CLK : STD_LOGIC := '0';
signal OUTEN : STD_LOGIC := '0';
signal Q : STD_LOGIC_VECTOR (7 downto
0);
begin
test: CounterUp port map(CLK, OUTEN, Q);
process
begin
wait for period;
OUTEN <= '1';
wait for period;
OUTEN <= '0';
wait for period*16;
end process;
process
begin
wait for period / 2;
CLK <= not CLK;
end process;
end Behavioral;
```

```
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 (CLK'event and CLK = '1') then
if (J = '0' and K = '1') then
temp <= '0';
elsif (J = '1' and K = '0') then
temp <= '1';
elsif (J = '1' and K = '1') then
temp <= not temp;
end if;
end if;
end process;
Q <= temp;
nQ <= not temp;
end Behavioral;
```

синх. jk,
8р., сумм. счет., с выходами с третьим сост.

zad 12

```
entity TestCounterUp is
-- Port ( );
end TestCounterUp;
architecture Behavioral of
TestCounterUp is
component CounterUp
Port (
CLK : in STD_LOGIC;
OUTEN : in STD_LOGIC;
Q : out STD_LOGIC_VECTOR (7
downto 0) );
end component;
CONSTANT period : time := 10 ns;
signal CLK : STD_LOGIC := '0';
signal OUTEN : STD_LOGIC := '0';
signal Q : STD_LOGIC_VECTOR (7
downto 0);
begin
test: CounterUp port map(CLK,
OUTEN, Q);
process begin
wait for period;
OUTEN <= '1';
wait for period;
OUTEN <= '0';
wait for period*16;
end process; process
begin wait for period / 2;
CLK <= not CLK; end process;
end Behavioral;
```

```
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 (CLK'event and CLK = '1') then
temp <= D;
end if;
end process;
Q <= temp;
nQ <= not temp;
end Behavioral;
```

синх. d,
8р., сумм. счет., с выходами с третьим сост.

```
entity ParalReg is 9
Port (
CLK : in STD_LOGIC;
OUTEN : in STD_LOGIC;
D : in STD_LOGIC_VECTOR (7 downto 0);
Q : out STD_LOGIC_VECTOR (7 downto 0)
);
end ParalReg;
architecture Behavioral of ParalReg 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 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;
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 OUTEN = '0' else "ZZZZZZZZ";
end Behavioral;
```

```
entity CounterUp is 8
Port (
CLK : in STD_LOGIC;
CLEAR : in STD_LOGIC;
Q : out STD_LOGIC_VECTOR (7 downto 0)
);
end CounterUp;
architecture Behavioral of CounterUp 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, CLEAR, outnQ(0), outQ(0), outnQ(0));
DTrigger1: DTrigger port map (outnQ(0), CLEAR, outnQ(1), outQ(1), outnQ(1));
DTrigger2: DTrigger port map (outnQ(1), CLEAR, outnQ(2), outQ(2), outnQ(2));
DTrigger3: DTrigger port map (outnQ(2), CLEAR, outnQ(3), outQ(3), outnQ(3));
DTrigger4: DTrigger port map (outnQ(3), CLEAR, outnQ(4), outQ(4), outnQ(4));
DTrigger5: DTrigger port map (outnQ(4), CLEAR, outnQ(5), outQ(5), outnQ(5));
DTrigger6: DTrigger port map (outnQ(5), CLEAR, outnQ(6), outQ(6), outnQ(6));
DTrigger7: DTrigger port map (outnQ(6), CLEAR, outnQ(7), outQ(7), outnQ(7));
Q <= outQ;
end Behavioral;
```

```
entity CounterUp is 7
Port (
CLK : in STD_LOGIC;
CLEAR : in STD_LOGIC;
Q : out STD_LOGIC_VECTOR (7 downto 0)
);
end CounterUp;
architecture Behavioral of CounterUp 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, CLEAR, '1', '1', outQ(0), outnQ(0));
JKTrigger1: JKTrigger port map (outnQ(0), CLEAR, '1', '1', outQ(1), outnQ(1));
JKTrigger2: JKTrigger port map (outnQ(1), CLEAR, '1', '1', outQ(2), outnQ(2));
JKTrigger3: JKTrigger port map (outnQ(2), CLEAR, '1', '1', outQ(3), outnQ(3));
JKTrigger4: JKTrigger port map (outnQ(3), CLEAR, '1', '1', outQ(4), outnQ(4));
JKTrigger5: JKTrigger port map (outnQ(4), CLEAR, '1', '1', outQ(5), outnQ(5));
JKTrigger6: JKTrigger port map (outnQ(5), CLEAR, '1', '1', outQ(6), outnQ(6));
JKTrigger7: JKTrigger port map (outnQ(6), CLEAR, '1', '1', outQ(7), outnQ(7));
Q <= outQ;
end Behavioral;
```

```
entity CounterUp is 12
Port (
CLK : in STD_LOGIC;
OUTEN : in STD_LOGIC;
Q : out STD_LOGIC_VECTOR (7 downto 0)
);
end CounterUp;
architecture Behavioral of CounterUp 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 OUTEN = '0' else "ZZZZZZZZ";
end Behavioral;
```

```
entity CounterUp is 11
Port (
CLK : in STD_LOGIC;
OUTEN : in STD_LOGIC;
Q : out STD_LOGIC_VECTOR (7 downto 0)
);
end CounterUp;
architecture Behavioral of CounterUp 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 OUTEN = '0' else "ZZZZZZZZ";
end Behavioral;
```

```
entity ParalReg is 10
Port (
CLK : in STD_LOGIC;
OUTEN : in STD_LOGIC;
D : in STD_LOGIC_VECTOR (7 downto 0);
Q : out STD_LOGIC_VECTOR (7 downto 0)
);
end ParalReg;
architecture Behavioral of ParalReg is
component DTrigger
Port (
CLK : 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, 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 OUTEN = '0' else "ZZZZZZZZ";
end Behavioral;
```

зад 13

```
entity TestCounterDown is
-- Port ( );
end TestCounterDown;
architecture Behavioral of
TestCounterDown is
component CounterDown
Port (
CLK : in STD_LOGIC;
OUTEN : in STD_LOGIC;
Q : out STD_LOGIC_VECTOR (7
downto 0));
end component;
CONSTANT period : time := 10 ns;
signal CLK : STD_LOGIC := '0';
signal OUTEN : STD_LOGIC := '0';
signal Q : STD_LOGIC_VECTOR (7
downto 0);
begin
test: CounterDown port map(CLK,
OUTEN, Q);
process
begin
wait for period;
wait for period;
OUTEN <= '1';
wait for period;
OUTEN <= '0';
wait for period*16;
end process;
process begin
wait for period / 2;
CLK <= not CLK;
end process;
end Behavioral;
```

```
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 (CLK'event and CLK = '1') then
if (J = '0' and K = '1') then
temp <= '0';
elsif (J = '1' and K = '0') then
temp <= '1';
elsif (J = '1' and K = '1') then
temp <= not temp;
end if;
end if;
end process;
Q <= temp;
nQ <= not temp;
end Behavioral;
```

синх. jk,
8р., вычит. счет., с выходами с третьим сост.

zad 14

```
entity TestCounterDown is
-- Port ( );
end TestCounterDown;
architecture Behavioral of
TestCounterDown is
component CounterDown
Port (
CLK : in STD_LOGIC;
OUTEN : in STD_LOGIC;
Q : out STD_LOGIC_VECTOR (7
downto 0) );
end component;
CONSTANT period : time := 10 ns;
signal CLK : STD_LOGIC := '0';
signal OUTEN : STD_LOGIC := '0';
signal Q : STD_LOGIC_VECTOR (7
downto 0);
begin
test: CounterDown port map(CLK,
OUTEN, Q);
process begin
wait for period;
wait for period;
OUTEN <= '1';
wait for period;
OUTEN <= '0';
wait for period*16;
end process;
process begin
wait for period / 2;
CLK <= not CLK;
end process;
end Behavioral;
```

```
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 (CLK'event and CLK = '1') then
temp <= D;
end if;
end process;
Q <= temp;
nQ <= not temp;
end Behavioral;
```

синх. d,
8р., вычит. счет., с выходами с третьим сост.

```
entity CounterDown is 14
Port (
CLK : in STD_LOGIC;
OUTEN : in STD_LOGIC;
Q : out STD_LOGIC_VECTOR (7 downto 0)
);
end CounterDown;
architecture Behavioral of CounterDown 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 OUTEN = '0' else "ZZZZZZZZ";
end Behavioral;
```

```
entity CounterDown is 13
Port (
CLK : in STD_LOGIC;
OUTEN : in STD_LOGIC;
Q : out STD_LOGIC_VECTOR (7 downto 0)
);
end CounterDown;
architecture Behavioral of CounterDown 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 OUTEN = '0' else "ZZZZZZZZ";
end Behavioral;
```