entity TestPosledReg is зад 1 entity TestParalReg is zad 2 entity TestParalReg is zad 3 -- Port (); -- Port (); entity JKTrigger is -- Port (): entity JKTrigger is entity DTrigger is end TestPosledRea: end TestParalRea: end TestParalReg: Port (Port (Port (architecture Behavioral of TestPosledBeg is architecture Behavioral of TestParalReg is architecture Behavioral of TestParalReg is CLK: in STD LOGIC; CLK: in STD LOGIC: CLK: in STD LOGIC; component PosledRea component ParalReg component ParalReg R: in STD_LOGIC: R: in STD LOGIC: R: in STD LOGIC: Port (CLK : in STD LOGIC: Port (CLEAR: in STD LOGIC: J: in STD LOGIC: CLK : in STD LOGIC: D: in STD LOGIC: CLK : in STD LOGIC: J: in STD LOGIC: D : in STD_LOGIC: K : in STD_LOGIC: CLEAR : in STD LOGIC: Q : out STD_LOGIC: CLEAR : in STD LOGIC: K : in STD_LOGIC: Q : out STD_LOGIC_VECTOR (7 downto 0)): D: in STD LOGIC VECTOR (7 downto 0): D: in STD LOGIC VECTOR (7 downto 0): Q : out STD LOGIC: nQ : out STD LOGIC): Q : out STD LOGIC: end component; CONSTANT period : time := 10 ns; Q : out STD LOGIC VECTOR (7 downto 0) Q : out STD_LOGIC VECTOR (7 downto f) nQ : out STD LOGIC): nQ : out STD LOGIC): end DTriager: end JKTrigger; architecture Behavioral of DTrigger end JKTrigger; signal CLK : STD LOGIC := '0': end component end component architecture Behavioral of JKTrigger is architecture Behavioral of JKTrigger is signal CLEAR : STD LOGIC := '1' CONSTANT period : time := 10 ns; CONSTANT period : time := 10 ns; signal temp; std logic := '0': signal temp; std logic := '0': signal temp; std logic := '0': signal D : STD LOGIC: signal CLK : STD LOGIC := '0': signal CLK : STD LOGIC := '0': signal Q : STD LOGIC VECTOR (7 downto 0): signal CLEAR · STD LOGIC ·= '1' signal CLEAR : STD LOGIC := '1': begin begin begin begin test: PosledReg port map(CLK, CLEAR, D, Q); signal D : STD_LOGIC VECTOR (7 downto f)): signal D : STD LOGIC VECTOR (7 downto 0): process (CLK, R) process (CLK, R) process (CLK, R) process begin signal Q : STD LOGIC VECTOR (7 downto 0): signal Q : STD LOGIC VECTOR (7 downto 0): begin begin begin D <= '1': if (R = '0') then if (R = '0') then if (R = '0') then CLEAR <= '1'; test: ParalReg port map(CLK, CLEAR, D, Q); test: ParalReg port map(CLK, CLEAR, D, Q); wait for period*8; D <= '0'; CLEAR <= '1'; temp <= '0': temp <= '0': temp <= '0': process process elsif (CLK'event and CLK = '1') then elsif (CLK'event and CLK = '1') then elsif (CLK'event and CLK = '1') then wait for period*8; beain begin D <= "00000000": D <= '1' if (J = '0') and K = '1') then D <= "00000000". temp <= D; if (J = '0' and K = '1') then CLEAR <- '1' CLEAR <= '1' CLEAR <= '1'; temp <= '0' end if temp <= '0'. wait for period*8; wait for period; wait for period; elsif (J = '1' and K = '0') then end process: elsif (J = '1' and K = '0') then CLEAR <= '0'; D <= "11111111" D <= "11111111" temp <= '1': O <- temp: temn <- '1' wait for period; CLEAR <= '1' CI FAR <= '1' elsif (J = '1' and K = '1') then nQ <= not temp: elsif (J = '1' and K = '1') then CLEAR <= '1'; wait for period: wait for period: D <= '1'; temp <= not temp: end Behavioral temp <= not temp: CLEAR <= '0': CLEAR <= '0': wait for period; end if: wait for period: end if: wait for period D <= '0'; wait for period; D <= "10010100": D <= "10010100": end if синх. d, инверс. асинх. вход end if D <= '0'; wait for period; CLEAR <= '1': CLEAR <= '1': end process; end process; D <= '1'; wait for period; wait for period: wait for period; Q <= temp; 8р., парал. рег., инверс. асинх. Q <= temp; D <= '0': end process; end process; nQ <= not temp: вход сброса nQ <= not temp: wait for period; process process end Behavioral: D <= '1'; end Behavioral begin begin wait for period; end process; process begin wait for period / 2; wait for period / 2; wait for period / 2; CLK <= not CLK; CLK <= not CLK; синх. jk, инверс. асинх. вход синх. јк, инверс. асинх. вход CLK <= not CLK; end process: end process: сброса. сброса. end process: end Behavioral: end Rehavioral 8р., парал. рег., инверс. асинх. 8р., послед. рег., инверс. асинх. end Behavioral; вход сброса вход сброса entity TestCounterDown is zad 4 zad 5 zad 6 entity DTriager is entity TestCounterDown is -- Port (): entity TestPosledReg is entity JKTrigger is entity DTrigger is -- Port (); Port (end TestCounterDown; -- Port (): end TestCounterDown; Port (Port (CLK: in STD LOGIC: architecture Behavioral of end TestPosledReg; architecture Behavioral of CLK: in STD LOGIC: CLK: in STD LOGIC: architecture Behavioral of TestPosledReg is R: in STD LOGIC: TestCounterDown is TestCounterDown is component PosledRea R: in STD LOGIC: R: in STD LOGIC: component CounterDown D: in STD LOGIC; component CounterDown Port (CLK : in STD LOGIC: J: in STD_LOGIC: D: in STD_LOGIC: Port (Q: out STD LOGIC: CLEAR : in STD LOGIC: Port (CLK : in STD LOGIC: K: in STD LOGIC; Q: out STD LOGIC: D : in STD_LOGIC; nQ: out STD LOGIC): CLK : in STD LOGIC: CLEAR: in STD LOGIC; Q : out STD LOGIC VECTOR (7 downto 0)): Q: out STD LOGIC: nQ: out STD LOGIC): end DTrigger; CLEAR: in STD_LOGIC: end component; nQ: out STD LOGIC); Q: out STD LOGIC VECTOR (7 end DTrigger; architecture Behavioral of DTrigger is CONSTANT period : time := 10 ns; downto 0)): architecture Behavioral of DTrigger is signal CLK : STD_LOGIC := '0'; signal temp: std_logic := '0'; end JKTrigger; signal CLEAR : STD_LOGIC := '1'; end component; architecture Behavioral of JKTrigger is signal temp: std_logic := '0'; begin end component:

signal D · STD I OGIC signal Q : STD LOGIC VECTOR (7 downto 0): beain 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: $D \le 0$ wait for period: $D \le 0$ ' wait for period: D <= '1': wait for period: D <= '0'; wait for period; D <= '1' wait for period; end process: process beain wait for period / 2: CLK <= not CLK: end process: end Behavioral:

process (CLK, R) beain if (R = '0') then temp <= '0': elsif (CLK'event and CLK = '1') then temp <= D;end if: end process; $Q \le temp$: nQ <= not temp;

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

end Behavioral:

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 process; Q <= temp; nQ <= not temp: end Behavioral:

CONSTANT period : time := 10 ns: signal CLK : STD LOGIC := '0': signal CLEAR : STD_LOGIC := '1': signal Q: STD_LOGIC_VECTOR (7 downto 0). 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. ік. инверс. асинх. вход 8р., вычит. счет., инверс. асинх. вход сброса

begin process (CLK, R) beain 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

Q : out STD LOGIC VECTOR (7 downto 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 beain wait for period / 2; CLK <= not CLK; end process: end Behavioral;

entity PosledReg is 3 entity ParalReg is 2 entity ParalReg is 1 Port (Port (Port (CLK: in STD LOGIC: CLK : in STD LOGIC: CLK: in STD LOGIC: CLEAR: in STD LOGIC; CLEAR: in STD LOGIC; CLEAR: in STD LOGIC; D: in STD LOGIC: D: in STD LOGIC VECTOR (7 downto 0): D: in STD LOGIC VECTOR (7 downto 0): Q: out STD_LOGIC_VECTOR (7 downto 0) Q : out STD LOGIC VECTOR (7 downto 0) Q: out STD LOGIC VECTOR (7 downto 0) end PosledRea: end ParalRea: architecture Behavioral of ParalReg is architecture Behavioral of PosledReg is architecture Behavioral of ParalReg is component JKTrigger component DTrigger component JKTrigger Port (Port (Port (CLK: in STD LOGIC: CLK : in STD LOGIC: CLK: in STD LOGIC: R: in STD LOGIC; R: in STD LOGIC; R: in STD LOGIC; J: in STD LOGIC: D: in STD LOGIC: J : in STD LOGIC: Q : out STD_LOGIC; K: in STD_LOGIC; K: in STD_LOGIC; Q : out STD_LOGIC: nQ: out STD_LOGIC): Q : out STD_LOGIC: nQ: out STD LOGIC): end component: nQ: out STD LOGIC): signal nD: STD LOGIC VECTOR (7 downto 0): end component: end component: signal nD: STD LOGIC VECTOR (7 downto 0); signal nD: STD LOGIC; signal outQ: STD LOGIC VECTOR (7 downto 0); signal outQ: STD LOGIC VECTOR (7 downto 0): signal outnQ : STD LOGIC VECTOR (7 downto 0): signal outQ : STD LOGIC VECTOR (7 downto 0): signal outnQ: STD LOGIC VECTOR (7 downto 0): beain signal outnQ: STD LOGIC VECTOR (7 downto 0): begin nD <= not D: begin nD <= not D: DTrigger0: DTrigger port map (CLK, CLEAR, D(0), outQ(0), outnQ(0)): nD <= not D: DTrigger1: DTrigger port map (CLK, CLEAR, D(1), outQ(1), outnQ(1)); JKTrigger0: JKTrigger port map (CLK, CLEAR, D. nD, outQ(0), outnQ(0)); JKTrigger0: JKTrigger port map (CLK, CLEAR, D(0), nD(0), outQ(0), outnQ(0)); DTrigger2: DTrigger port map (CLK, CLEAR, D(2), outQ(2), outnQ(2)); JKTrigger1: JKTrigger port map (CLK, CLEAR, outQ(0), outnQ(0), outQ(1), outnQ(1)); JKTrigger1: JKTrigger port map (CLK, CLEAR, D(1), nD(1), outQ(1), outnQ(1)); JKTrigger2: JKTrigger port map (CLK, CLEAR, outQ(1), outnQ(1), outQ(2), outnQ(2)); DTrigger3: DTrigger port map (CLK, CLEAR, D(3), outQ(3), outnQ(3)); JKTrigger2: JKTrigger port map (CLK, CLEAR, D(2), nD(2), outQ(2), outnQ(2)); JKTrigger3: JKTrigger port map (CLK, CLEAR, outQ(2), outnQ(2), outQ(3), outnQ(3)); DTrigger4: DTrigger port map (CLK, CLEAR, D(4), outQ(4), outnQ(4)); JKTrigger3: JKTrigger port map (CLK, CLEAR, D(3), nD(3), outQ(3), outnQ(3)); JKTrigger4: JKTrigger port map (CLK, CLEAR, outQ(3), outnQ(3), outQ(4), outnQ(4)); DTrigger5: DTrigger port map (CLK, CLEAR, D(5), outQ(5), outnQ(5)); JKTrigger4: JKTrigger port map (CLK, CLEAR, D(4), nD(4), outQ(4), outnQ(4)); JKTrigger5: JKTrigger port map (CLK, CLEAR, outQ(4), outnQ(4), outQ(5), outnQ(5)); DTrigger6: DTrigger port map (CLK, CLEAR, D(6), outQ(6), outnQ(6)); JKTrigger5: JKTrigger port map (CLK, CLEAR, D(5), nD(5), outQ(5), outnQ(5)); JKTrigger6: JKTrigger port map (CLK, CLEAR, outQ(5), outnQ(5), outQ(6), outnQ(6)); DTrigger7: DTrigger port map (CLK, CLEAR, D(7), outQ(7), outnQ(7)); JKTrigger6: JKTrigger port map (CLK, CLEAR, D(6), nD(6), outQ(6), outnQ(6)); JKTrigger7: JKTrigger port map (CLK, CLEAR, outQ(6), outnQ(6), outQ(7), outnQ(7)); O <= outO: JKTrigger7: JKTrigger port map (CLK, CLEAR, D(7), nD(7), outQ(7), outnQ(7)); $Q \le outQ$: end Behavioral $Q \le outQ$: end Behavioral: end Behavioral: entity CounterDown is 5 entity PosledRea is 4 entity CounterDown is 6 Port (Port (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);
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 \le outQ;
end Behavioral;
```

```
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 \le outQ;
end Behavioral;
```

```
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 \le 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': process (CLK, R) beain 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; nO < - not temp end Behavioral: синх. jk, инверс. асинх. вход end Behavioral: сброса. 8р., сум. счет., инверс. асинх. вход сброса

entity TestCounterUp is -- Port (); end TestCounterUp: architecture Behavioral of TestCounterUp is component CounterUp 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); 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 beain wait for period / 2; CLK <= not CLK; end process:

70d Q 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): hegin 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:

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 DTriager: architecture Behavioral of DTrigger is signal temp: std logic := '0'; beain process (CLK, R) begin if $(\ddot{R} = '0')$ 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

entity JKTrigger is

Port (

70d Q entity TestParalReg is -- Port (); end TestParalReg: architecture Behavioral of TestParalReg is component ParalRea 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;

end if: end process: Q <= temp; nQ <= not temp: end process; process begin end Behavioral:

CLK <= not CLK: синх. јк, end process: 8р., парал. рег., инверс. асинх. end Behavioral; вход разреш. чтен. out en

entity JKTrigger is

J: in STD_LOGIC:

K: in STD LOGIC:

end JKTrigger;

process (CLK)

temp <= '0':

temp <= '1';

temp <= not temp:

Q : out STD LOGIC:

nQ: out STD LOGIC):

architecture Behavioral of JKTrigger

if (ČLK'event and CLK = '1') then

signal temp: std logic := '0':

if (J = '0') and K = '1') then

elsif (J = '1') and K = '0') then

elsif (J = '1') and K = '1') then

CLK: in STD LOGIC:

Port (

begin

beain

end if:

zad 10

D <= "10010100"

wait for period / 2:

CLK <= not CLK:

end process:

end Behavioral

wait for period;

end process;

process

entity TestParalReg is -- Port (); end TestParalReg: architecture Behavioral of TestParalReg is component ParalReg 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); beain 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'

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) if (CLK'event and CLK = '1') then temp <= D;end if: end process: $Q \le \text{temp}$: nQ <= not temp:

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

end Behavioral:

zad 11

process begin

end process:

wait for period / 2;

CLK <= not CLK:

end Behavioral:

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 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 beain wait for period / 2; CLK <= not CLK; end process: end Behavioral;

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 JKTriager is signal temp: std logic := '0'; beain 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: $Q \le \text{temp}$: $nQ \le not temp$:

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

end Behavioral:

zad 12

OUTEN <= '0':

wait for period:

D <= "10010100";

wait for period / 2:

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 Behavioral:

end process; process

begin wait for period / 2;

CLK <= not CLK; end process;

entity DTrigger is Port (CLK: in STD LOGIC: D: in STD LOGIC: Q: out STD LOGIC; nQ: out STD_LOGIC); end DTriager: architecture Behavioral of DTriager is signal temp: std_logic := '0'; begin process (CLK) beain 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
                                                                                       entity CounterUp is 8
                                                                                                                                                                                entity CounterUp is 7
Port (
                                                                                                                                                                               Port (
                                                                                      Port (
CLK : in STD_LOGIC;
                                                                                                                                                                               CLK : in STD LOGIC:
                                                                                      CLK: in STD LOGIC;
OUTEN: in STD LOGIC:
                                                                                                                                                                               CLEAR: in STD LOGIC:
                                                                                      CLEAR: in STD LOGIC;
D: in STD_LOGIC_VECTOR (7 downto 0):
                                                                                                                                                                               Q : out STD_LOGIC_VECTOR (7 downto 0)
                                                                                      Q: out STD LOGIC VECTOR (7 downto 0)
Q: out STD LOGIC VECTOR (7 downto 0)
                                                                                                                                                                               end CounterUp;
end ParalReg:
                                                                                      end CounterUp:
                                                                                                                                                                               architecture Behavioral of CounterUp is
architecture Behavioral of ParalReg is
                                                                                      architecture Behavioral of CounterUp is
                                                                                                                                                                               component JKTrigger
component JKTrigger
                                                                                                                                                                               Port (
                                                                                      component DTrigger
                                                                                                                                                                               CLK : in STD LOGIC;
Port (
                                                                                      Port (
CLK in STD LOGIC:
                                                                                                                                                                               B · in STD I OGIC:
                                                                                      CLK: in STD LOGIC:
J: in STD LOGIC;
                                                                                                                                                                               J: in STD LOGIC:
                                                                                      R: in STD_LOGIC:
K: in STD LOGIC;
                                                                                                                                                                               K: in STD LOGIC:
                                                                                      D: in STD LOGIC:
Q : out STD LOGIC:
                                                                                                                                                                               Q : out STD LOGIC:
nQ : out STD_LOGIC):
                                                                                      Q : out STD LOGIC:
                                                                                                                                                                               nQ : out STD_LOGIC):
end component;
                                                                                      nQ: out STD LOGIC):
                                                                                                                                                                               end component;
signal nD: STD LOGIC VECTOR (7 downto 0):
                                                                                                                                                                               signal outQ : STD LOGIC VECTOR (7 downto 0):
                                                                                      end component:
signal outQ : STD_LOGIC_VECTOR (7 downto 0):
                                                                                                                                                                                signal outnQ: STD_LOGIC_VECTOR (7 downto 0):
                                                                                      signal outQ: STD LOGIC VECTOR (7 downto 0):
signal outnQ : STD LOGIC VECTOR (7 downto 0);
                                                                                      signal outnQ: STD_LOGIC_VECTOR (7 downto 0):
                                                                                                                                                                                JKTrigger0: JKTrigger port map (CLK, CLEAR, '1', '1', outQ(0), outnQ(0));
beain
                                                                                      begin
                                                                                                                                                                                JKTrigger1: JKTrigger port map (outnQ(0), CLEAR, '1', '1', outQ(1), outnQ(1));
nD \le not D:
                                                                                       DTrigger0: DTrigger port map (CLK, CLEAR, outnQ(0), outQ(0), outnQ(0));
JKTrigger0: JKTrigger port map (CLK, D(0), nD(0), outQ(0), outnQ(0));
                                                                                                                                                                                JKTrigger2: JKTrigger port map (outnQ(1), CLEAR, '1', '1', outQ(2), outnQ(2));
JKTrigger1: JKTrigger port map (CLK, D(1), nD(1), outQ(1), outnQ(1)):
                                                                                       DTrigger1: DTrigger port map (outnQ(0), CLEAR, outnQ(1), outQ(1), outnQ(1));
                                                                                                                                                                               JKTrigger3: JKTrigger port map (outnQ(2), CLEAR, '1', '1', outQ(3), outnQ(3));
JKTrigger2: JKTrigger port map (CLK, D(2), nD(2), outQ(2), outnQ(2));
                                                                                       DTrigger2: DTrigger port map (outnQ(1), CLEAR, outnQ(2), outQ(2), outnQ(2));
                                                                                                                                                                               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));
JKTrigger3: JKTrigger port map (CLK, D(3), nD(3), outQ(3), outnQ(3));
                                                                                       DTrigger3: DTrigger port map (outnQ(2), CLEAR, outnQ(3), outQ(3), outnQ(3));
JKTrigger4: JKTrigger port map (CLK, D(4), nD(4), outQ(4), outnQ(4)):
                                                                                                                                                                                JKTrigger6: JKTrigger port map (outnQ(5), CLEAR, '1', '1', outQ(6), outnQ(6));
                                                                                       DTrigger4: DTrigger port map (outnQ(3), CLEAR, outnQ(4), outQ(4), outnQ(4)):
JKTrigger5: JKTrigger port map (CLK, D(5), nD(5), outQ(5), outnQ(5));
                                                                                                                                                                               JKTrigger7: JKTrigger port map (outnQ(6), CLEAR, '1', '1', outQ(7), outnQ(7));
                                                                                       DTrigger5: DTrigger port map (outnQ(4), CLEAR, outnQ(5), outQ(5), outnQ(5));
JKTrigger6: JKTrigger port map (CLK, D(6), nD(6), outQ(6), outnQ(6));
                                                                                                                                                                               O \le OrtO.
                                                                                       DTrigger6: DTrigger port map (outnQ(5), CLEAR, outnQ(6), outQ(6), outnQ(6));
JKTrigger7: JKTrigger port map (CLK, D(7), nD(7), outQ(7), outnQ(7));
                                                                                                                                                                               end Behavioral:
                                                                                       DTrigger7: DTrigger port map (outnQ(6), CLEAR, outnQ(7), outQ(7), outnQ(7));
Q <= outQ when OUTEN = '0' else "ZZZZZZZZ":
end Behavioral:
                                                                                      Q \le outQ:
                                                                                       end Behavioral:
```

```
entity CounterUp is 12
                                                                              entity CounterUp is 11
                                                                                                                                                               entity ParalReg is 10
                                                                              Port (
                                                                                                                                                               Port (
CLK: in STD LOGIC;
                                                                              CLK: in STD LOGIC:
                                                                                                                                                               CLK: in STD LOGIC:
OUTEN: in STD LOGIC:
                                                                              OUTEN: in STD LOGIC:
                                                                                                                                                               OUTEN: in STD LOGIC:
Q: out STD LOGIC VECTOR (7 downto 0)
                                                                              Q: out STD LOGIC VECTOR (7 downto 0)
                                                                                                                                                               D: in STD LOGIC VECTOR (7 downto 0);
                                                                                                                                                               Q : out STD_LOGIC_VECTOR (7 downto 0)
end CounterUp:
                                                                              end CounterUp:
architecture Behavioral of CounterUp is
                                                                              architecture Behavioral of CounterUp is
                                                                                                                                                               end ParalReg;
component DTrigger
                                                                              component JKTrigger
                                                                                                                                                               architecture Behavioral of ParalReg is
Port (
                                                                              Port (
                                                                                                                                                               component DTrigger
CLK: in STD LOGIC;
                                                                              CLK: in STD_LOGIC;
                                                                                                                                                               Port (
D: in STD LOGIC;
                                                                              J: in STD LOGIC;
                                                                                                                                                               CLK: in STD LOGIC;
                                                                                                                                                               D: in STD LOGIC:
Q: out STD LOGIC:
                                                                              K: in STD LOGIC:
nQ: out STD LOGIC);
                                                                              Q: out STD LOGIC;
                                                                                                                                                               Q: out STD LOGIC;
end component;
                                                                              nQ: out STD LOGIC);
                                                                                                                                                               nQ: out STD LOGIC);
signal outQ: STD_LOGIC_VECTOR (7 downto 0);
                                                                              end component:
                                                                                                                                                               end component;
signal outnQ: STD LOGIC VECTOR (7 downto 0);
                                                                              signal outQ: STD LOGIC VECTOR (7 downto 0);
                                                                                                                                                               signal nD: STD LOGIC VECTOR (7 downto 0);
                                                                              signal outnQ: STD LOGIC VECTOR (7 downto 0);
                                                                                                                                                               signal outQ: STD LOGIC VECTOR (7 downto 0);
                                                                                                                                                               signal outnQ: STD_LOGIC_VECTOR (7 downto 0):
DTrigger0: DTrigger port map (CLK, outnQ(0), outQ(0), outnQ(0));
                                                                              beain
DTrigger1: DTrigger port map (outnQ(0), outnQ(1), outQ(1), outnQ(1));
                                                                              JKTrigger0: JKTrigger port map (CLK, '1', '1', outQ(0), outnQ(0));
                                                                                                                                                               begin
DTrigger2: DTrigger port map (outnQ(1), outnQ(2), outQ(2), outnQ(2));
                                                                              JKTrigger1: JKTrigger port map (outnQ(0), '1', '1', outQ(1), outnQ(1));
                                                                                                                                                               nD \le not D:
DTrigger3: DTrigger port map (outnQ(2), outnQ(3), outQ(3), outnQ(3));
                                                                              JKTrigger2: JKTrigger port map (outnQ(1), '1', '1', outQ(2), outnQ(2));
                                                                                                                                                               DTrigger0: DTrigger port map (CLK, D(0), outQ(0), outnQ(0));
DTrigger4: DTrigger port map (outnQ(3), outnQ(4), outQ(4), outnQ(4));
                                                                              JKTrigger3: JKTrigger port map (outnQ(2), '1', '1', outQ(3), outnQ(3));
                                                                                                                                                               DTrigger1: DTrigger port map (CLK, D(1), outQ(1), outnQ(1));
DTrigger5: DTrigger port map (outnQ(4), outnQ(5), outQ(5), outnQ(5));
                                                                              JKTrigger4: JKTrigger port map (outnQ(3), '1', '1', outQ(4), outnQ(4));
                                                                                                                                                               DTrigger2: DTrigger port map (CLK, D(2), outQ(2), outnQ(2));
DTrigger6: DTrigger port map (outnQ(5), outnQ(6), outQ(6), outnQ(6)):
                                                                              JKTrigger5: JKTrigger port map (outnQ(4), '1', '1', outQ(5), outnQ(5));
                                                                                                                                                               DTrigger3: DTrigger port map (CLK, D(3), outQ(3), outnQ(3)):
DTrigger7: DTrigger port map (outnQ(6), outnQ(7), outQ(7), outnQ(7));
                                                                              JKTrigger6: JKTrigger port map (outnQ(5), '1', '1', outQ(6), outnQ(6));
                                                                                                                                                               DTrigger4: DTrigger port map (CLK, D(4), outQ(4), outnQ(4));
Q <= outQ when OUTEN = '0' else "ZZZZZZZZZ";
                                                                              JKTrigger7: JKTrigger port map (outnQ(6), '1', '1', outQ(7), outnQ(7));
                                                                                                                                                               DTrigger5: DTrigger port map (CLK, D(5), outQ(5), outnQ(5));
end Behavioral:
                                                                              Q <= outQ when OUTEN = '0' else "ZZZZZZZZ":
                                                                                                                                                               DTrigger6: DTrigger port map (CLK, D(6), outQ(6), outnQ(6)):
                                                                              end Behavioral;
                                                                                                                                                               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'; beain process (CLK) 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):
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:
```