VJEŽBA 4- MUX (funkcija 8)

```
ENTITY mult21 IS PORT (
      E: IN Std_logic;
      S: IN Std_logic;
      D0: IN Std_logic;
      D1: IN std_logic;
      Y: OUT Std_logic);
END mult21;

ARCHITECTURE arch OF mult21 IS

BEGIN
  PROCESS (E,S,D0,D1)
  BEGIN
    IF E='1' THEN
      CASE S IS
        WHEN '0' => Y <= D0 AFTER 10 ns;
        WHEN '1' => y <= D1 AFTER 10 ns;
        WHEN OTHERS => Y <= '0' AFTER 10 ns;
      END CASE;
    ELSE
    Y <= '0' AFTER 10 ns;
    END IF;
  END PROCESS;
  END arch;

ENTITY mux161 IS PORT(                                  -- mux stablo
        D: IN std_logic_vector(0 TO 15);
        S: IN std_logic_vector(0 TO 3);
        E: IN std_logic;
        F: OUT  std_logic);
END mux161;

ARCHITECTURE arch OF mux161 IS
 COMPONENT mult21 PORT( E,S,D0,D1: IN std_logic;
                        Y: OUT std_logic);
                         END COMPONENT;

SIGNAL i: std_logic_vector( 0 TO 13);
BEGIN
   c0: entity  work.mult21 PORT MAP( D0=>D(0),D1=>D(1),E=>E,S=>S(0),Y=>i(0));
   c1: entity  work.mult21 PORT MAP( D0=>D(2),D1=>D(3),E=>E,S=>S(0),Y=>i(1));
   c2: entity  work.mult21 PORT MAP( D0=>D(4),D1=>D(5),E=>E,S=>S(0),Y=>i(2));
   c3: entity  work.mult21 PORT MAP( D0=>D(6),D1=>D(7),E=>E,S=>S(0),Y=>i(3));
   c4: entity  work.mult21 PORT MAP( D0=>D(8),D1=>D(9),E=>E,S=>S(0),Y=>i(4));
   c5: entity  work.mult21 PORT MAP( D0=>D(10),D1=>D(11),E=>E,S=>S(0),Y=>i(5));
   c6: entity  work.mult21 PORT MAP( D0=>D(12),D1=>D(13),E=>E,S=>S(0),Y=>i(6));
   c7: entity  work.mult21 PORT MAP( D0=>D(14),D1=>D(15),E=>E,S=>S(0),Y=>i(7));
   c8: entity  work.mult21 PORT MAP( D0=>i(0),D1=>i(1),E=>E,S=>S(1),Y=>i(8));
   c9: entity  work.mult21 PORT MAP( D0=>i(2),D1=>i(3),E=>E,S=>S(1),Y=>i(9));
   c10: entity  work.mult21 PORT MAP( D0=>i(4),D1=>i(5),E=>E,S=>S(1),Y=>i(10));
```

```
  c11: entity  work.mult21 PORT MAP( D0=>i(6),D1=>i(7),E=>E,S=>S(1),Y=>i(11));
  c12: entity  work.mult21 PORT MAP( D0=>i(8),D1=>i(9),E=>E,S=>S(2),Y=>i(12));
  c13: entity  work.mult21 PORT MAP( D0=>i(10),D1=>i(11),E=>E,S=>S(2),Y=>i(13));
  c14: entity  work.mult21 PORT MAP( D0=>i(12),D1=>i(13),E=>E,S=>S(3),Y=>F);
END arch;

ENTITY f8 IS PORT (                    -- ostvarenje osme funkcije
            A: IN std_logic;
            B: IN std_logic;
            C: IN std_logic;
            D: IN std_logic;
            F: OUT std_logic);
END f8;

ARCHITECTURE arch OF f8 IS
SIGNAL ulaz:std_logic_vector(0 TO 3);
BEGIN
ulaz<=(D,C,B,A);
sklopka: entity work.mux161 PORT MAP (('0','0','1','0','0','0','0','0','1','1','0','1','1','1','0','1'),ulaz,'1',F);
END arch;

VJEŽBA 4- DEKODER (funkcija 8)

ENTITY dek12 IS PORT (
    A: IN std_logic;
    E: IN std_logic;
    y0: OUT std_logic;
    y1: OUT std_logic);
END dek12;

ARCHITECTURE arch OF dek12 IS
BEGIN
y0 <= E AND NOT A AFTER 10 ns;
Y1 <= E AND A AFTER 10 ns;
END arch;

ENTITY dek1 IS PORT (                          -- dekodersko stablo
          E: IN std_logic;
          A: IN std_logic_vector (0 TO 3);
          Y: OUT std_logic_vector (0 TO 15));
END dek1;

ARCHITECTURE arch OF dek1 IS
SIGNAL i: std_logic_vector (0 TO 13);
BEGIN
 C0: ENTITY work.dek12 PORT MAP (A(0), E, i(0), i(1));
 C1: ENTITY work.dek12 PORT MAP (A(1), i(0), i(2), i(3));
 C2: ENTITY work.dek12 PORT MAP (A(1), i(1), i(4), i(5));
 C3: ENTITY work.dek12 PORT MAP (A(2), i(2), i(6), i(7));
 C4: ENTITY work.dek12 PORT MAP (A(2), i(3), i(8), i(9));
 C5: ENTITY work.dek12 PORT MAP (A(2), i(4), i(10), i(11));
 C6: ENTITY work.dek12 PORT MAP (A(2), i(5), i(12), i(13));
```

```
 C7: ENTITY work.dek12 PORT MAP (A(3), i(6), Y(0), Y(1));
 C8: ENTITY work.dek12 PORT MAP (A(3), i(7), Y(2), Y(3));
 C9: ENTITY work.dek12 PORT MAP (A(3), i(8), Y(4), Y(5));
 C10: ENTITY work.dek12 PORT MAP (A(3), i(9), Y(6), Y(7));
 C11: ENTITY work.dek12 PORT MAP (A(3), i(10), Y(8), Y(9));
 C12: ENTITY work.dek12 PORT MAP (A(3), i(11), Y(10), Y(11));
 C13: ENTITY work.dek12 PORT MAP (A(3), i(12), Y(12), Y(13));
 C14: ENTITY work.dek12 PORT MAP (A(3), i(13), Y(14), Y(15));
END arch;

ENTITY fja8 IS PORT (                                   -- ostvarenje osme funkcije
                A: IN std_logic_VECTOR (0 TO 3);
              F: OUT std_logic);
END fja8;

ARCHITECTURE arch OF fja8 IS
COMPONENT dek1 PORT (
          E: IN std_logic;
          A: IN std_logic_vector (0 TO 3);
          Y: OUT std_logic_vector (0 TO 15));
           END COMPONENT;
SIGNAL izlaz : std_logic_vector(0 TO 15);

BEGIN
 sklop: dek1 PORT MAP ('1',A(0)=>A(0),A(1)=>A(1),A(2)=>A(2),A(3)=>A(3),Y=>izlaz);
  F <= izlaz(0) OR izlaz(2) OR izlaz(10) OR izlaz(11) OR izlaz(12) OR izlaz(13) OR izlaz(15);
END arch;


VJEŽBA 5

ENTITY dmux IS PORT (
     x: IN std_logic_vector (1 DOWNTO 0);
     y: IN std_logic_vector (1 DOWNTO 0);
     s: IN std_logic;
     z: OUT std_logic_vector (1 DOWNTO 0));
END dmux;

ARCHITECTURE arch OF dmux IS
BEGIN
z(1)<= ((not s and x(1)) OR (s and y(1))) AFTER 10 ns;
z(0)<= ((not s and x(0)) OR (s and y(0))) AFTER 10 ns;
END arch;

ENTITY FA IS PORT (
          A,B: IN std_logic_vector ( 1 downto 0);
          cin: IN std_logic;
          r: OUT std_logic_vector (1 downto 0);
          cout: Out std_logic);
END FA;

ARCHITECTURE arch OF FA IS
```

```
BEGIN
cout <= (A(1) AND B(1)) OR (NOT A(0) AND NOT B(0)) OR (cin AND(A(1) AND NOT B(0))) OR (cin
AND(B(1) AND NOT A(0)))OR (cin AND(A(1) AND NOT A(0))) OR (cin AND(B(1) AND NOT B(1)))AFTER
10 ns;
r(1) <= ((( A(1) XOR B(1)) AND NOT cin) OR (NOT( A(1) XOR B(1)) AND cin)) AFTER 10 ns;
r(0) <= ((not cin AND A(1) AND NOT A(0) AND B(1) AND B(0)) OR (not cin AND A(1) AND A(0) AND B(1)
AND NOT B(0)) OR (not cin AND A(0) AND NOT B(1) AND B(0)) OR (not cin AND NOT A(1) AND A(0)
AND B(0)) OR (not cin AND NOT A(1) AND NOT A(0)AND NOT B(0)) OR (not cin AND NOT A(0) AND
NOT B(1) AND NOT B(0))) OR  ((cin AND NOT A(1) AND NOT A(0) AND NOT B(1) AND NOT B(0)) OR
(cin AND A(0) AND NOT A(1) AND NOT B(1) AND B(0)) OR (cin AND A(0) AND B(1) AND NOT B(0)) OR
(cin AND NOT A(0) AND B(1) AND B(0)) OR (cin AND A(1) AND A(0) AND NOT B(0)) OR (cin AND A(1)
AND NOT A(0)AND B(0))) AFTER 10 ns;
END arch;

ENTITY b1kompl IS PORT (
      x: IN std_logic_vector (1 DOWNTO 0);
      y: OUT std_logic_vector (1 DOWNTO 0));
END b1kompl;

ARCHITECTURE arch OF b1kompl IS
BEGIN
 y(1)<= not x(1) AFTER 10 ns;
 y(0)<= not x(0) AFTER 10 ns;
END arch;

ENTITY primitiv IS PORT(
           a: IN std_logic_vector (1 DOWNTO 0);
           b: IN std_logic_vector (1 DOWNTO 0);
        cin, oper: IN std_logic;
        r: OUT  std_logic_vector (1 DOWNTO 0);
           cout: OUT std_logic);
END primitiv;

ARCHITECTURE arch OF primitiv IS
signal i,j:std_logic_vector (1 DOWNTO 0);
BEGIN
sklop1: ENTITY work.b1kompl PORT MAP(b(1 DOWNTO 0),i(1 DOWNTO 0));
sklop2: ENTITY  work.dmux PORT MAP (b(1 DOWNTO 0),i(1 DOWNTO 0),oper,j(1 DOWNTO 0));
sklop3: ENTITY work.FA PORT MAP (a(1 DOWNTO 0),j(1 DOWNTO 0),cin,r(1 DOWNTO 0),cout);
END arch;

ENTITY zbrajalo IS PORT (
     A: IN std_logic_vector (7 DOWNTO 0);
     B: IN std_logic_vector (7 DOWNTO 0);
     oper: IN std_logic;
      R: OUT std_logic_vector (7 DOWNTO 0);
      cout: OUT std_logic);
END zbrajalo;

ARCHITECTURE arch OF zbrajalo IS
   signal i1,i2,i3: std_logic;
BEGIN
```

```
c0: ENTITY work.primitiv PORT MAP (A(1 DOWNTO 0),B(1 DOWNTO 0),oper,oper,R(1 DOWNTO 0),i1);
c1: ENTITY work.primitiv PORT MAP (A(3 DOWNTO 2),B(3 DOWNTO 2),i1,oper,R(3 DOWNTO 2),i2);
C2: ENTITY work.primitiv PORT MAP (A(5 DOWNTO 4),B(5 DOWNTO 4),i2,oper,R(5 DOWNTO 4),i3);
c3: ENTITY work.primitiv PORT MAP (A(7 DOWNTO 6),B(7 DOWNTO 6),i3,oper,R(7 DOWNTO 6),cout);
END arch;
```

VJEŽBA 6

```
ENTITY asintff IS PORT(
    clk, t, clr, st: IN std_logic;
     q, qn: OUT std_logic);
END asintff;

ARCHITECTURE arch OF asintff IS
BEGIN
 PROCESS (clr,st,clk)
 VARIABLE mem:std_logic :='0';
 BEGIN
   IF clr='1' THEN mem:='0';
    ELSIF st='1' THEN mem:='1';
    ELSIF falling_edge(clk) THEN
      IF t='1' THEN mem:= not mem;
      END IF;
   END IF;
 q<= mem AFTER 10 ns;
 qn<= not mem AFTER 10 ns;
 END PROCESS;
  END arch;

ENTITY sintff IS PORT (
     clk, t, clr, st: IN std_logic;
     q, qn: OUT std_logic);
END sintff;

ARCHITECTURE arch OF sintff IS
BEGIN
 PROCESS (clk)
 VARIABLE mem:std_logic :='0';
 BEGIN
  IF falling_edge(clk) THEN
     IF clr='1' THEN mem:='0';
     ELSIF st='1' THEN mem:='1';
     ELSIF t='1' THEN mem:= not mem;
     END IF;
  END IF;
 q<= mem AFTER 10 ns;
 qn<= not mem AFTER 10 ns;
 END PROCESS;
  END arch;
```

```
ENTITY sekvsklop IS PORT (
   clk, reset, p :IN std_logic;
   q : OUT std_logic_vector (4 downto 0));
END sekvsklop;

ARCHITECTURE arch OF sekvsklop IS
SIGNAL i1,i2,i3,i4,i5 :std_logic;
SIGNAL qp:std_logic_vector(4 downto 0);
BEGIN
i1<= reset AND p;
i2<= reset AND not p;
i3<= qp(0) AND qp(1);
i4<= i3 AND qp(2);
i5<= i4 AND qp(3);
s1: ENTITY work.sintff PORT MAP(clk,'1',i2,i1,qp(0),open);
s2: ENTITY work.sintff PORT MAP(clk,qp(0),reset,'0',qp(1),open);
s3: ENTITY work.sintff PORT MAP(clk,i3,reset,'0',qp(2),open);
s4: ENTITY work.sintff PORT MAP(clk,i4,reset,'0',qp(3),open);
s5: ENTITY work.sintff PORT MAP(clk,i5,reset,'0',qp(4),open);
q<=qp;
END arch;
```

VJEŽBA 7

Sintff i sekv sklop iz 6 vježbe

```
ENTITY BROJILO IS PORT (
   cp,reset: IN std_logic;
   q: OUT std_logic_vector (4 downto 0));
END BROJILO;

ARCHITECTURE arch OF BROJILO IS
BEGIN
c1: ENTITY work.sekvsklop PORT MAP (cp,reset,'1',q);
END arch;

ENTITY TIMER IS PORT(
cp,reset,masreset: IN std_logic;
t2,t4,t8,t16,t32: OUT std_logic);
END TIMER;

ARCHITECTURE arch OF TIMER IS
signal i: std_logic;
signal q: std_logic_vector (4 downto 0);
BEGIN
i<= reset or masreset;
c1: ENTITY work.BROJILO PORT MAP (cp,i,q);
t2<= (q(0)and not q(1) and not q(2) and not q(3) and not q(4)) after 10 ns;
t4<= (q(0)and q(1) and not q(2) and not q(3) and not q(4)) after 10 ns;
t8<= (q(0)and q(1) and q(2) and not q(3) and not q(4)) after 10 ns;
t16<= (q(0)and q(1) and q(2) and q(3) and not q(4)) after 10 ns;
t32<= (q(0)and q(1) and q(2) and q(3) and q(4)) after 10 ns;END arch;
```

```
ENTITY automat_mealy IS PORT(
t2,t4,t8,t16,t32,cp,masreset: IN std_logic;
cr,cy,cg,wr,reset: OUT std_logic);
END automat_mealy;

ARCHITECTURE arch OF automat_mealy IS
CONSTANT S0:std_logic_vector(2 downto 0):= "000";
CONSTANT S1:std_logic_vector(2 downto 0):= "001";
CONSTANT S2:std_logic_vector(2 downto 0):= "010";
CONSTANT S3:std_logic_vector(2 downto 0):= "011";
CONSTANT S4:std_logic_vector(2 downto 0):= "100";
CONSTANT S5:std_logic_vector(2 downto 0):= "101";
CONSTANT S6:std_logic_vector(2 downto 0):= "110";
SIGNAL state_present, state_next: std_logic_vector(2 downto 0);
SIGNAL izlazi_next: STD_LOGIC_VECTOR (4 DOWNTO 0);

BEGIN
 PROCESS(t2,t4,t8,t16,t32, state_present)
 begin
 CASE state_present IS
   WHEN S0 => IF (t8='1') THEN state_next<=S1; izlazi_next <= "11001";
              ELSE state_next<= S0; izlazi_next <= "10000"; END IF;
   WHEN S1 => IF (t2='1') THEN state_next<=S2; izlazi_next <= "00101";
              ELSE state_next<= S1; izlazi_next <= "11000"; END IF;
   WHEN S2 => IF (t32='1') THEN state_next<=S3; izlazi_next <= "01001";
              ELSE state_next<= S2; izlazi_next <= "00100"; END IF;
   WHEN S3 => IF (t4='1') THEN state_next<=S4; izlazi_next <= "10001";
              ELSE state_next<= S3; izlazi_next <= "01000"; END IF;
   WHEN S4 => IF (t2='1') THEN state_next<=S5; izlazi_next <= "10011";
              ELSE state_next<= S4; izlazi_next <= "10000"; END IF;
   WHEN S5 => IF (t16='1') THEN state_next<=S6; izlazi_next <= "10001";
              ELSE state_next<= S5; izlazi_next <= "10010"; END IF;
  WHEN S6 => IF (t4='1') THEN state_next<=S1; izlazi_next <= "11001";
              ELSE state_next<= S6; izlazi_next <= "10000"; END IF;
              End case;
END process;

PROCESS( cp,masreset)
begin
IF falling_edge(cp) THEN
 IF(masreset='1') THEN   state_present<=S0; cr<='1' AFTER 10 ns; cy<='0' AFTER 10 ns; cg<='0' after 10
ns; wr<='0' after 10 ns; reset<='1' after 10 ns;
 ELSE  state_present<=state_next;
     cr<= izlazi_next(4) after 10 ns;
     cy<= izlazi_next(3) after 10 ns;
     cg<= izlazi_next(2) after 10 ns;
     wr<= izlazi_next(1) after 10 ns;
     reset<= izlazi_next(0) after 10 ns;
 END IF;END if;
END PROCESS;
END arch;
```

```
ENTITY upravljac_mealy IS PORT (
masreset, cp : IN std_logic;
cr, cy ,cg, wr : OUT std_logic);
END upravljac_mealy;

ARCHITECTURE arch OF upravljac_mealy IS
SIGNAL t2, t4, t8, t16, t32:std_logic;
SIGNAL reset:std_logic;
BEGIN
TIMER1: ENTITY work.TIMER PORT MAP (cp, reset, masreset, t2, t4, t8, t16, t32);
automat1: ENTITY work.automat_mealy PORT MAP(t2,t4,t8,t16,t32,cp,masreset,cr,cy,cg,wr,reset);

END arch;
```

```
ENTITY upravljac_moore IS PORT (
cp, masreset : IN STD_LOGIC;
cr,cy,cg,wr : OUT STD_LOGIC);
end upravljac_moore;

ARCHITECTURE arch OF upravljac_moore IS
SIGNAL reset : STD_LOGIC;
SIGNAL t2, t4, t8, t16, t32 : STD_LOGIC;
BEGIN
TIMER0 : ENTITY work.TIMER PORT MAP(cp, reset, masreset, t2, t4, t8, t16, t32);
AUTOMAT0 : ENTITY work.automat_moore PORT MAP(cp, masreset, t2, t4, t8, t16, t32, cr, cy, cg, wr,
reset);
END arch;
```

```
ENTITY automat_moore IS PORT (
cp,masreset,t2,t4,t8,t16,t32 : IN STD_LOGIC;
cr,cy,cg,wr,reset : OUT STD_LOGIC);
end automat_moore;

ARCHITECTURE arch OF automat_moore IS

SIGNAL state_present, state_next : STD_LOGIC_VECTOR(3 DOWNTO 0);
CONSTANT S0 : STD_LOGIC_VECTOR(3 DOWNTO 0) := "0000";
CONSTANT S1 : STD_LOGIC_VECTOR(3 DOWNTO 0) := "0001";
CONSTANT S2 : STD_LOGIC_VECTOR(3 DOWNTO 0) := "0010";
CONSTANT S3 : STD_LOGIC_VECTOR(3 DOWNTO 0) := "0011";
CONSTANT S4 : STD_LOGIC_VECTOR(3 DOWNTO 0) := "0100";
CONSTANT S5 : STD_LOGIC_VECTOR(3 DOWNTO 0) := "0101";
CONSTANT S6 : STD_LOGIC_VECTOR(3 DOWNTO 0) := "0110";
CONSTANT S7 : STD_LOGIC_VECTOR(3 DOWNTO 0) := "0111";
CONSTANT S8 : STD_LOGIC_VECTOR(3 DOWNTO 0) := "1000";
CONSTANT S9 : STD_LOGIC_VECTOR(3 DOWNTO 0) := "1001";
CONSTANT S10 : STD_LOGIC_VECTOR(3 DOWNTO 0) := "1010";
CONSTANT S11 : STD_LOGIC_VECTOR(3 DOWNTO 0) := "1011";
CONSTANT S12 : STD_LOGIC_VECTOR(3 DOWNTO 0) := "1100";

BEGIN
PROCESS(t2, t4, t8, t16, t32, state_present)
BEGIN
CASE state_present IS
    WHEN S0 => IF (t8 = '1') THEN state_next <= S1;
    ELSE state_next <= S0;
    END IF;
    WHEN S1 => IF (t2 = '1') THEN state_next <= S3;
    ELSE state_next <= S2;
    END IF;
    WHEN S2 => IF (t2 = '1') THEN state_next <= S3;
    ELSE state_next <= S2;
    END IF;
    WHEN S3 => IF (t32 = '1') THEN state_next <= S5;
    ELSE state_next <= S4;
    END IF;
    WHEN S4 => IF (t32 = '1') THEN state_next <= S5;
    ELSE state_next <= S4;
    END IF;
   WHEN S5 => IF (t4 = '1') THEN state_next <= S7;
   ELSE state_next <= S6;
   END IF;
   WHEN S6 => IF (t4 = '1') THEN state_next <= S7;
   ELSE state_next <= S6;
   END IF;
   WHEN S7 => IF (t2 = '1') THEN state_next <= S9;
   ELSE state_next <= S8;
    END IF;
   WHEN S8 => IF (t2 = '1') THEN state_next <= S9;
   ELSE state_next <= S8;
```

```
   END IF;
   WHEN S9 => IF (t16 = '1') THEN state_next <= S11;
   ELSE state_next <= S10;
   END IF;
   WHEN S10 => IF (t16 = '1') THEN state_next <= S11;
   ELSE state_next <= S10;
   END IF;
   WHEN S11 => IF (t4 = '1') THEN state_next <= S1;
   ELSE state_next <= S12;
   END IF;
   WHEN S12 => IF (t4 = '1') THEN state_next <= S1;
   ELSE state_next <= S12;
   END IF;
  WHEN OTHERS => IF (t8 = '1') THEN state_next <= S1;
  ELSE state_next <= S0;
  END IF;
 END CASE;
 END PROCESS;

PROCESS(state_present)
BEGIN
CASE state_present IS
WHEN S0 => cr <= '1' AFTER 10 ns; cy <= '0' AFTER 10 ns; cg <= '0' AFTER 10 ns; wr <= '0' AFTER 10 ns;
reset <= '0' AFTER 10 ns;
WHEN S1 => cr <= '1' AFTER 10 ns; cy <= '1' AFTER 10 ns; cg <= '0' AFTER 10 ns; wr <= '0' AFTER 10 ns;
reset <= '1' AFTER 10 ns;
WHEN S2 => cr <= '1' AFTER 10 ns; cy <= '1' AFTER 10 ns; cg <= '0' AFTER 10 ns; wr <= '0' AFTER 10 ns;
reset <= '0' AFTER 10 ns;
WHEN S3 => cr <= '0' AFTER 10 ns; cy <= '0' AFTER 10 ns; cg <= '1' AFTER 10 ns; wr <= '0' AFTER 10 ns;
reset <= '1' AFTER 10 ns;
WHEN S4 => cr <= '0' AFTER 10 ns; cy <= '0' AFTER 10 ns; cg <= '1' AFTER 10 ns; wr <= '0' AFTER 10 ns;
reset <= '0' AFTER 10 ns;
WHEN S5 => cr <= '0' AFTER 10 ns; cy <= '1' AFTER 10 ns; cg <= '0' AFTER 10 ns; wr <= '0' AFTER 10 ns;
reset <= '1' AFTER 10 ns;
WHEN S6 => cr <= '0' AFTER 10 ns; cy <= '1' AFTER 10 ns; cg <= '0' AFTER 10 ns; wr <= '0' AFTER 10 ns;
reset <= '0' AFTER 10 ns;
WHEN S7 => cr <= '1' AFTER 10 ns; cy <= '0' AFTER 10 ns; cg <= '0' AFTER 10 ns; wr <= '0' AFTER 10 ns;
reset <= '1' AFTER 10 ns;
WHEN S8 => cr <= '1' AFTER 10 ns; cy <= '0' AFTER 10 ns; cg <= '0' AFTER 10 ns; wr <= '0' AFTER 10 ns;
reset <= '0' AFTER 10 ns;
WHEN S9 => cr <= '1' AFTER 10 ns; cy <= '0' AFTER 10 ns; cg <= '0' AFTER 10 ns; wr <= '1' AFTER 10 ns;
reset <= '1' AFTER 10 ns;
WHEN S10 => cr <= '1' AFTER 10 ns; cy <= '0' AFTER 10 ns; cg <= '0' AFTER 10 ns; wr <= '1' AFTER 10
ns; reset <= '0' AFTER 10 ns;
WHEN S11 => cr <= '1' AFTER 10 ns; cy <= '0' AFTER 10 ns; cg <= '0' AFTER 10 ns; wr <= '0' AFTER 10
ns; reset <= '1' AFTER 10 ns;
WHEN S12 => cr <= '1' AFTER 10 ns; cy <= '0' AFTER 10 ns; cg <= '0' AFTER 10 ns; wr <= '0' AFTER 10
ns; reset <= '0' AFTER 10 ns;
WHEN OTHERS => cr <= '1' AFTER 10 ns; cy <= '0' AFTER 10 ns; cg <= '0' AFTER 10 ns; wr <= '0' AFTER
10 ns; reset <= '0' AFTER 10 ns;
END CASE;
END PROCESS;
```

```
PROCESS(cp, masreset)
BEGIN
IF falling_edge(cp) THEN
IF (masreset = '1') THEN
state_present <= S0;
ELSE
state_present <= state_next;
END IF;
END IF;
END PROCESS;
END arch;
```