**FA:**

ENTITY FA IS PORT (

a: IN STD\_LOGIC\_VECTOR(1 DOWNTO 0);

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

r(1) <= (( a(1) and not a(0) and b(1) and not cin) or

( a(1) and b(1) and b(0) and cin) or

(a(1) and a(0) and b(1) and not b(0)) or

(not a(1) and not a(0) and not b(1) and not cin) or

(not a(1) and not b(1) and b(0) and cin) or

(not a(1) and a(0) and not b(1) and not b(0)) or

(not a(1) and a(0) and b(1) and b(0) and not cin) or

(not a(1) and not a(0) and b(1) and not b(0) and cin) or

(a(1) and a(0) and not b(1) and b(0) and not cin) or

(a(1) and not a(0) and not b(1) and not b(0) and cin)) after 10 ns;

r(0) <= ((a(0) and not b(0) and not cin) or

(a(0) and b(0) and cin) or

(not a(0) and b(0) and not cin) or

(not a(0) and not b(0) and cin)) after 10 ns;

cout <= ((not a(1) and not b(1) and not b(0) and cin) or

(not a(1) and not a(0) and cin ) or

(a(0) and b(1) and b(0)) or

(not a(1) and b(1) and b(0)) or

(a(1) and a(0) and cin) or

(not a(1) and not a(0) and not b(1) and not b(0)) or

(a(1) and a(0) and not b(1)) or

(b(1) and b(0) and cin)) after 10 ns;

END arch;

**B1-Komplement**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

ENTITY b\_1komplement IS PORT (

b: IN STD\_LOGIC\_VECTOR(1 DOWNTO 0);

y: OUT STD\_LOGIC\_VECTOR(1 DOWNTO 0)

);

END b\_1komplement;

ARCHITECTURE arch OF b\_1komplement IS

BEGIN

y(1) <= (b(1)) after 10 ns;

y(0) <= (not b(0)) after 10 ns;

END arch;

**DMUX:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

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) <= ((x(1) and not s) or (y(1) and s)) after 10 ns;

z(0) <= ((x(0) and not s) or (y(0) and s)) after 10 ns;

END arch;

**Primitiv:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

ENTITY primitiv IS PORT (

a: IN STD\_LOGIC\_VECTOR(1 DOWNTO 0);

b: IN STD\_LOGIC\_VECTOR(1 DOWNTO 0);

cin: IN STD\_LOGIC;

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

KOMPLEMENT: ENTITY WORK.b\_1komplement PORT MAP(b(1 DOWNTO 0), i(1 DOWNTO 0));

MUX: ENTITY WORK.dmux PORT MAP(b(1 DOWNTO 0),i(1 DOWNTO 0), oper, j(1 DOWNTO 0));

ADDER: ENTITY WORK.FA PORT MAP(a(1 DOWNTO 0), j(1 DOWNTO 0), cin, r(1 DOWNTO 0), cout);

END arch;

**Zbrajalo:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

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 i: STD\_LOGIC\_VECTOR(2 DOWNTO 0);

BEGIN

ADD0: ENTITY WORK.primitiv PORT MAP(a(1 DOWNTO 0), b(1 DOWNTO 0), oper, oper, r(1 DOWNTO 0), i(0));

ADD1: ENTITY WORK.primitiv PORT MAP(a(3 DOWNTO 2), b(3 DOWNTO 2), i(0), oper, r(3 DOWNTO 2), i(1));

ADD2: ENTITY WORK.primitiv PORT MAP(a(5 DOWNTO 4), b(5 DOWNTO 4), i(1), oper, r(5 DOWNTO 4), i(2));

ADD3: ENTITY WORK.primitiv PORT MAP(a(7 DOWNTO 6), b(7 DOWNTO 6), i(2), oper, r(7 DOWNTO 6), cout);

END arch;