library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

use ieee.std\_logic\_arith.all;

use ieee.numeric\_std.all;

entity alu is

port( A,B: in std\_logic\_vector(3 downto 0);

Sel: in std\_logic\_vector(2 downto 0);

Cin: in std\_logic;

Output: out std\_logic\_vector(3 downto 0);

Cout: out std\_logic);

end alu;

architecture alu1 of alu is

signal OUTPUT\_INTEGER: integer:=0;

begin

process(a,b,sel,cin)

variable forIn, forA, forB, forS: std\_logic\_vector(4 downto 0);

variable prod: std\_logic\_vector(7 downto 0);

variable intA, intB, intOUT: integer;

begin

if sel="000" then --add

forIn:="00000";

forIn(0):= Cin;

forA(4):='0';

forA(3 downto 0):=a;

forB(4):='0';

forB(3 downto 0):=b;

forS:=forA+forB+forIn;

Output<=forS(3 downto 0);

Cout<=forS(4);

elsif sel="001" then --scadere

forIn:="00000";

forIn(0):= Cin;

forA(4):='1';

forA(3 downto 0):=a;

forB(4):='0';

forB(3 downto 0):=b;

forS:=forA-forB-forIn;

Output<=forS(3 downto 0);

Cout<=forS(4);

elsif sel="010" then -- inmultire

Prod:=A\*B;

Output<=Prod(3 downto 0);

elsif sel="011" then --impartire

intA:=conv\_integer(A);

intB:=conv\_integer(B);

intOUT:= intA / intB;

output\_integer <= intOUT;

elsif sel="100" then -- and

Output<=A and B;

elsif Sel = "101" then -- NOT

Output <= not (A);

elsif Sel = "110" then -- OR

Output <= A or B;

elsif Sel = "100" then -- XOR

Output <= A xor B;

end if;

end process;

end alu1;