library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity processor is

    Port ( a : in STD\_LOGIC\_VECTOR (3 downto 0);

           b : in STD\_LOGIC\_VECTOR (3 downto 0);

           o : out STD\_LOGIC\_VECTOR (3 downto 0);

           s: in STD\_LOGIC\_VECTOR (5 downto 0);

           c0 : in STD\_LOGIC;

           c4 : out STD\_LOGIC);

end processor;

architecture Behavioral of processor is

        signal c3, c2, c1: std\_logic;

        signal x: STD\_LOGIC\_VECTOR (3 downto 0);

        signal y: STD\_LOGIC\_VECTOR (3 downto 0);

        signal and\_op: STD\_LOGIC\_VECTOR (3 downto 0);

        signal add\_op: STD\_LOGIC\_VECTOR (3 downto 0);

    begin

        x(0) <= (a(0) and s(4)) xor s(2);

        x(1) <= (a(1) and s(4)) xor s(2);

        x(2) <= (a(2) and s(4)) xor s(2);

        x(3) <= (a(3) and s(4)) xor s(2);

        y(0) <= (b(0) and s(3)) xor s(1);

        y(1) <= (b(1) and s(3)) xor s(1);

        y(2) <= (b(2) and s(3)) xor s(1);

        y(3) <= (b(3) and s(3)) xor s(1);

        and\_op(0) <= a(0) and b(0);

        and\_op(1) <= a(1) and b(1);

        and\_op(2) <= a(2) and b(2);

        and\_op(3) <= a(3) and b(3);

        -- The first full adder

        add\_op(0) <= x(0) xor y(0) xor c0;

        c1 <= ( (x(0) xor y(0)) and c0 ) or (x(0) and y(0));

        -- The second full adder

        add\_op(1) <= x(1) xor y(1) xor c1;

        c2 <= ( (x(1) xor y(1)) and c1 ) or (x(1) and y(1));

        -- The third full adder

        add\_op(2) <= x(2) xor y(2) xor c2;

        c3 <= ( (x(2) xor y(2)) and c2 ) or (x(2) and y(2));

        -- The fourth full adder

        add\_op(3) <= x(3) xor y(3) xor c3;

        c4 <= ( (x(3) xor y(3)) and c3 ) or (x(3) and y(3));

        o(0) <= add\_op(0) when (s(5) = '1') else and\_op(0);

        o(1) <= add\_op(1) when (s(5) = '1') else and\_op(1);

        o(2) <= add\_op(2) when (s(5) = '1') else and\_op(2);

        o(3) <= add\_op(3) when (s(5) = '1') else and\_op(3);

end Behavioral;