**Case Study**

**Modul 6: for loop**

**Kode Multiplier**

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.numeric\_std.all;

entity MultiplierDewa is

    generic (

        n : positive := 4

    );

    port (

        dataA, dataB : IN std\_logic\_vector (n - 1 downto 0);

        dataOut : OUT std\_logic\_vector (2\*n - 1 downto 0)

    );

end entity MultiplierDewa;

architecture rtl of *MultiplierDewa* is

begin

    shift: process(dataA, dataB)

        variable loopLen : integer := 0;

        variable andTemp : std\_logic := '0';

        variable dataTemp : std\_logic\_vector (2\*n - 1 downto 0) := (others => '0');

    begin

        -- dataTemp := dataIn;

        loopLen := n - 1;

        dataTemp := (others => '0');

        for i in 0 to loopLen loop

            for j in 0 to loopLen loop

                dataTemp(i+j)  := dataTemp(i+j) xor (dataA(i) AND dataB(j));

                dataTemp(i+j) := dataTemp(i+j) xor andTemp;

            end loop;

        end loop;

        dataOut <= dataTemp;

    end process shift;

end architecture rtl;

A screenshot of a computer

Description automatically generated

**Kode Test Bench**

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.numeric\_std.all;

entity MultiplierDewa\_tb is

end entity MultiplierDewa\_tb;

architecture rtl of *MultiplierDewa\_tb* is

    constant n : positive := 8;

    signal dataA, dataB : std\_logic\_vector (n - 1 downto 0);

    signal dataOut: std\_logic\_vector (2\*n - 1 downto 0) := (others => '0');

    -- constant DELAY : time := 100 ps;

begin

    UUT : entity work.MultiplierDewa

        GENERIC MAP (*n* => n)

        port map (*dataA* => dataA, *dataB* => dataB, *dataOut* => dataOut);

    test1: process

    begin

        dataA <= "00101011";

        dataB <= "11010010";

        assert dataOut = "0001110110100110" report "jawaban salah" severity error;

        wait for 100 ps;

        dataA <= "11001100";

        dataB <= "01010101";

        assert dataOut = "0011110000111100" report "jawaban salah" severity error;

        wait for 100 ps;

        dataA <= "11110000";

        dataB <= "00001111";

        assert dataOut = "0000010101010000" report "jawaban salah" severity error;

        wait for 100 ps;

        dataA <= "00000001";

        dataB <= "11111111";

        assert dataOut = "0000000011111111" report "jawaban salah" severity error;

        wait for 100 ps;

        dataA <= "10101010";

        dataB <= "01010101";

        assert dataOut = "0010001000100010" report "jawaban salah" severity error;

        wait for 100 ns;

        wait;

    end process test1;

end architecture rtl;

A table with numbers and symbols

Description automatically generated

A computer screen with multiple lines

Description automatically generated with medium confidence

library IEEE;

use IEEE.std\_logic\_1164.all;

use IEEE.numeric\_std.all;

entity MultiplierDewa\_tb is

end entity MultiplierDewa\_tb;

architecture rtl of *MultiplierDewa\_tb* is

    constant n : positive := 16;

    signal dataA, dataB : std\_logic\_vector (n - 1 downto 0);

    signal dataOut: std\_logic\_vector (2\*n - 1 downto 0) := (others => '0');

    -- constant DELAY : time := 100 ps;

begin

    UUT : entity work.MultiplierDewa

        GENERIC MAP (*n* => n)

        port map (*dataA* => dataA, *dataB* => dataB, *dataOut* => dataOut);

    test1: process

    begin

        dataA <= "1101101010110101";

        dataB <= "0101010101010101";

        assert dataOut = "00111000100011101100011101110001" report "jawaban salah" severity error;

        wait for 100 ps;

        dataA <= "1111111111111111";

        dataB <= "0000000000000001";

        assert dataOut = "00000000000000001111111111111111" report "jawaban salah" severity error;

        wait for 100 ps;

        dataA <= "0000111100001111";

        dataB <= "1111000011110000";

        assert dataOut = "00000101010100000000010101010000" report "jawaban salah" severity error;

        wait for 100 ps;

        dataA <= "0000000000000000";

        dataB <= "1111111111111111";

        assert dataOut = "00000000000000000000000000000000" report "jawaban salah" severity error;

        wait for 100 ps;

        dataA <= "1010101010101010";

        dataB <= "0101010101010101";

        assert dataOut = "00100010001000100010001000100010" report "jawaban salah" severity error;

        wait for 100 ns;

        wait;

    end process test1;

end architecture rtl;

A screenshot of a computer code

Description automatically generated