|  |  |  |  |
| --- | --- | --- | --- |
| **Nama** | Edgrant Henderson Suryajaya | **Kode Asisten** | ………………………… |
| **NPM** | 2206025016 | **Jenis Tugas** | TP/CS…………………. |

**Jawaban**

1. Code

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

use IEEE.MATH\_REAL.ALL;

entity CS\_RF\_PSD8\_EdgrantHendersonSuryajaya\_2206025016 is

    port (

        CLK : in std\_logic;

        dataIn : in std\_logic\_vector (15 downto 0);

        key : in std\_logic\_vector (3 downto 0);

        mode : in std\_logic;

        enable : in std\_logic;

        dataOut : out std\_logic\_vector (15 downto 0)

    );

end CS\_RF\_PSD8\_EdgrantHendersonSuryajaya\_2206025016;

architecture rtl of *CS\_RF\_PSD8\_EdgrantHendersonSuryajaya\_2206025016* is

    type state is (SINIT, SADD, SXOR, SSWAP, SSUB, SDONE);

    signal currentState, nextState : state := SINIT;

    -- init

    function fInit(oldKey : std\_logic\_vector) return std\_logic\_vector  is

    begin

        return oldKey & oldKey & oldKey & oldKey;

    end function;

    -- add

    function fAdd(data, newKey : std\_logic\_vector ) return std\_logic\_vector  is

        variable oldKey : unsigned (3 downto 0) := unsigned(newKey(3 downto 0));

        variable block0 : unsigned (3 downto 0) := unsigned(data(3 downto 0));

        variable block1 : unsigned (3 downto 0) := unsigned(data(7 downto 4));

        variable block2 : unsigned (3 downto 0) := unsigned(data(11 downto 8));

        variable block3 : unsigned (3 downto 0) := unsigned(data(15 downto 12));

    begin

        block0 := block0 + oldKey;

        block1 := block1 + oldKey;

        block2 := block2 + oldKey;

        block3 := block3 + oldKey;

        return std\_logic\_vector(block3 & block2 & block1 & block0);

    end function;

    -- xor

    function fXor(data, newKey : std\_logic\_vector ) return std\_logic\_vector  is

    begin

        return data xor newKey;

    end function;

    -- swap

    function fSwap(data: std\_logic\_vector ) return std\_logic\_vector  is

    begin

        return data(7 downto 0) & data(15 downto 8);

    end function;

    -- subtract

    function fSub(data, newKey : std\_logic\_vector) return std\_logic\_vector  is

        variable oldKey : unsigned (3 downto 0) := unsigned(newKey(3 downto 0));

        variable block0 : unsigned (3 downto 0) := unsigned(data(3 downto 0));

        variable block1 : unsigned (3 downto 0) := unsigned(data(7 downto 4));

        variable block2 : unsigned (3 downto 0) := unsigned(data(11 downto 8));

        variable block3 : unsigned (3 downto 0) := unsigned(data(15 downto 12));

    begin

        block0 := block0 - oldKey;

        block1 := block1 - oldKey;

        block2 := block2 - oldKey;

        block3 := block3 - oldKey;

        return std\_logic\_vector(block3 & block2 & block1 & block0);

    end function;

    -- Done

    function fDone(data : std\_logic\_vector) return std\_logic\_vector  is

    begin

        return data;

    end function;

    signal tempData : std\_logic\_vector (15 downto 0);

    signal longKey : std\_logic\_vector (15 downto 0);

begin

    -- Mengupdate state setiap clocktick

    clock\_upadte: process(clk)

    begin

        if rising\_edge(clk) then

            currentState <= nextState;

        end if;

    end process clock\_upadte;

    -- state is (SINIT, SADD, SXOR, SSWAP, SSUB, SDONE);

    -- Mengupdate state

    state\_update: process

    begin

        case currentState is

            when SINIT =>

                tempData <= dataIn;

                nextState <= SADD;

                if enable = '1' then

                    longKey <= fInit(key);

                end if;

            when SADD =>

                tempData <= fAdd(tempData, longKey);

                if mode = '0' then

                    nextState <= SXOR;

                else

                    nextState <= SSWAP;

                end if;

            when SXOR =>

                tempData <= fXor(tempData, longKey);

                if mode = '0' then

                    nextState <= SSWAP;

                else

                    nextState <= SSUB;

                end if;

            when SSWAP =>

                tempData <= fSwap(tempData);

                if mode = '0' then

                    nextState <= SSUB;

                else

                    nextState <= SXOR;

                end if;

            when SSUB =>

                nextState <= SDONE;

                tempData <= fSub(tempData, longKey);

            when SDONE =>

                nextState <= SINIT;

                dataOut <= fDone(tempData);

        end case;

        wait until falling\_edge(clk);

    end process state\_update;

end rtl;

1. Simulasi.

Input 1001010001110101 output 1011100101011000 key 0100

A screenshot of a computer

Description automatically generated

Input 1011100101011000 output 1001010001110101 key 0100

A screen shot of a computer

Description automatically generated

Input 0001001001001000 output 0100100000010010 key 0000

A screen shot of a computer

Description automatically generated

Input 0100100000010010 output 0001001001001000 key 0000

A screen shot of a computer

Description automatically generated

1. **Urutan state untuk dekripsi**

INIT 🡪 ADD 🡪 SWAP 🡪 XOR 🡪 SUB 🡪 DONE 🡪 INIT

1. **Apakah hasil output dekripsi sama dengan input enkripsi?**

Sama, karena urutan dekripsi adalah kebalikan dari enkripsi, enkripsi dibalik dari akhir, jadinya

* DONE 🡪 SUB 🡪 SWAP 🡪 XOR 🡪 ADD 🡪 INIT.

Done dan Init tidak melakukan operasi, jadi urutannya tetap sama, proses jadinya

* INIT 🡪 SUB 🡪 SWAP 🡪 XOR 🡪 ADD 🡪 DONE.

XOR dan SWAP adalah operasi simetris, artinya kalau dilakukan 2 kali akan balik ke nilai awal. Namun add dan sub tidak simetris, jika ingin balik ke nilai awal, sub harus diganti ke add dan add harus ganti ke sub. Jadinya proses jadinya

* INIT 🡪 ADD 🡪 SWAP 🡪 XOR 🡪 SUB 🡪 DONE.

Karena hal tersebut, dekripsi nilai hasil enkripsi akan mendapatkan nilai awal.