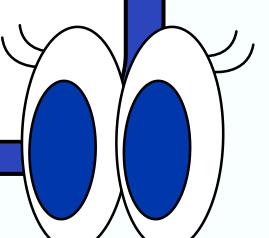
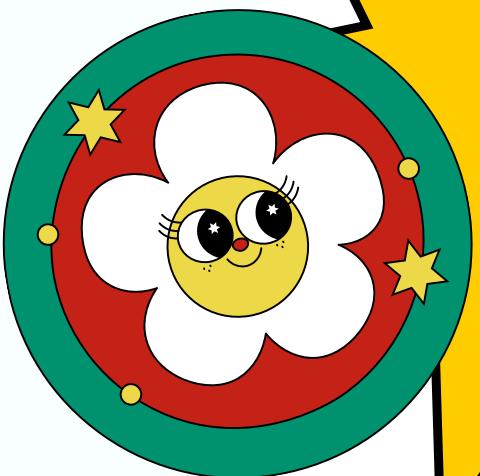
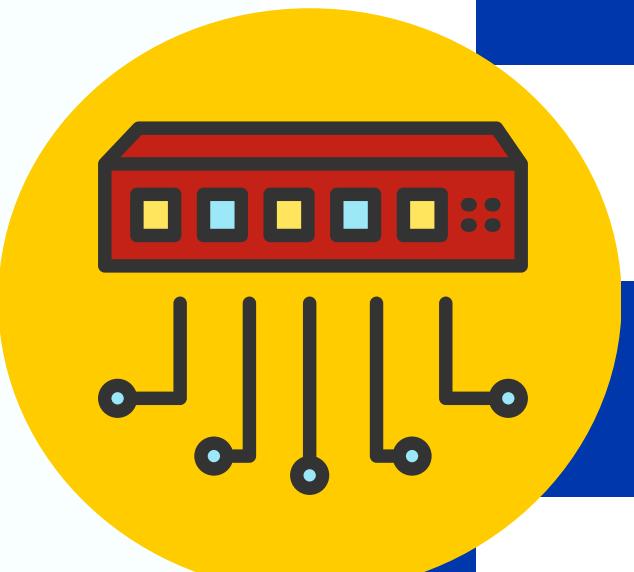


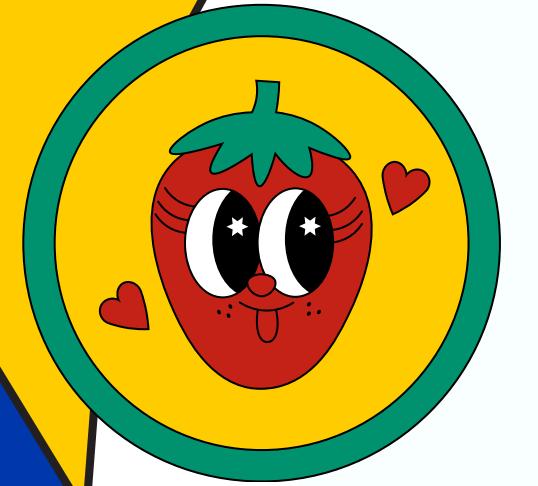
FINAL PROJECT  
PSD 2024

# NETWORK SWITCHING SYSTEM DESIGN VHDL



Kelompok PA 09





# **ANGGOTA**

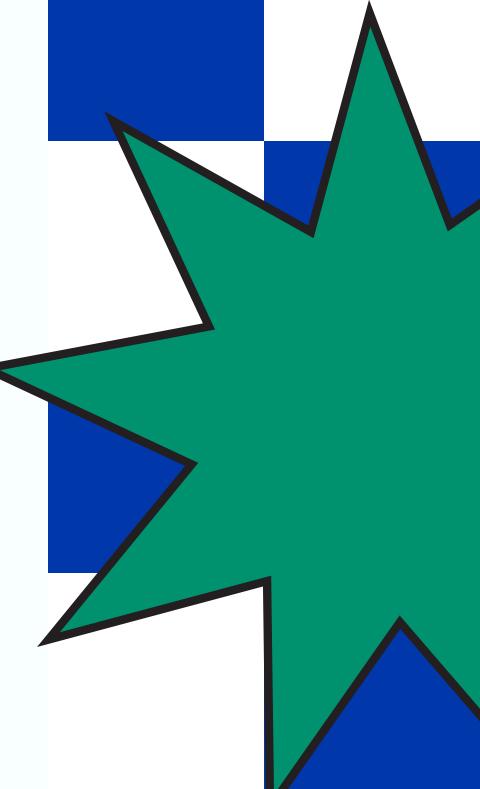


**ABEDNEGO ZEBUA (2306161883)**

**GRACE YUNIKE MARGARETHA S. (2306267031)**

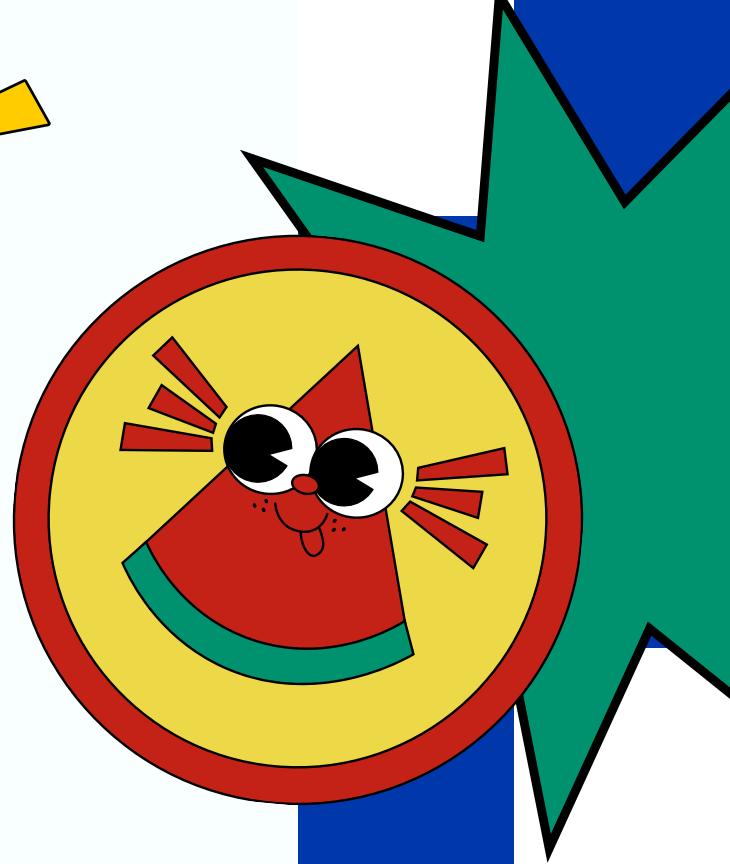
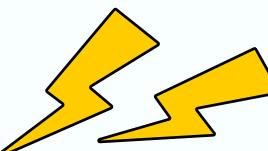
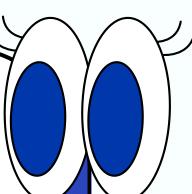
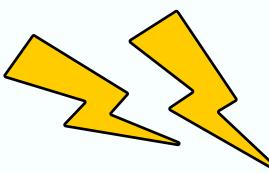
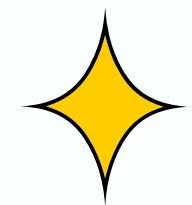
**NAUFAL HADI RASIKHIN (2306231366)**

**DIMAS ANANDA SUTIARDI (2306250586)**





# LATAR BELAKANG



## LATAR BELAKANG

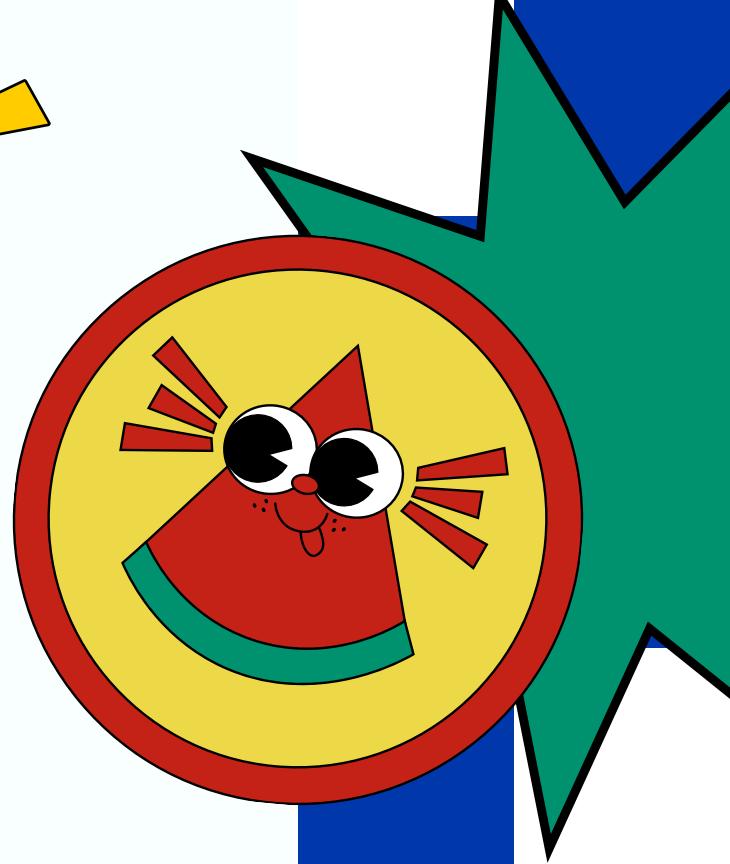
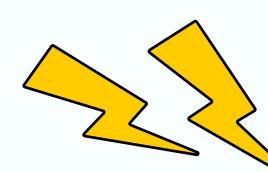
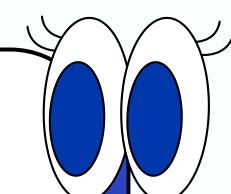
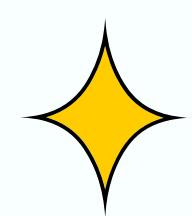
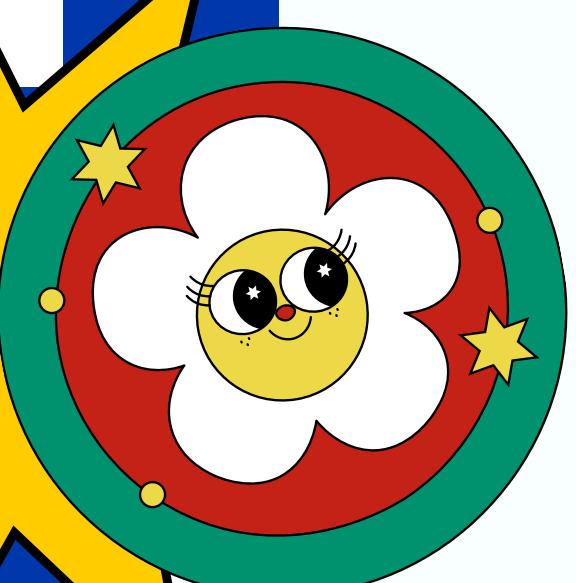
Setelah mempelajari lebih dalam mengenai Jaringan Komputer, kami menemukan bahwa switch merupakan perangkat penting dalam ekosistem jaringan, baik skala kecil maupun besar. Switch adalah perangkat digital yang bekerja secara sekuensial untuk memfilter dan meneruskan paket antar segmen LAN berdasarkan alamat MAC, sehingga berperan penting dalam memastikan efisiensi komunikasi jaringan. Namun, switch berbasis perangkat lunak sering kali terbatas dalam memproses data secara real-time, terutama pada jaringan yang kompleks. Berdasarkan hal tersebut, kelompok kami merasa bahwa replikasi logika kerja perangkat ini menggunakan VHDL (VHSIC Hardware Description Language) dapat menjadi solusi. Proyek ini mengimplementasikan FSM, looping construct, dan microprogramming untuk menciptakan Network Switching System berbasis perangkat keras yang lebih efisien dan mendukung kebutuhan jaringan yang semakin berkembang.

# **DESKRIPSI PROYEK**

## DESKRIPSI PROYEK

Project ini bertujuan untuk membuat replika perangkat keras dari sistem switch pada jaringan komputer dengan menggunakan VHDL. Switch yang didesain memiliki kemampuan menerima paket dari beberapa input port yang dimilikinya, kemudian akan menentukan alamat pengiriman packet tersebut berdasarkan mac-address tujuan dalam MAC table yang dimilikinya. Model desain akan mereplikasi switch yang memiliki sistem buffer, sehingga menggunakan structural style untuk penggunaan registers.

Design akan berupa FSM dengan Microprogramming yang akan berperan sebagai processor dari Switch tersebut. Loop construct akan digunakan didalam Testbench bersamaan dengan Procedure/Functions untuk mereplikasi contoh input berupa paket yang akan diterima oleh switch.



# **TUJUAN**

# TUJUAN

1.

MEMBUAT REPLIKA PERANGKAT KERAS SWITCH JARINGAN KOMPUTER MENGGUNAKAN VHDL

2.

MEMODELKAN LOGIKA PEMROSESAN PAKET DATA BERDASARKAN MAC ADDRESS MENGGUNAKAN FSM DAN MICROPROGRAMMING

3.

MENGIMPLEMENTASIKAN SISTEM BUFFER MENGGUNAKAN STRUCTURAL STYLE UNTUK PEMROSESAN DATA

4.

MENGINTEGRASIKAN PERANGKAT KERAS DAN LUNAK

# EQUIPMENT



# EQUIPMENT

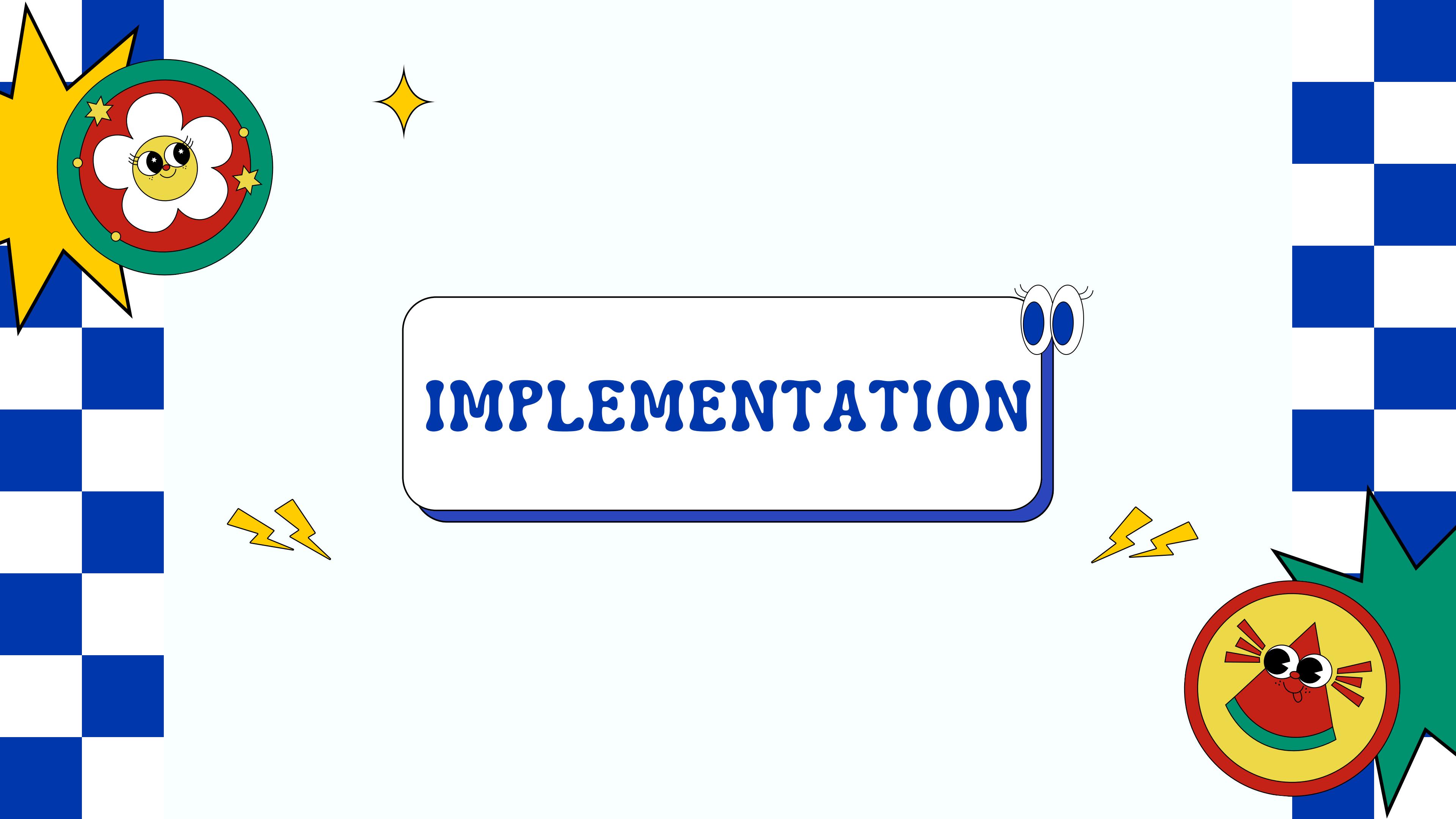
VISUAL STUDIO CODE

MODELSIM

GITHUB

QUARTUS

# IMPLEMENTATION



# SWITCH

Komponen ini menjadi switch jaringan sederhana yang mengelola data menggunakan frame encoder, decoder, dan tabel MAC (SwCAM). Komponen seperti Switchport menangani port data, sementara state machine mengatur alur pengiriman frame dari pembacaan hingga distribusi, baik secara langsung ke port tujuan atau broadcast jika alamat tujuan tidak ditemukan.

```
ENTITY switch IS
  PORT (
    clk : IN STD_LOGIC;
    reset : IN STD_LOGIC;
    rw_bit_in : IN STD_LOGIC; --for SwCAM
    sent_frame : OUT STD_LOGIC_VECTOR(167 DOWNTO 0); --Assuming 168-bit frame
    src_port : OUT STD_LOGIC_VECTOR(3 DOWNTO 0); -- from decoder
    src_mac : OUT STD_LOGIC_VECTOR(47 DOWNTO 0); -- from decoder
    dest_port : OUT STD_LOGIC_VECTOR(3 DOWNTO 0); -- from decoder
    dest_mac : OUT STD_LOGIC_VECTOR(47 DOWNTO 0); -- from decoder
    output_payload : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    --fa01
    fa01_MAC : IN STD_LOGIC_VECTOR(47 DOWNTO 0);
    fa01_InoutBit : IN STD_LOGIC; --indicator for port receiving = "0", sending = "1";
    fa01_Payload : IN STD_LOGIC_VECTOR(7 DOWNTO 0); --data to send
    fa01_DataIn : OUT STD_LOGIC_VECTOR(7 DOWNTO 0); --data that received
    fa01_DestMac : IN STD_LOGIC_VECTOR(47 DOWNTO 0);
    fa01_FrameOut : OUT STD_LOGIC_VECTOR(167 DOWNTO 0);
    --fa02
    fa02_MAC : IN STD_LOGIC_VECTOR(47 DOWNTO 0);
    fa02_InoutBit : IN STD_LOGIC;
    fa02_Payload : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    fa02_DataIn : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    fa02_DestMac : IN STD_LOGIC_VECTOR(47 DOWNTO 0);
    fa02_FrameOut : OUT STD_LOGIC_VECTOR(167 DOWNTO 0);
    --faa03
    fa03_MAC : IN STD_LOGIC_VECTOR(47 DOWNTO 0);
    fa03_InoutBit : IN STD_LOGIC;
    fa03_Payload : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    fa03_DataIn : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
    fa03_DestMac : IN STD_LOGIC_VECTOR(47 DOWNTO 0);
    fa03_FrameOut : OUT STD_LOGIC_VECTOR(167 DOWNTO 0)
    --add ports later
  );
END ENTITY switch;

ARCHITECTURE rtl OF switch IS
  COMPONENT switchport IS
    PORT (
      clk : IN STD_LOGIC;
      port_id : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
      frame_out : OUT STD_LOGIC_VECTOR(167 DOWNTO 0); -- Changed to OUT
      inout_bit : IN STD_LOGIC; -- generate frame when "1", 0 is idle or read mode
      data_in : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
      payload : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
      MAC_dest : IN STD_LOGIC_VECTOR(47 DOWNTO 0);
      MAC_add : IN STD_LOGIC_VECTOR(47 DOWNTO 0)
    );
  
```

# SWITCHPORT

Komponen yang bertugas mengirim maupun menerima data frame. Ketika menerima, dapat melakukan decode frame yang diterima tersebut. sedangkan ketika mengirim, dapat menyusun data-data yang ada menjadi sebuah frame dengan bantuan frame\_encoder

```
ENTITY switchport IS
  PORT (
    clk : IN STD_LOGIC;
    port_id : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
    frame_out : OUT STD_LOGIC_VECTOR(167 DOWNTO 0); -- Changed to OUT
    inout_bit : IN STD_LOGIC; -- generate frame when "1", 0 is idle or read mode
    data_in : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    payload : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    MAC_dest : IN STD_LOGIC_VECTOR(47 DOWNTO 0);
    MAC_add : IN STD_LOGIC_VECTOR(47 DOWNTO 0)
  );
END ENTITY switchport;

ARCHITECTURE rtl OF switchport IS
  CONSTANT total_port : INTEGER := 12;
  COMPONENT frame_encoder IS -- frame encoder
    PORT (
      frame_out : OUT STD_LOGIC_VECTOR(167 DOWNTO 0); -- Assuming 168-bit frame
      dest_mac : IN STD_LOGIC_VECTOR(47 DOWNTO 0);
      src_mac : IN STD_LOGIC_VECTOR(47 DOWNTO 0);
      payload_byte : IN STD_LOGIC_VECTOR(7 DOWNTO 0)
    );
  END COMPONENT frame_encoder;

  SIGNAL tempDest : STD_LOGIC_VECTOR(47 DOWNTO 0);
  SIGNAL tempAdd : STD_LOGIC_VECTOR(47 DOWNTO 0);
  SIGNAL tempOut : STD_LOGIC_VECTOR(167 DOWNTO 0);
  SIGNAL tempPayload : STD_LOGIC_VECTOR(7 DOWNTO 0);
  SIGNAL frame_ready : STD_LOGIC := '0'; -- Trigger for frame_out update
BEGIN
  -- Instance of frame_encoder
  encoder : frame_encoder PORT MAP(
    frame_out => tempOut,
    dest_mac => tempDest,
    src_mac => tempAdd,
    payload_byte => tempPayload
  );
  PROCESS (inout_bit, clk)
  BEGIN
    IF rising_edge(clk) THEN
      CASE inout_bit IS
        WHEN '0'=>
          frame_out <= (others=> '0');
          frame_ready <= '0';
        WHEN '1' => -- Send state
          tempDest <= MAC_dest;
          tempAdd <= MAC_add;
          tempPayload <= payload;
      END CASE;
    END IF;
  END PROCESS;
END ARCHITECTURE;
```

# SWITCHCAM

Komponen yang menyimpan MAC Table untuk menentukan port mana yang akan menerima frame apabila terdapat frame yang dikirim melalui switch. Mengambil inspirasi dari CAM atau Content Addressable Memory.

```
--processes
begin
    process (main_clk, mac_find)
    begin
        if main_rst = '1' then
            state <= LOAD;
            port_out <= (others => '0');
            hit_flag <= "00";
        elsif rising_edge(main_clk) then
            case state is
                when LOAD =>
                    port_out <= (others => '0');
                    hit_flag <= "00";
                    fill_cam_from_file(MAC, "macTable.txt");
                    state <= ACTIVE;
                when ACTIVE => --constantly in read mode if not write
                    macIn <= mac_find;
                    if (enable = '1') then
                        if (rw_bit = '0') then
                            state <= READ;
                        else state <= WRITE;
                        end if;
                    else state <= ACTIVE;
                    end if;
                when READ =>
                    find_MAC(MAC, portOut, hitFlag, macIn);
                    if (k > 0 and hitFlag = "00") then
                        state <= READ;
                        k <= k - 1;
                    else state <= ASSIGN;
                    end if;
                when WRITE =>
                    state <= ASSIGN;
                    --not now, future update;
                when ASSIGN =>
                    k <= max_port;
                    port_out <= portOut;
                    hit_flag <= hitFlag;
                    state <= COMPLETE;
                when COMPLETE =>
                    state <= ACTIVE;
            end case;
        end if;
    end process;
```

# FRAME ENCODER

```
entity frame_encoder is -- frame encoder
  port (
    frame_out : out std_logic_vector(167 downto 0); -- Assuming 168-bit frame
    dest_mac : in std_logic_vector(47 downto 0);
    src_mac : in std_logic_vector(47 downto 0);
    payload_byte : in std_logic_vector(7 downto 0)
  );
end entity frame_encoder;

architecture rtl of frame_encoder is
begin
  process (dest_mac, payload_byte)
  begin
    frame_out <= "0101010110101011" & dest_mac & src_mac & "0000100000000000" & payload_byte & "11111111111111111111111111111111"; -- Gathering all the fields of the frame
  end process;
end architecture rtl;
```

Komponen yang berfungsi untuk menyusun payload, destination address, dan source address menjadi sebuah frame

# FRAME DECODER

```
entity frame_decoder is -- frame decoder
  port (
    frame_in : in std_logic_vector(167 downto 0); -- Assuming 168-bit frame
    dest_mac : out std_logic_vector(47 downto 0);
    src_mac : out std_logic_vector(47 downto 0);
    payload_byte : out std_logic_vector(7 downto 0)
  );
end entity frame_decoder;

architecture rtl of frame_decoder is
begin
  process (frame_in)
  begin
    dest_mac <= frame_in(151 downto 104);
    src_mac <= frame_in(103 downto 56);
    payload_byte <= frame_in(39 downto 32);
  end process;
end architecture rtl;
```

Komponen yang berfungsi untuk memecah frame yang diterima menjadi data-data seperti payload, dest. Address, dan source address menjadi sebuah frame

# TESTBENCH

Testbench ini digunakan untuk menguji desain switch jaringan dengan mensimulasikan berbagai skenario pengiriman frame antar port. Komponen utama adalah clock generator untuk memicu perubahan state, stimulus process untuk memberikan input seperti MAC address dan payload, serta output verifikasi untuk memeriksa respon switch. Tujuannya adalah memastikan desain switch berfungsi sesuai spesifikasi tanpa menggunakan perangkat keras.

```
-- Clock process definition
clk_process : PROCESS
BEGIN
    clk <= '0';
    WAIT FOR clk_period/2;
    clk <= '1';
    WAIT FOR clk_period/2;
END PROCESS;

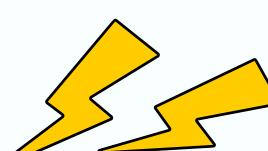
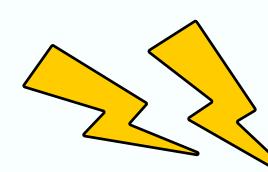
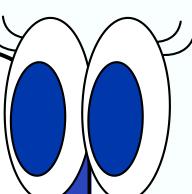
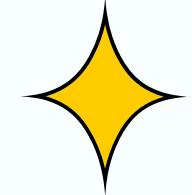
-- Stimulus process
stim_proc: PROCESS
BEGIN
    -- Reset the switch
    reset <= '1';
    WAIT FOR 20 ns;
    reset <= '0';

    -- Test case 1: Sending a frame from port 1
    fa01_MAC <= x"112233445566";
    fa01_InoutBit <= '1';
    fa01_Payload <= x"AA";
    fa01_DestMac <= x"AABBCCDDEEFF";
    WAIT FOR 50 ns;

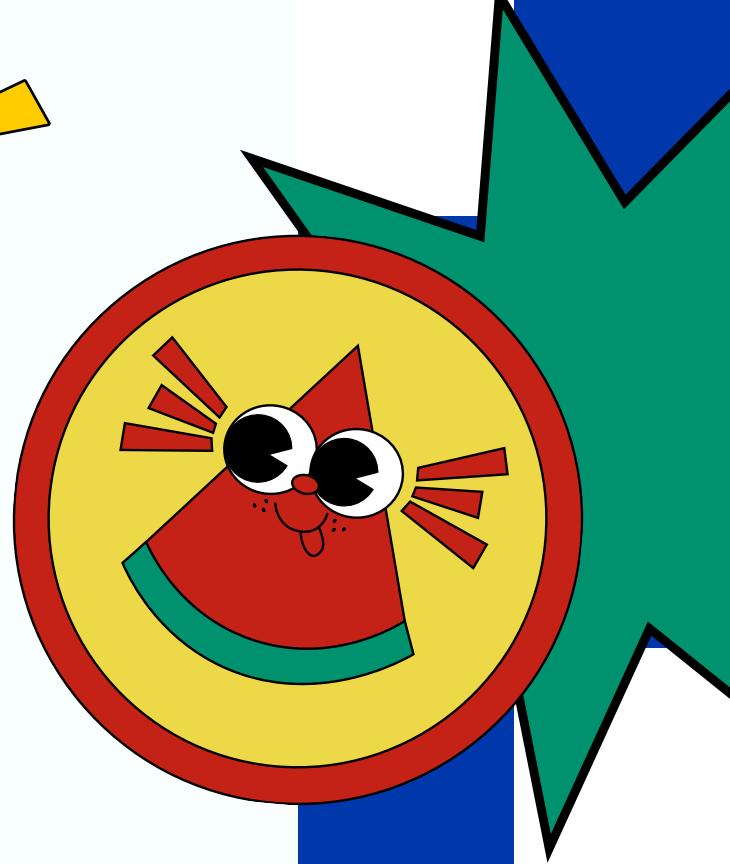
    -- Test case 2: Sending a frame from port 2
    fa02_MAC <= x"223344556677";
    fa02_InoutBit <= '1';
    fa02_Payload <= x"BB";
    fa02_DestMac <= x"FFAABBCCDDEE";
    WAIT FOR 50 ns;

    -- Test case 3: Broadcast scenario
    fa03_MAC <= x"334455667788";
    fa03_InoutBit <= '1';
    fa03_Payload <= x"CC";
    fa03_DestMac <= x"FFFFFFFFFFFF";
    WAIT FOR 50 ns;

    -- End simulation
    WAIT;
END PROCESS;
```

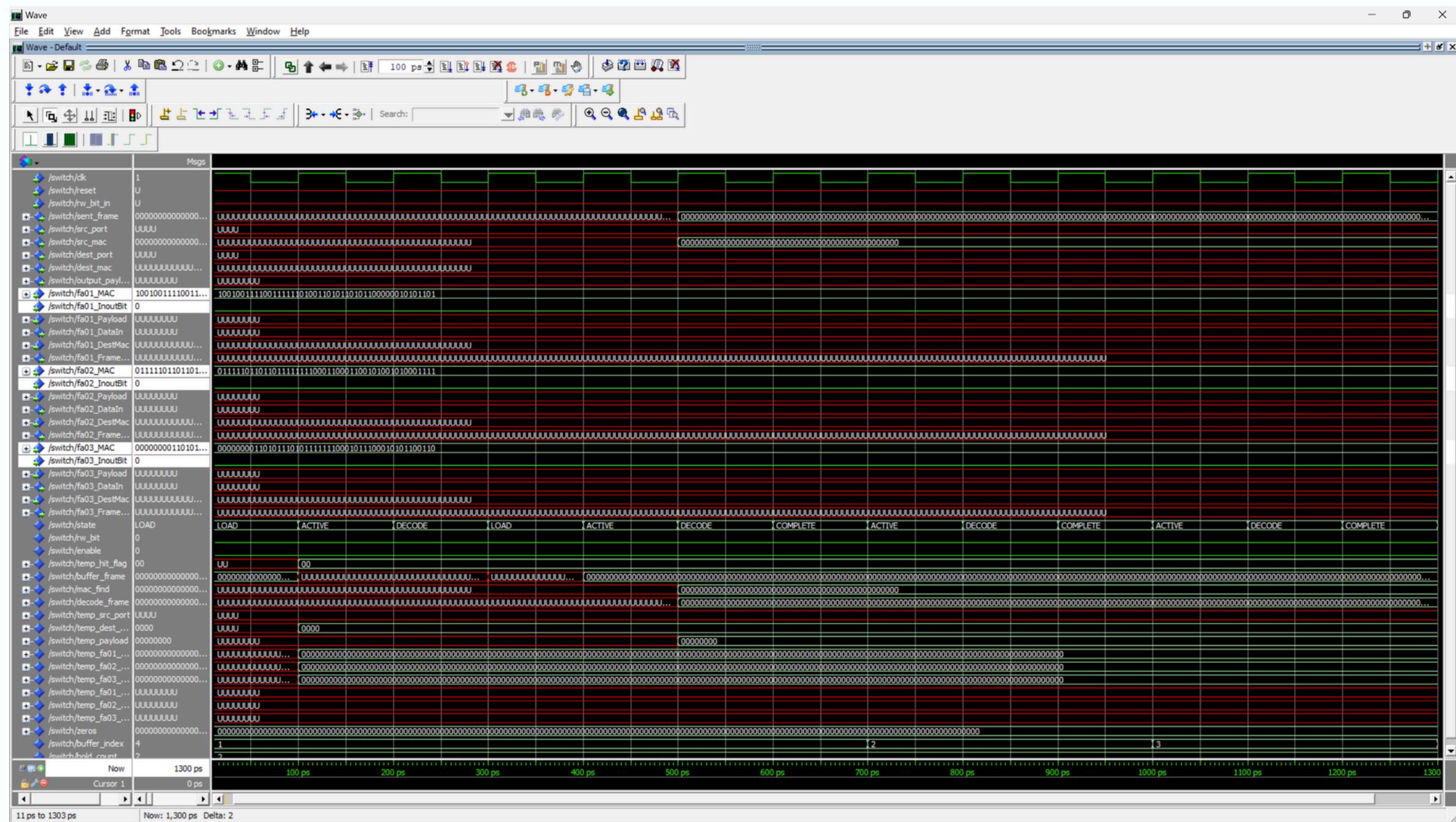


# SIMULATION



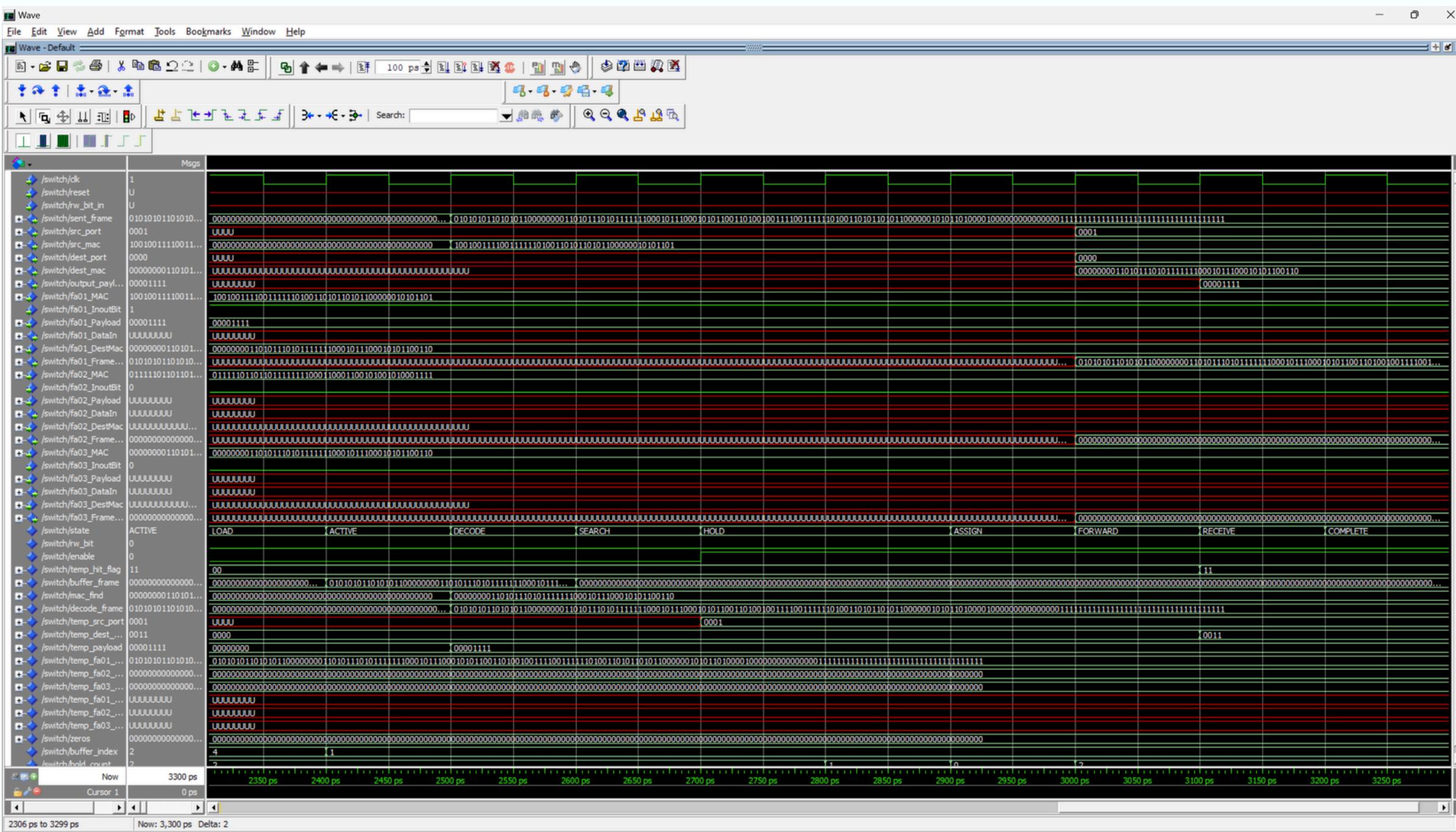
# SIMULASI

Case 1 - MAC Address Assigned, idle state  
Purpose : Initial Assignment Test



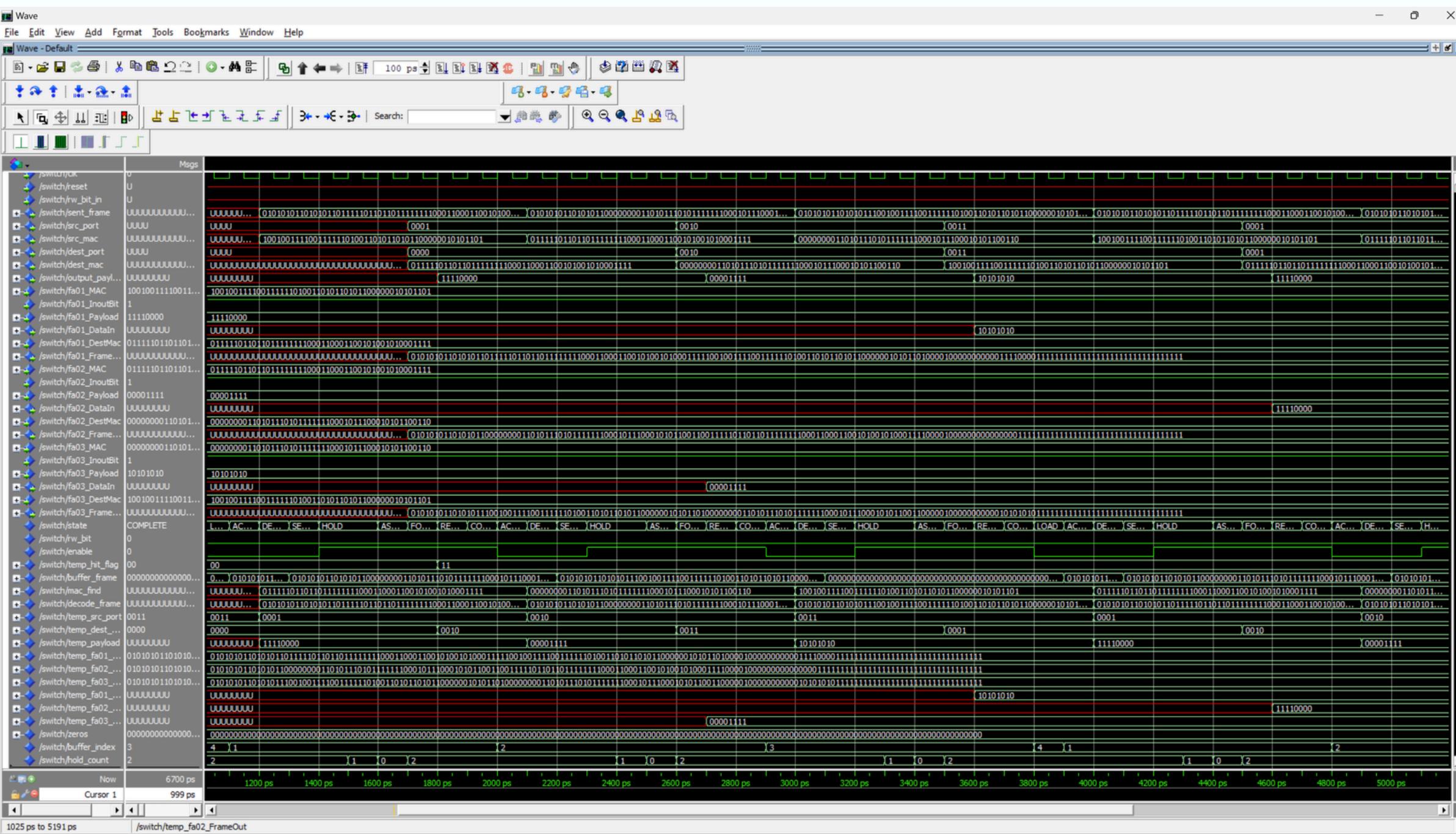
# SIMULASI

Case 2 - Single Send, Fa01 Send to Fa03  
Purpose : Frame Sending Test



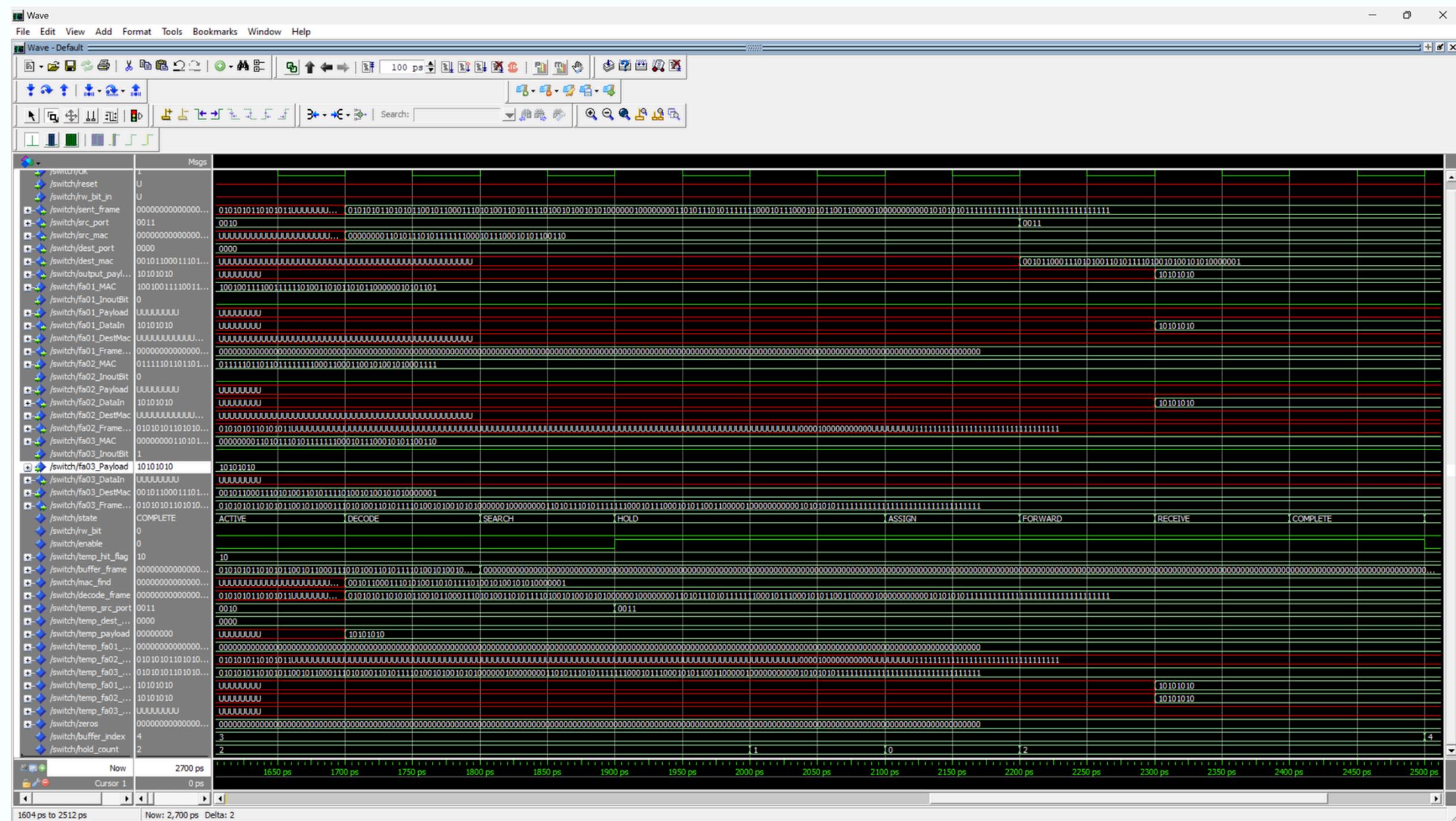
# SIMULASI

Case 3 - Triple Send, Fa01 to Fa02, Fa02 to Fa03, and Fa03 to Fa01,  
Purpose : Check Buffer System

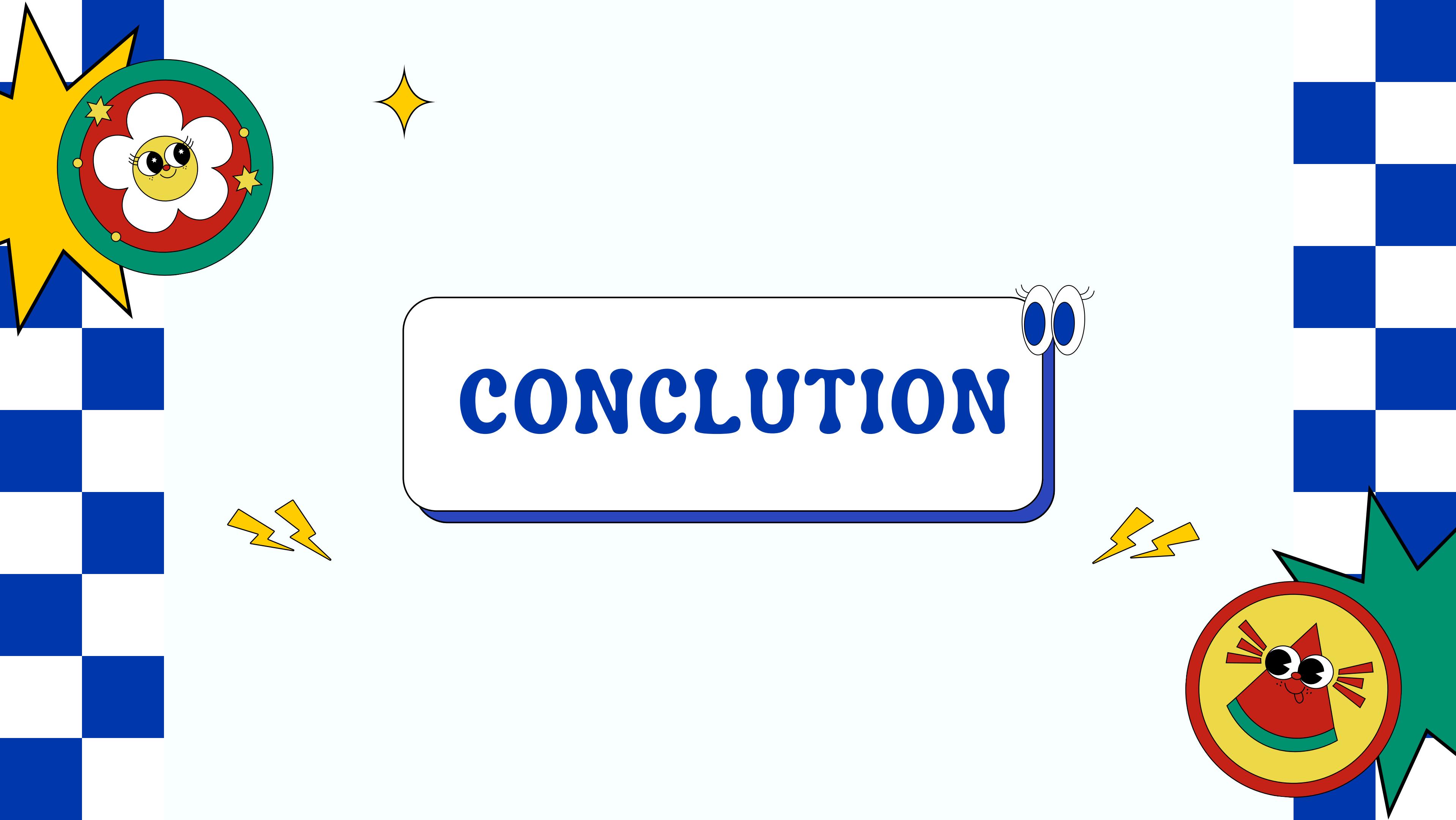


# SIMULASI

Case 4 - Fa03 Send to Unknown MAC  
Purpose : Test Broadcast



# CONCLUSION





## CONCLUSION

Desain frame encoder dan decoder telah memenuhi persyaratan dasar untuk manajemen frame jaringan dengan format tetap, termasuk penggabungan komponen seperti MAC address, payload, dan header frame, serta kemampuan decoder untuk mengekstrak komponen dengan tepat. Pengujian pada empat skenario menunjukkan bahwa sistem dapat mengelola inisialisasi yang stabil, pengiriman frame, pengelolaan buffer, dan fungsi broadcast dengan baik, meskipun efisiensi lalu lintas perlu ditingkatkan. Untuk memenuhi kebutuhan jaringan modern yang lebih kompleks, pengembangan lebih lanjut diperlukan, seperti penerapan tabel MAC yang dinamis, pengoptimalan mekanisme broadcast, penerapan keamanan data dengan checksum, dan peningkatan efisiensi buffer.



SAMPAI  
JUMPA DAN,

**TERIMAKASIH!**

