Project Schedule:

1. Sub-block : Router FIFO

Design Code:

```
module router_fifo(input clock,
                           resetn,
                           write_enb,
                           soft_reset,
                           read_enb,
                           lfd_state,
                   input [7:0] data_in,
                   output empty,full,
                   output reg [7:0] data_out);

    reg [8:0] mem [0:15];
    reg [4:0] wr_ptr,rd_ptr;
    reg [6:0] fifo_counter;
    reg lfd_tmp;
    integer i;

    // Logic for lfd_state
    always@(posedge clock)
      begin
        if(!resetn)
          lfd_tmp <= 0;
        else
          lfd_tmp <= lfd_state;
      end

    // Logic for read and write pointer
    always@(posedge clock)
      begin
        if(!resetn)
          begin
            wr_ptr <= 0;
            rd_ptr <= 0;
          end
        else if(soft_reset)
          begin
            wr_ptr <= 0;
            rd_ptr <= 0;
          end
        else
          begin
            if(!full && write_enb)
              wr_ptr <= wr_ptr + 1'b1;
            if(!empty && read_enb)
              rd_ptr <= rd_ptr + 1'b1;
          end
      end


    // Logic for FIFO counter
    always@(posedge clock)
      begin
        if(!resetn)
          fifo_counter <= 0;
        else if(soft_reset)
          fifo_counter <= 0;
        else if(read_enb && !empty)
          begin
            if(mem[rd_ptr[3:0]][8] == 1'b1)
              fifo_counter <= mem[rd_ptr[3:0]][7:2] + 1'b1;
            else if(fifo_counter)
              fifo_counter <= fifo_counter - 1'b1;
          end
      end
```

```
    // Logic for read operation
    always@(posedge clock)
      begin
        if(!resetn)
          data_out <= 0;
        else  if(soft_reset)
          data_out <= 8'bz;
        else if(!fifo_counter && empty)
          data_out <= 8'bz;
        else if(read_enb && !empty)
          data_out <= mem[rd_ptr[3:0]][7:0];
      end

    // Logic for write operation
    always@(posedge clock)
      begin
        if(!resetn)
          for(i=0;i<16;i=i+1)
            mem[i] <= 0;
        else if(soft_reset)
          for(i=0;i<16;i=i+1)
            mem[i] <= 0;
        else if(write_enb && !full)
          mem[wr_ptr[3:0]] <= {lfd_tmp,data_in};
      end

    // Logic for empty and full
    assign empty = (wr_ptr==rd_ptr)?1'b1:1'b0;
    assign full = (wr_ptr=={~rd_ptr[4],rd_ptr[3:0]})?1'b1:1'b0;

endmodule
```

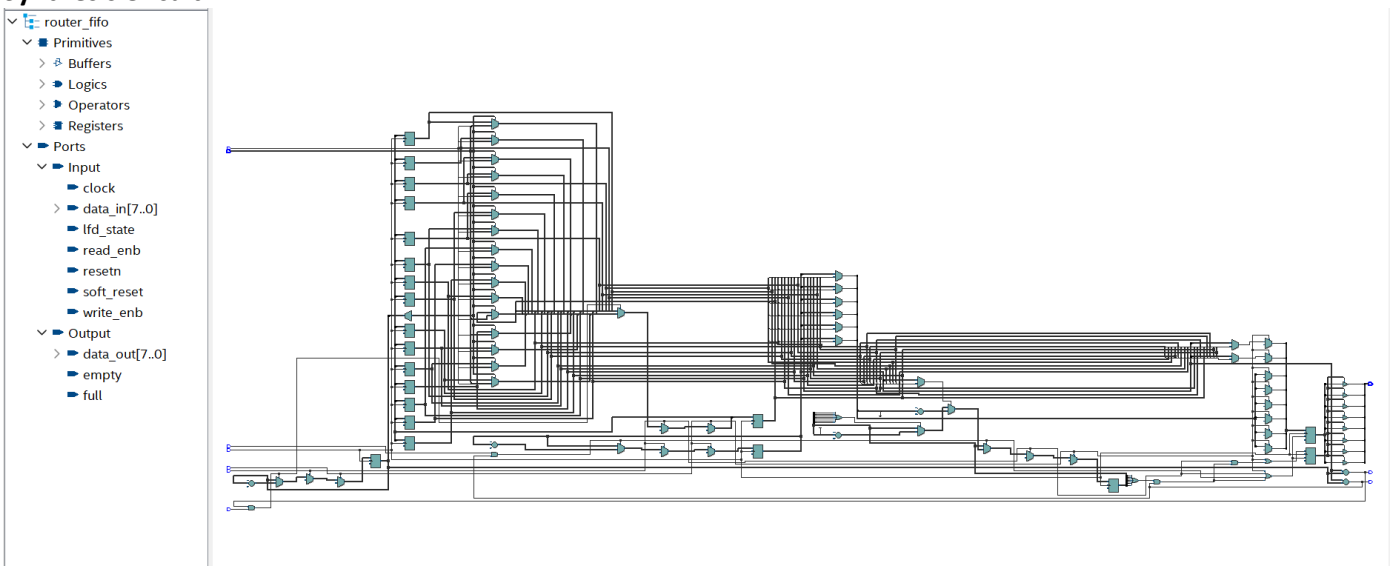
Simulation Code:

```
module router_fifo_tb();
    parameter T = 10,
              TIMEOUT = 30,
              PAYLOAD_LENGTH = 6'd14,
              ADDRESS = 2'd1;
    reg clock_tb,resetn_tb,write_enb_tb,soft_reset_tb,read_enb_tb,lfd_state_tb;
    reg [7:0] data_in_tb;
    wire full_tb, empty_tb;
    wire [7:0] data_out_tb;

    integer i,j,k;
    reg [7:0] header,payload,parity;

    router_fifo
DUT(clock_tb,resetn_tb,write_enb_tb,soft_reset_tb,read_enb_tb,lfd_state_tb,data_in_tb,e
mpty_tb,full_tb,data_out_tb);

    initial
      begin
        clock_tb = 1'b0;
        forever #(T/2) clock_tb = ~clock_tb;
      end

    task initialize;
      begin
        resetn_tb = 1;
        soft_reset_tb = 0;
        write_enb_tb = 0;
        read_enb_tb = 0;
      end
    endtask

    task resetn_ip;
```

```
  begin
    @(negedge clock_tb);
    resetn_tb = 1'b0;
    @(negedge clock_tb);
    resetn_tb = 1'b1;
  end
endtask

task soft_reset_ip;
  begin
    @(negedge clock_tb);
    soft_reset_tb = 1'b1;
    #TIMEOUT;
    @(negedge clock_tb);
    soft_reset_tb = 1'b0;
  end
endtask

task write;
  begin
    lfd_state_tb = 1'b1;
    @(negedge clock_tb);
    header = {PAYLOAD_LENGTH,ADDRESS};
    data_in_tb = header;
    @(negedge clock_tb);
    lfd_state_tb = 1'b0;
    write_enb_tb = 1'b1;
    parity  = 0;
    for(i=0;i<PAYLOAD_LENGTH;i=i+1)
      begin
        @(negedge clock_tb);
        payload = {$random}%256;
        parity = payload ^ parity;
        data_in_tb = payload;
      end
    @(negedge clock_tb);
    data_in_tb = parity;
  end
endtask

task read(input j,k);
  begin
    write_enb_tb = j;
    read_enb_tb = k;
  end
endtask

initial
  begin
    initialize;
    resetn_ip;
    #20;
    write;
    #20;
    for(i=0;i<PAYLOAD_LENGTH + 2;i=i+1)
      begin
        read(1'b0,1'b1);
        #15;
      end
    read(1'b0,1'b0);
    #20;
    resetn_ip;
    #10;
    write;
    #30;
    soft_reset_ip;
    #30;
```

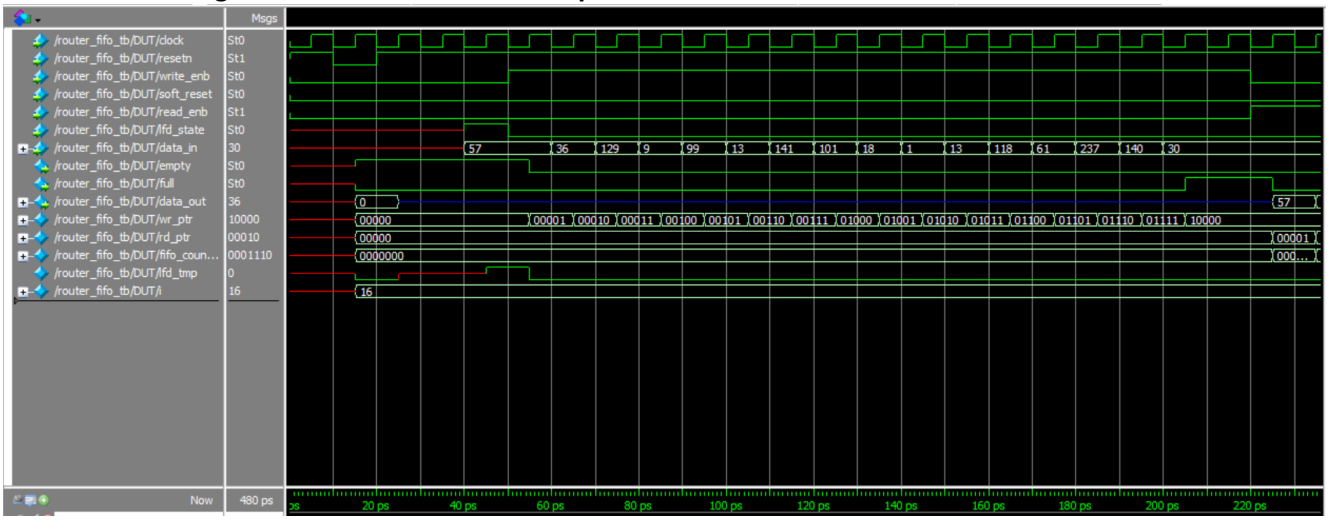
```
        $finish;
      end

    initial
      $monitor("Data Input = %b, Data_output = %b, Reset = %b, Soft Reset = %b, Write
Enable = %b, Read Enable = %b,Lfd State = %b, FIFO Counter = %b, Empty = %b, Full =
%b",
data_in_tb,data_out_tb,resetn_tb,soft_reset_tb,write_enb_tb,read_enb_tb,lfd_state_tb,DU
T.fifo_counter,empty_tb,full_tb);
endmodule
```

Synthesis Circuit:

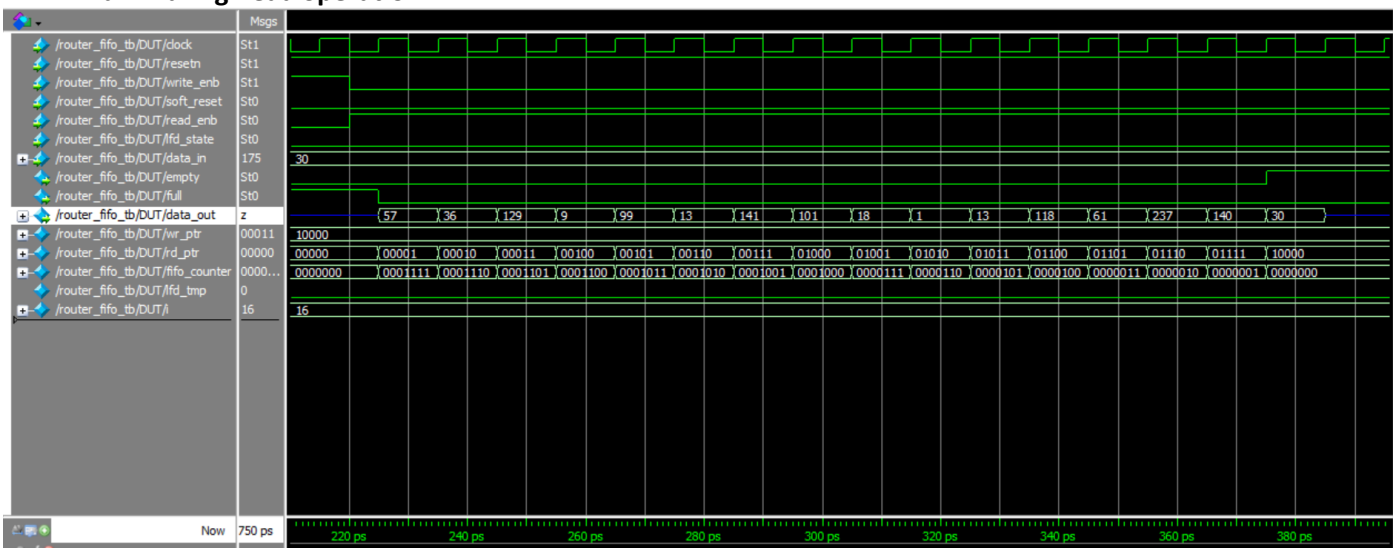


Simulation Waveform:

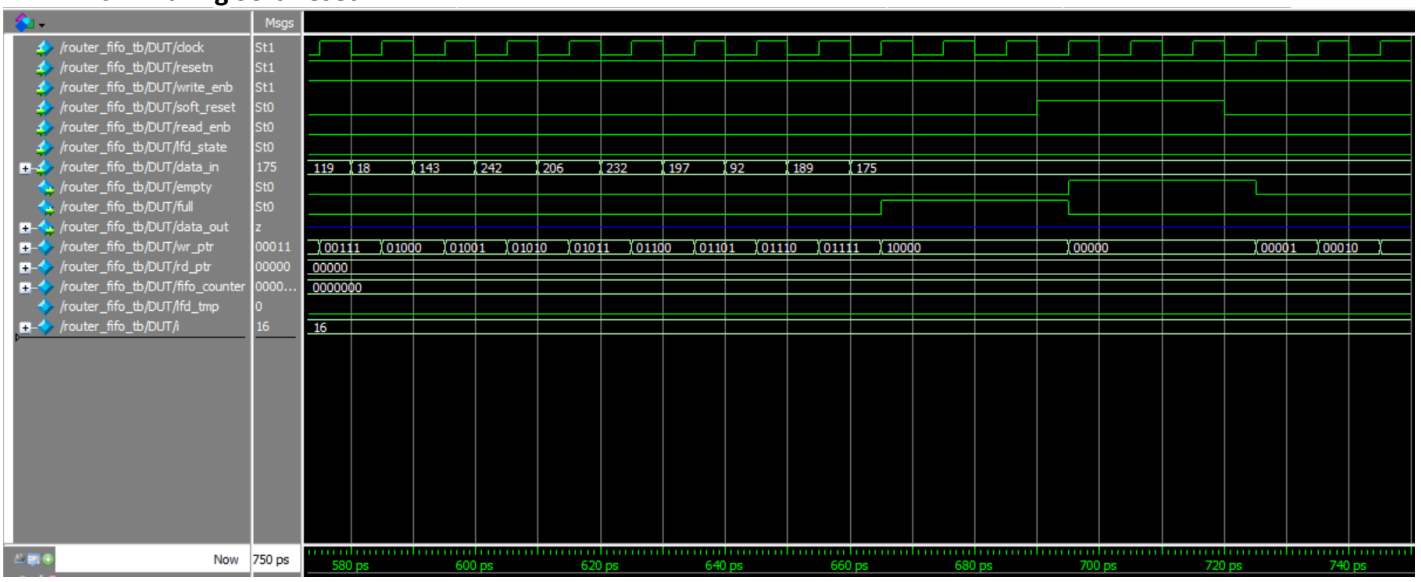
a. **During Active low Reset and write operation**



b. During Read operation



c. During Soft Reset



Transcript:

```
# Data Input = xxxxxxxx, Data_output = xxxxxxxx, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 0,Lfd State = x, FIFO Counter = xxxxxxx, Empty = x, Full = x
# Data Input = xxxxxxxx, Data_output = xxxxxxxx, Reset = 0, Soft Reset = 0, Write Enable = 0, Read Enable = 0,Lfd State = x, FIFO Counter = xxxxxxx, Empty = x, Full = x
# Data Input = xxxxxxxx, Data_output = 00000000, Reset = 0, Soft Reset = 0, Write Enable = 0, Read Enable = 0,Lfd State = x, FIFO Counter = 0000000, Empty = 1, Full = 0
# Data Input = xxxxxxxx, Data_output = 00000000, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 0,Lfd State = x, FIFO Counter = 0000000, Empty = 1, Full = 0
# Data Input = xxxxxxxx, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 0,Lfd State = x, FIFO Counter = 0000000, Empty = 1, Full = 0
# Data Input = 00111001, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 0,Lfd State = 1, FIFO Counter = 0000000, Empty = 1, Full = 0
# Data Input = 00111001, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 1, Full = 0
# Data Input = 00111001, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 00100100, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 10000001, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 00001001, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 01100011, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 00001101, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 10001101, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 01100101, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 00010010, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 00000001, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 00001101, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 01110110, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 00111101, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 11101101, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 10001100, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 00011110, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 00011110, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 1
# Data Input = 00011110, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 1,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 1
# Data Input = 00011110, Data_output = 00111001, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 1,Lfd State = 0, FIFO Counter = 0001111, Empty = 0, Full = 0
# Data Input = 00011110, Data_output = 00100100, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 1,Lfd State = 0, FIFO Counter = 0001110, Empty = 0, Full = 0
# Data Input = 00011110, Data_output = 10000001, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 1,Lfd State = 0, FIFO Counter = 0001101, Empty = 0, Full = 0
# Data Input = 00011110, Data_output = 00001001, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 1,Lfd State = 0, FIFO Counter = 0001100, Empty = 0, Full = 0
# Data Input = 00011110, Data_output = 01100011, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 1,Lfd State = 0, FIFO Counter = 0001011, Empty = 0, Full = 0
# Data Input = 00011110, Data_output = 00001101, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 1,Lfd State = 0, FIFO Counter = 0001010, Empty = 0, Full = 0
# Data Input = 00011110, Data_output = 10001101, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 1,Lfd State = 0, FIFO Counter = 0001001, Empty = 0, Full = 0
# Data Input = 00011110, Data_output = 01100101, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 1,Lfd State = 0, FIFO Counter = 0001000, Empty = 0, Full = 0
# Data Input = 00011110, Data_output = 00010010, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 1,Lfd State = 0, FIFO Counter = 0000111, Empty = 0, Full = 0
# Data Input = 00011110, Data_output = 00000001, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 1,Lfd State = 0, FIFO Counter = 0000110, Empty = 0, Full = 0
# Data Input = 00011110, Data_output = 00001101, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 1,Lfd State = 0, FIFO Counter = 0000101, Empty = 0, Full = 0
```

```
# Data Input = 00011110, Data_output = 01110110, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 1,Lfd State = 0, FIFO Counter = 0000100, Empty = 0, Full = 0
# Data Input = 00011110, Data_output = 00111101, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 1,Lfd State = 0, FIFO Counter = 0000011, Empty = 0, Full = 0
# Data Input = 00011110, Data_output = 11101101, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 1,Lfd State = 0, FIFO Counter = 0000010, Empty = 0, Full = 0
# Data Input = 00011110, Data_output = 10001100, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 1,Lfd State = 0, FIFO Counter = 0000001, Empty = 0, Full = 0
# Data Input = 00011110, Data_output = 00011110, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 1,Lfd State = 0, FIFO Counter = 0000000, Empty = 1, Full = 0
# Data Input = 00011110, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 1,Lfd State = 0, FIFO Counter = 0000000, Empty = 1, Full = 0
# Data Input = 00011110, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 1, Full = 0
# Data Input = 00011110, Data_output = zzzzzzzz, Reset = 0, Soft Reset = 0, Write Enable = 0, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 1, Full = 0
# Data Input = 00011110, Data_output = 00000000, Reset = 0, Soft Reset = 0, Write Enable = 0, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 1, Full = 0
# Data Input = 00011110, Data_output = 00000000, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 1, Full = 0
# Data Input = 00011110, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 1, Full = 0
# Data Input = 00111001, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 0, Read Enable = 0,Lfd State = 1, FIFO Counter = 0000000, Empty = 1, Full = 0
# Data Input = 00111001, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 1, Full = 0
# Data Input = 00111001, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 11111001, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 11000110, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 11000101, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 10101010, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 11100101, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 01110111, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 00010010, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 10001111, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 11110010, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 11001110, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 11101000, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 11000101, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 01011100, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 10111101, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 10101111, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
# Data Input = 10101111, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 1
# Data Input = 10101111, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 1, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 1
# Data Input = 10101111, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 1, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 1, Full = 0
# Data Input = 10101111, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 1, Full = 0
# Data Input = 10101111, Data_output = zzzzzzzz, Reset = 1, Soft Reset = 0, Write Enable = 1, Read Enable = 0,Lfd State = 0, FIFO Counter = 0000000, Empty = 0, Full = 0
```

2. Sub-Block : Router Synchronizer

Design Code:

```
module router_sync(input clock,
                           resetn,
                           detect_add,
                           write_enb_reg,
                           read_enb_0,
                           read_enb_1,
                           read_enb_2,
                           empty_0,
                           empty_1,
                           empty_2,
                           full_0,
                           full_1,
                           full_2,
                   input [1:0] data_in,
                   output reg fifo_full,
                              soft_reset_0,
                              soft_reset_1,
                              soft_reset_2,
                   output vld_out_0,
                          vld_out_1,
                          vld_out_2,
                   output reg [2:0] write_enb);

    reg[1:0] fifo_addr;
    reg[4:0] timer_0,timer_1,timer_2;

    // Logic for storing address
    always@(posedge clock)
      begin
        if(!resetn)
          fifo_addr <= 0;
        else if(detect_add)
          fifo_addr <= data_in;
      end

    // Logic for fifo full status
    always@(*)
      begin
        if(!resetn)
          fifo_full = 0;
        else
          begin
            case(fifo_addr)
              2'b00 : fifo_full = full_0;
              2'b01 : fifo_full = full_1;
              2'b10 : fifo_full = full_2;
              default: fifo_full = 0;
            endcase
          end
      end

    // Logic for write enable
    always@(*)
      begin
        if(!resetn)
          write_enb = 0;
        else if(write_enb_reg)
          begin
            case(fifo_addr)
              2'b00 : write_enb = 3'b001;
              2'b01 : write_enb = 3'b010;
```

```
            2'b10 : write_enb = 3'b100;
            default : write_enb = 3'b0;
          endcase
        end
      else
        write_enb = 0;
    end

  // Logic for valid out signal
  assign vld_out_0 = ~empty_0;
  assign vld_out_1 = ~empty_1;
  assign vld_out_2 = ~empty_2;

  // Logic for soft reset 0
  always@(posedge clock)
    begin
      if(!resetn)
        begin
          soft_reset_0 <= 0;
          timer_0 <= 0;
        end
      else if(~vld_out_0)
        begin
          soft_reset_0 <= 0;
          timer_0 <= 0;
        end
      else if(read_enb_0)
        begin
          soft_reset_0 <= 0;
          timer_0 <= 0;
        end
      else
        begin
          if(timer_0 == 5'd29)
            begin
              soft_reset_0 <= 1'b1;
              timer_0 <= 0;
            end
          else
            begin
              timer_0 <= timer_0 + 1'b1;
              soft_reset_0 <= 0;
            end
        end
    end


  // Logic for soft reset 1
  always@(posedge clock)
    begin
      if(!resetn)
        begin
          soft_reset_1 <= 0;
          timer_1 <= 0;
        end
      else if(~vld_out_1)
        begin
          soft_reset_1 <= 0;
          timer_1 <= 0;
        end
      else if(read_enb_1)
        begin
          soft_reset_1 <= 0;
          timer_1 <= 0;
        end
      else
        begin
```

```
            if(timer_1 == 5'd29)
```

```
            begin
              soft_reset_1 <= 1'b1;
              timer_1 <= 0;
            end
          else
            begin
              timer_1 <= timer_1 + 1'b1;
              soft_reset_1 <= 0;
            end
        end
    end

  // Logic for soft reset 2
  always@(posedge clock)
    begin
      if(!resetn)
        begin
          soft_reset_2 <= 0;
          timer_2 <= 0;
        end
      else if(~vld_out_2)
        begin
          soft_reset_2 <= 0;
          timer_2 <= 0;
        end
      else if(read_enb_2)
        begin
          soft_reset_2 <= 0;
          timer_2 <= 0;
        end
      else
        begin
          if(timer_2 == 5'd29)
            begin
              soft_reset_2 <= 1'b1;
              timer_2 <= 0;
            end
          else
            begin
              timer_2 <= timer_2 + 1'b1;
              soft_reset_2 <= 0;
            end
        end
    end
endmodule
```

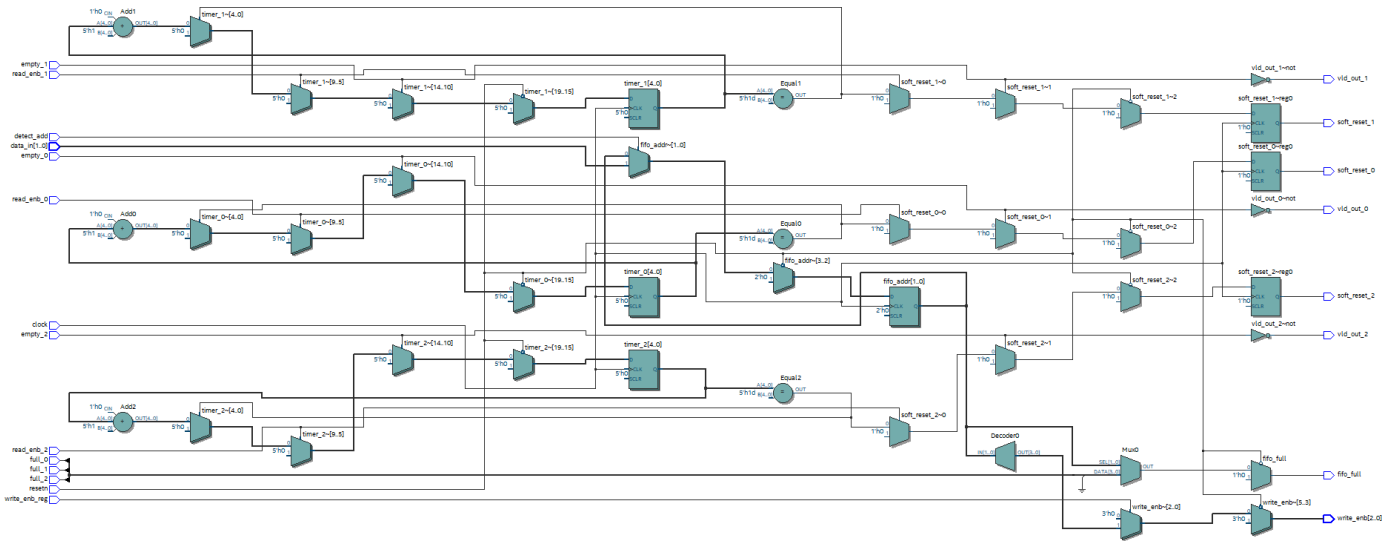
Simulation Code:

```
module router_sync_tb();
  reg
clock_tb,resetn_tb,detect_add_tb,write_enb_reg_tb,read_enb_0_tb,read_enb_1_tb,read_enb_
2_tb,empty_0_tb,empty_1_tb,empty_2_tb,full_0_tb,full_1_tb,full_2_tb;
  reg [1:0] data_in_tb;
  wire
fifo_full_tb,soft_reset_0_tb,soft_reset_1_tb,soft_reset_2_tb,vld_out_0_tb,vld_out_1_tb,
vld_out_2_tb;
  wire [2:0] write_enb_tb;

  parameter T = 10,
            TIMEOUT = 30;

  router_sync DUT(clock_tb,
                  resetn_tb,
                  detect_add_tb,
                  write_enb_reg_tb,
                  read_enb_0_tb,
                  read_enb_1_tb,
```

```
                    read_enb_2_tb,
                    empty_0_tb,
                    empty_1_tb,
                    empty_2_tb,
                    full_0_tb,
                    full_1_tb,
                    full_2_tb,
                    data_in_tb,
                    fifo_full_tb,
                    soft_reset_0_tb,
                    soft_reset_1_tb,
                    soft_reset_2_tb,
                    vld_out_0_tb,
                    vld_out_1_tb,
                    vld_out_2_tb,
                    write_enb_tb);

  integer i,m;

  initial
    begin
      clock_tb = 1'b0;
      forever #(T/2) clock_tb = ~clock_tb;
    end

  task resetn_ip;
    begin
      @(negedge clock_tb);
      resetn_tb = 1'b0;
      @(negedge clock_tb);
      resetn_tb = 1'b1;
    end
  endtask

  task initialize;
    begin
      resetn_tb = 1'b1;
      detect_add_tb = 1'b0;
      write_enb_reg_tb = 1'b0;
      read_enb_0_tb = 1'b0;
      read_enb_1_tb = 1'b0;
      read_enb_2_tb = 1'b0;
      empty_0_tb = 1'b0;
      empty_1_tb = 1'b0;
      empty_2_tb = 1'b0;
      full_0_tb = 1'b0;
      full_1_tb = 1'b0;
      full_2_tb = 1'b0;
    end
  endtask

  task address_ip(input [1:0] addr_tb);
    begin
      @(negedge clock_tb);
      detect_add_tb = 1'b1;
      data_in_tb = addr_tb;
      @(negedge clock_tb);
      detect_add_tb = 1'b0;
    end
  endtask

  task write_enb_ip;
    begin
      @(negedge clock_tb);
      write_enb_reg_tb = 1'b1;
      @(negedge clock_tb);
      write_enb_reg_tb = 1'b0;
```

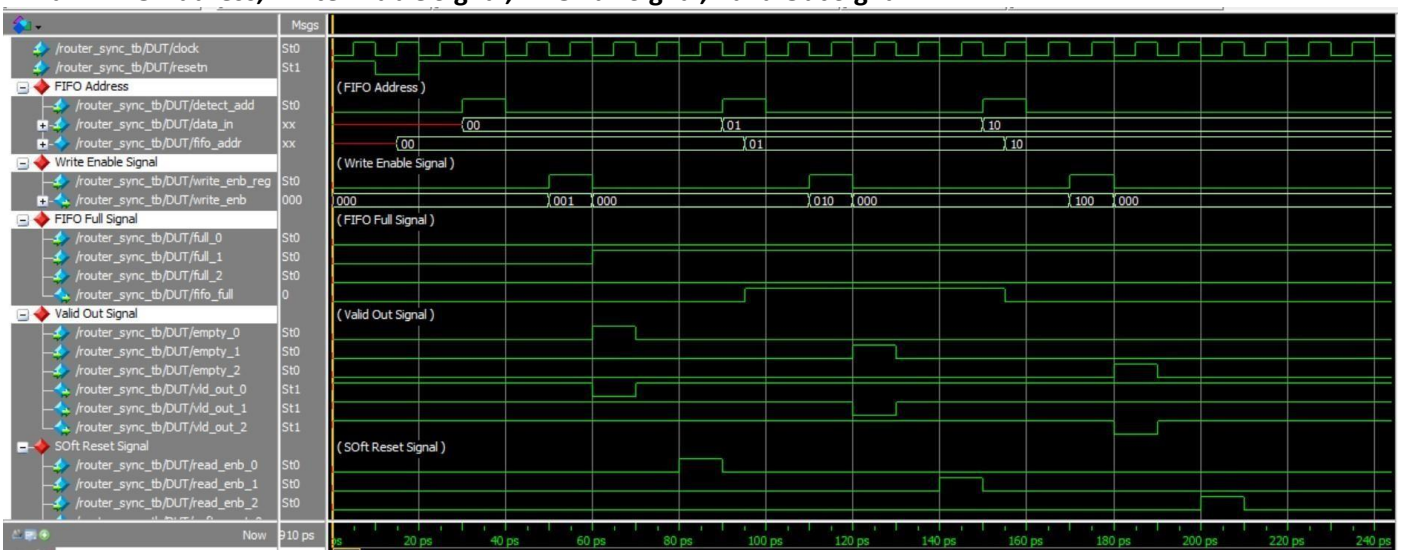
```
      end
    endtask

    task fifo_full_ip;
      begin
        full_0_tb = 1'b0;
        full_1_tb = 1'b1;
        full_2_tb = 1'b0;
      end
    endtask

    task empty_ip(input [3:0] j,k);
      begin
        empty_0_tb = j[0];
        empty_1_tb = j[1];
        empty_2_tb = j[2];
        read_enb_0_tb = k[0];
        read_enb_1_tb = k[1];
        read_enb_2_tb = k[2];
      end
    endtask

    initial
      begin
        initialize;
        resetn_ip;
        #10;
        m = 1;
        for(i=0;i<3;i=i+1)
          begin
            address_ip(i);
            write_enb_ip;
            fifo_full_ip;
            empty_ip(m,0);
            #10;
            empty_ip(0,0);
            #10;
            empty_ip(0,m);
            #10;
            empty_ip(0,0);
            m = m*2;
          end
        #700;
        $finish;
      end

    initial
      $monitor("Inputs : Detect Address = %b, Data_in = %b, Write Enable Reg = %b,
Empty_0 = %b, Empty_1 = %b, Empty_2 = %b, Full_0 = %b, Full_1 = %b, Full_2 = %b,
ReadEnable_0 = %b, ReadEnable_1 = %b, ReadEnable_2 = %b \nOutputs : Fifo Address = %b,
Write Enable = %b,Fifo Full = %b, ValidOut_0 = %b, ValidOut_1 = %b, ValidOut_2 = %b,
SoftReset_0 = %b, SoftReset_1 = %B, SoftReset_2 = %b",

detect_add_tb,data_in_tb,write_enb_reg_tb,empty_0_tb,empty_1_tb,empty_2_tb,full_0_tb,fu
ll_1_tb,full_2_tb,read_enb_0_tb,read_enb_1_tb,read_enb_2_tb,DUT.fifo_addr,
write_enb_tb,fifo_full_tb,vld_out_0_tb,vld_out_1_tb,vld_out_2_tb,soft_reset_0_tb,soft_r
eset_1_tb,soft_reset_2_tb);
endmodule
```

Synthesis Circuit:



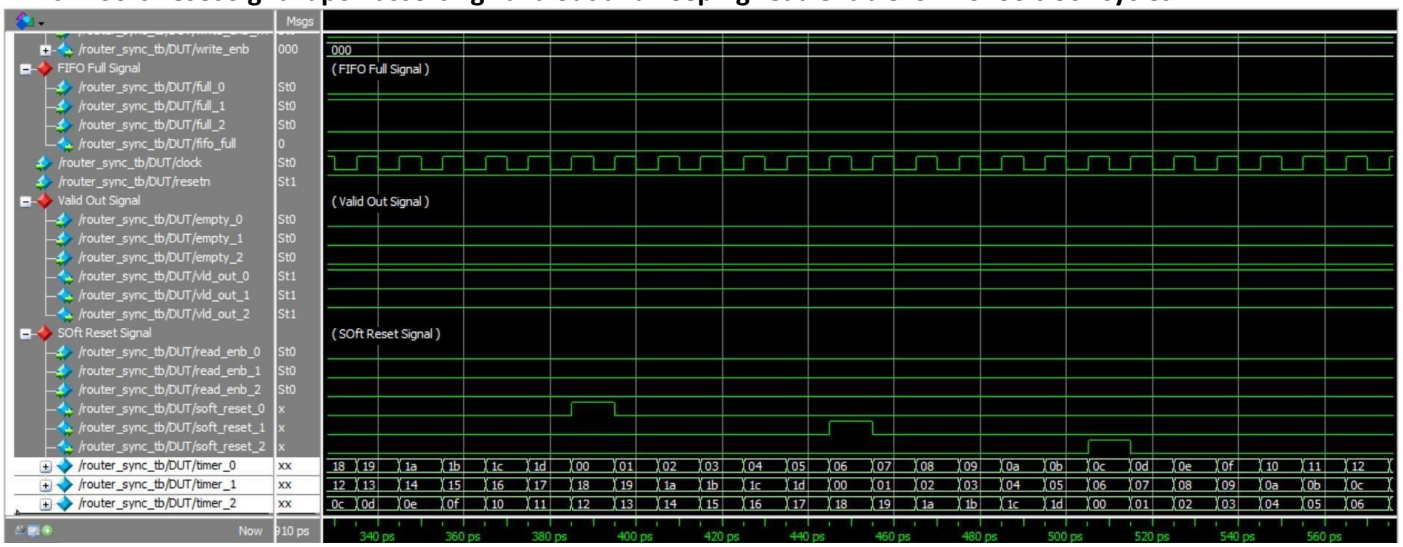
Simulation Waveform:

a. FIFO Address, Write Enable Signal, FIFO Full Signal, Valid Out Signal



b. Soft Reset Signal upon application of empty and read enable signal

c. Soft Reset Signal upon asserting valid out and keeping read enable low for 30 clock cycles



Transcript:

```
# Inputs :  Detect Address = 0, Data_in = xx, Write Enable Reg = 0, Empty_0 = 0, Empty_1 = 0, Empty_2 = 0, Full_0 = 0, Full_1 = 0, Full_2 = 0, ReadEnable_0 = 0, ReadEnable_1 = 0,
ReadEnable_2 = 0
# Outputs :  Fifo Address = xx, Write Enable = 000,Fifo Full = 0, ValidOut_0 = 1, ValidOut_1 = 1, ValidOut_2 = 1, SoftReset_0 = x, SoftReset_1 = x, SoftReset_2 = x
# Inputs :  Detect Address = 0, Data_in = xx, Write Enable Reg = 0, Empty_0 = 0, Empty_1 = 0, Empty_2 = 0, Full_0 = 0, Full_1 = 0, Full_2 = 0, ReadEnable_0 = 0, ReadEnable_1 = 0,
ReadEnable_2 = 0
# Outputs :  Fifo Address = xx, Write Enable = 000,Fifo Full = 0, ValidOut_0 = 1, ValidOut_1 = 1, ValidOut_2 = 1, SoftReset_0 = 0, SoftReset_1 = 0, SoftReset_2 = 0
# Inputs :  Detect Address = 0, Data_in = xx, Write Enable Reg = 0, Empty_0 = 0, Empty_1 = 0, Empty_2 = 0, Full_0 = 0, Full_1 = 0, Full_2 = 0, ReadEnable_0 = 0, ReadEnable_1 = 0,
ReadEnable_2 = 0
# Outputs :  Fifo Address = 00, Write Enable = 000,Fifo Full = 0, ValidOut_0 = 1, ValidOut_1 = 1, ValidOut_2 = 1, SoftReset_0 = 0, SoftReset_1 = 0, SoftReset_2 = 0
# Inputs :  Detect Address = 1, Data_in = 00, Write Enable Reg = 0, Empty_0 = 0, Empty_1 = 0, Empty_2 = 0, Full_0 = 0, Full_1 = 0, Full_2 = 0, ReadEnable_0 = 0, ReadEnable_1 = 0,
ReadEnable_2 = 0
# Outputs :  Fifo Address = 00, Write Enable = 000,Fifo Full = 0, ValidOut_0 = 1, ValidOut_1 = 1, ValidOut_2 = 1, SoftReset_0 = 0, SoftReset_1 = 0, SoftReset_2 = 0
# Inputs :  Detect Address = 0, Data_in = 00, Write Enable Reg = 0, Empty_0 = 0, Empty_1 = 0, Empty_2 = 0, Full_0 = 0, Full_1 = 0, Full_2 = 0, ReadEnable_0 = 0, ReadEnable_1 = 0,
ReadEnable_2 = 0
# Outputs :  Fifo Address = 00, Write Enable = 000,Fifo Full = 0, ValidOut_0 = 0, ValidOut_1 = 1, ValidOut_2 = 1, SoftReset_0 = 0, SoftReset_1 = 0, SoftReset_2 = 0
# Inputs :  Detect Address = 0, Data_in = 00, Write Enable Reg = 1, Empty_0 = 0, Empty_1 = 0, Empty_2 = 0, Full_0 = 0, Full_1 = 0, Full_2 = 0, ReadEnable_0 = 0, ReadEnable_1 = 0,
ReadEnable_2 = 0
# Outputs :  Fifo Address = 00, Write Enable = 001,Fifo Full = 0, ValidOut_0 = 1, ValidOut_1 = 1, ValidOut_2 = 1, SoftReset_0 = 0, SoftReset_1 = 0, SoftReset_2 = 0
# Inputs :  Detect Address = 0, Data_in = 00, Write Enable Reg = 0, Empty_0 = 1, Empty_1 = 0, Empty_2 = 0, Full_0 = 0, Full_1 = 1, Full_2 = 0, ReadEnable_0 = 0, ReadEnable_1 = 0,
ReadEnable_2 = 0
# Outputs :  Fifo Address = 00, Write Enable = 000,Fifo Full = 0, ValidOut_0 = 0, ValidOut_1 = 1, ValidOut_2 = 1, SoftReset_0 = 0, SoftReset_1 = 0, SoftReset_2 = 0
# Inputs :  Detect Address = 0, Data_in = 00, Write Enable Reg = 0, Empty_0 = 0, Empty_1 = 0, Empty_2 = 0, Full_0 = 0, Full_1 = 1, Full_2 = 0, ReadEnable_0 = 0, ReadEnable_1 = 0,
ReadEnable_2 = 0
# Outputs :  Fifo Address = 00, Write Enable = 000,Fifo Full = 0, ValidOut_0 = 1, ValidOut_1 = 1, ValidOut_2 = 1, SoftReset_0 = 0, SoftReset_1 = 0, SoftReset_2 = 0
# Inputs :  Detect Address = 0, Data_in = 00, Write Enable Reg = 0, Empty_0 = 0, Empty_1 = 0, Empty_2 = 0, Full_0 = 0, Full_1 = 1, Full_2 = 0, ReadEnable_0 = 1, ReadEnable_1 = 0,
ReadEnable_2 = 0
# Outputs :  Fifo Address = 00, Write Enable = 000,Fifo Full = 0, ValidOut_0 = 1, ValidOut_1 = 1, ValidOut_2 = 1, SoftReset_0 = 0, SoftReset_1 = 0, SoftReset_2 = 0
# Inputs :  Detect Address = 1, Data_in = 01, Write Enable Reg = 0, Empty_0 = 0, Empty_1 = 0, Empty_2 = 0, Full_0 = 0, Full_1 = 1, Full_2 = 0, ReadEnable_0 = 0, ReadEnable_1 = 0,
ReadEnable_2 = 0
# Outputs :  Fifo Address = 00, Write Enable = 000,Fifo Full = 0, ValidOut_0 = 1, ValidOut_1 = 1, ValidOut_2 = 1, SoftReset_0 = 0, SoftReset_1 = 0, SoftReset_2 = 0
# Inputs :  Detect Address = 1, Data_in = 01, Write Enable Reg = 0, Empty_0 = 0, Empty_1 = 0, Empty_2 = 0, Full_0 = 0, Full_1 = 1, Full_2 = 0, ReadEnable_0 = 0, ReadEnable_1 = 0,
ReadEnable_2 = 0
# Outputs :  Fifo Address = 01, Write Enable = 000,Fifo Full = 1, ValidOut_0 = 1, ValidOut_1 = 1, ValidOut_2 = 1, SoftReset_0 = 0, SoftReset_1 = 0, SoftReset_2 = 0
# Inputs :  Detect Address = 1, Data_in = 10, Write Enable Reg = 0, Empty_0 = 0, Empty_1 = 0, Empty_2 = 0, Full_0 = 0, Full_1 = 1, Full_2 = 0, ReadEnable_0 = 0, ReadEnable_1 = 0,
ReadEnable_2 = 0
# Outputs :  Fifo Address = 10, Write Enable = 000,Fifo Full = 0, ValidOut_0 = 1, ValidOut_1 = 1, ValidOut_2 = 1, SoftReset_0 = 0, SoftReset_1 = 0, SoftReset_2 = 0
# Inputs :  Detect Address = 0, Data_in = 10, Write Enable Reg = 0, Empty_0 = 0, Empty_1 = 0, Empty_2 = 0, Full_0 = 0, Full_1 = 1, Full_2 = 0, ReadEnable_0 = 0, ReadEnable_1 = 0,
ReadEnable_2 = 0
# Outputs :  Fifo Address = 10, Write Enable = 000,Fifo Full = 0, ValidOut_0 = 1, ValidOut_1 = 1, ValidOut_2 = 1, SoftReset_0 = 0, SoftReset_1 = 0, SoftReset_2 = 0
# Inputs :  Detect Address = 0, Data_in = 10, Write Enable Reg = 1, Empty_0 = 0, Empty_1 = 0, Empty_2 = 0, Full_0 = 0, Full_1 = 1, Full_2 = 0, ReadEnable_0 = 0, ReadEnable_1 = 0,
ReadEnable_2 = 0
# Outputs :  Fifo Address = 10, Write Enable = 100,Fifo Full = 0, ValidOut_0 = 1, ValidOut_1 = 1, ValidOut_2 = 1, SoftReset_0 = 0, SoftReset_1 = 0, SoftReset_2 = 0
# Inputs :  Detect Address = 0, Data_in = 10, Write Enable Reg = 0, Empty_0 = 0, Empty_1 = 0, Empty_2 = 1, Full_0 = 0, Full_1 = 1, Full_2 = 0, ReadEnable_0 = 0, ReadEnable_1 = 0,
ReadEnable_2 = 0
# Outputs :  Fifo Address = 10, Write Enable = 000,Fifo Full = 0, ValidOut_0 = 1, ValidOut_1 = 1, ValidOut_2 = 1, SoftReset_0 = 0, SoftReset_1 = 0, SoftReset_2 = 0
# Inputs :  Detect Address = 0, Data_in = 10, Write Enable Reg = 0, Empty_0 = 0, Empty_1 = 0, Empty_2 = 0, Full_0 = 0, Full_1 = 1, Full_2 = 0, ReadEnable_0 = 0, ReadEnable_1 = 0,
ReadEnable_2 = 0
# Outputs :  Fifo Address = 10, Write Enable = 000,Fifo Full = 0, ValidOut_0 = 1, ValidOut_1 = 1, ValidOut_2 = 1, SoftReset_0 = 0, SoftReset_1 = 0, SoftReset_2 = 0
# Inputs :  Detect Address = 0, Data_in = 10, Write Enable Reg = 0, Empty_0 = 0, Empty_1 = 0, Empty_2 = 0, Full_0 = 0, Full_1 = 1, Full_2 = 0, ReadEnable_0 = 0, ReadEnable_1 = 0,
ReadEnable_2 = 1
# Outputs :  Fifo Address = 10, Write Enable = 000,Fifo Full = 0, ValidOut_0 = 1, ValidOut_1 = 1, ValidOut_2 = 1, SoftReset_0 = 0, SoftReset_1 = 0, SoftReset_2 = 0
# Inputs :  Detect Address = 0, Data_in = 10, Write Enable Reg = 0, Empty_0 = 0, Empty_1 = 0, Empty_2 = 0, Full_0 = 0, Full_1 = 1, Full_2 = 0, ReadEnable_0 = 0, ReadEnable_1 = 0,
ReadEnable_2 = 0
# Outputs :  Fifo Address = 10, Write Enable = 000,Fifo Full = 0, ValidOut_0 = 1, ValidOut_1 = 1, ValidOut_2 = 1, SoftReset_0 = 0, SoftReset_1 = 0, SoftReset_2 = 0
# Inputs :  Detect Address = 0, Data_in = 10, Write Enable Reg = 0, Empty_0 = 0, Empty_1 = 0, Empty_2 = 0, Full_0 = 0, Full_1 = 1, Full_2 = 0, ReadEnable_0 = 0, ReadEnable_1 = 0,
ReadEnable_2 = 0
# Outputs :  Fifo Address = 10, Write Enable = 000,Fifo Full = 0, ValidOut_0 = 1, ValidOut_1 = 1, ValidOut_2 = 1, SoftReset_0 = 1, SoftReset_1 = 0, SoftReset_2 = 0
# Inputs :  Detect Address = 0, Data_in = 10, Write Enable Reg = 0, Empty_0 = 0, Empty_1 = 0, Empty_2 = 0, Full_0 = 0, Full_1 = 1, Full_2 = 0, ReadEnable_0 = 0, ReadEnable_1 = 0,
ReadEnable_2 = 0
# Outputs :  Fifo Address = 10, Write Enable = 000,Fifo Full = 0, ValidOut_0 = 1, ValidOut_1 = 1, ValidOut_2 = 1, SoftReset_0 = 0, SoftReset_1 = 0, SoftReset_2 = 0
```

3. Sub-Block : Router Controller

Design Code:

```
module router_fsm(input clock,
                        resetn,
                        pkt_valid,
                        parity_done,
                        soft_reset_0,
                        soft_reset_1,
                        soft_reset_2,
                        fifo_full,
                        low_pkt_valid,
                        fifo_empty_0,
                        fifo_empty_1,
                        fifo_empty_2,
                  input [1:0] data_in,
                  output busy,
                         detect_add,
                         ld_state,
                         laf_state,
                         full_state,
                         write_enb_reg,
                         rst_int_reg,
                         lfd_state);

    parameter DECODE_ADDRESS = 3'b000,
              LOAD_FIRST_DATA = 3'b001,
              LOAD_DATA = 3'b010,
              FIFO_FULL_STATE =3'b011,
              LOAD_AFTER_FULL = 3'b100,
              LOAD_PARITY = 3'b101,
              CHECK_PARITY_ERROR = 3'b110,
              WAIT_TILL_EMPTY = 3'b111;

    reg [1:0] addr;
    reg [2:0] present_state, next_state;

    // Present State Logic
    always@(posedge clock)
      begin
        if(!resetn)
          present_state <= DECODE_ADDRESS;
        else if((soft_reset_0 && data_in == 2'b00)|| (soft_reset_1 && data_in == 2'b01)
|| (soft_reset_2 && data_in == 2'b10))
          present_state <= DECODE_ADDRESS;
        else
          present_state <= next_state;
      end

    // Logic for storing the address
    always@(posedge clock)
      begin
        if(!resetn)
         addr <= 0;
        else if((soft_reset_0 && data_in == 2'b00)|| (soft_reset_1 && data_in == 2'b01)
|| (soft_reset_2 && data_in == 2'b10))
         addr <= 0;
        if(present_state == DECODE_ADDRESS && data_in!= 2'b11)
         addr <= data_in;
      end

    // Next State Logic
    always@(*)
      begin
        if(addr != 2'b11)
```

```
        begin
          next_state = DECODE_ADDRESS;
          case(present_state)
          DECODE_ADDRESS:          begin
                                 if((pkt_valid && (data_in[1:0]==0) && fifo_empty_0) ||
                                    (pkt_valid && (data_in[1:0]==1) && fifo_empty_1) ||
                                    (pkt_valid && (data_in[1:0]==2) && fifo_empty_2))
                                       next_state = LOAD_FIRST_DATA;
                               else if((pkt_valid && (data_in[1:0]==0) && !fifo_empty_0) ||
                                   (pkt_valid && (data_in[1:0]==1) && !fifo_empty_1) ||
                                    (pkt_valid && (data_in[1:0]==2) && !fifo_empty_2))
                                       next_state = WAIT_TILL_EMPTY;
                                    end
          LOAD_FIRST_DATA:         next_state = LOAD_DATA;
          LOAD_DATA:               begin
                                     if(fifo_full)
                                       next_state = FIFO_FULL_STATE;
                                     else if(!fifo_full && !pkt_valid)
                                       next_state = LOAD_PARITY;
                                    else
                                       next_state = LOAD_DATA;
                                    end
          FIFO_FULL_STATE:         begin
                                     if(!fifo_full)
                                       next_state = LOAD_AFTER_FULL;
                                     else
                                       next_state = FIFO_FULL_STATE;
                                    end
          LOAD_AFTER_FULL:         begin
                                     if(!parity_done && !low_pkt_valid)
                                       next_state = LOAD_DATA;
                                     else if(!parity_done && low_pkt_valid)
                                       next_state = LOAD_PARITY;
                                     else if(parity_done)
                                       next_state = DECODE_ADDRESS;
                                    end
          LOAD_PARITY :            next_state = CHECK_PARITY_ERROR;
          CHECK_PARITY_ERROR:      if(fifo_full)
                                      next_state = FIFO_FULL_STATE;
          WAIT_TILL_EMPTY:         begin
                                     if((fifo_empty_0 && (addr == 0)) ||
                                        (fifo_empty_1 && (addr == 1)) ||
                                        (fifo_empty_2 && (addr == 2)))
                                        next_state = LOAD_FIRST_DATA;
                                     else
                                        next_state = WAIT_TILL_EMPTY;
                                    end
          default :                next_state = DECODE_ADDRESS;
          endcase
      end


      else
        next_state = DECODE_ADDRESS;
    end

  // Logic for output logic
  assign detect_add = (present_state == DECODE_ADDRESS) ? 1'b1 : 1'b0;
  assign ld_state = (present_state == LOAD_DATA) ? 1'b1 : 1'b0;
  assign lfd_state = (present_state == LOAD_FIRST_DATA) ? 1'b1 : 1'b0;
  assign laf_state = (present_state == LOAD_AFTER_FULL) ? 1'b1 : 1'b0;
  assign full_state = (present_state == FIFO_FULL_STATE) ? 1'b1 : 1'b0;
  assign write_enb_reg = ((present_state == LOAD_DATA) ||
                          (present_state == LOAD_AFTER_FULL) ||
                        (present_state == LOAD_PARITY)) ? 1'b1 : 1'b0;
  assign rst_int_reg = (present_state == CHECK_PARITY_ERROR) ? 1'b1 : 1'b0;
  assign busy = ((present_state == LOAD_DATA) || (present_state == DECODE_ADDRESS)) ?
```

```
1'b0 : 1'b1;
```

```
endmodule
```

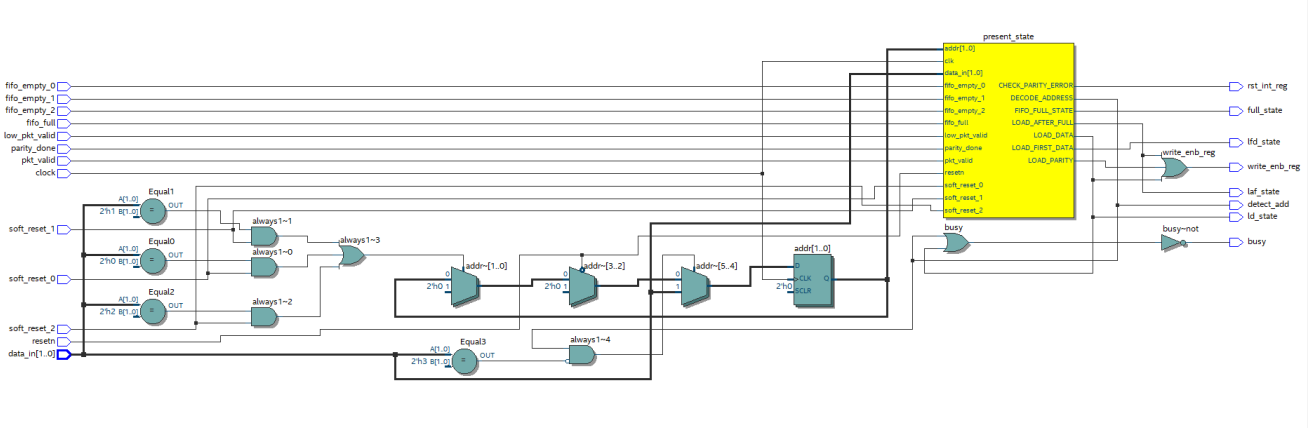
Simulation Code:

```
module router_fsm_tb();
    reg clock_tb,
resetn_tb,pkt_valid_tb,parity_done_tb,soft_reset_0_tb,soft_reset_1_tb,soft_reset_2_tb,f
ifo_full_tb,low_pkt_valid_tb,fifo_empty_0_tb,fifo_empty_1_tb,fifo_empty_2_tb;
    reg [1:0] data_in_tb;
    wire
busy_tb,detect_add_tb,ld_state_tb,laf_state_tb,full_state_tb,write_enb_reg_tb,rst_int_r
eg_tb,lfd_state_tb;

    parameter T = 10, ADDRESS = 2'd1;

    router_fsm DUT(clock_tb,
                   resetn_tb,
                   pkt_valid_tb,
                   parity_done_tb,
                   soft_reset_0_tb,
                   soft_reset_1_tb,
                   soft_reset_2_tb,
                   fifo_full_tb,
                   low_pkt_valid_tb,
                   fifo_empty_0_tb,
                   fifo_empty_1_tb,
                   fifo_empty_2_tb,
                   data_in_tb,
                   busy_tb,
                   detect_add_tb,
                   ld_state_tb,
                   laf_state_tb,
                   full_state_tb,
                   write_enb_reg_tb,
                   rst_int_reg_tb,
                   lfd_state_tb);


    initial
      begin
        clock_tb = 1'b0;
        forever #(T/2) clock_tb = ~clock_tb;
      end

    task resetn_ip;
      begin
        @(negedge clock_tb);
        resetn_tb = 1'b0;
        @(negedge clock_tb);
        resetn_tb = 1'b1;
      end
    endtask

    task soft_reset_ip(input [2:0] j);
      begin
        @(negedge clock_tb);
        soft_reset_0_tb = j[0];
        soft_reset_1_tb = j[1];
        soft_reset_2_tb = j[2];
        @(negedge clock_tb);
        soft_reset_0_tb = 0;
        soft_reset_1_tb = 0;
        soft_reset_2_tb = 0;
      end
    endtask

    task initialize;
      begin
```

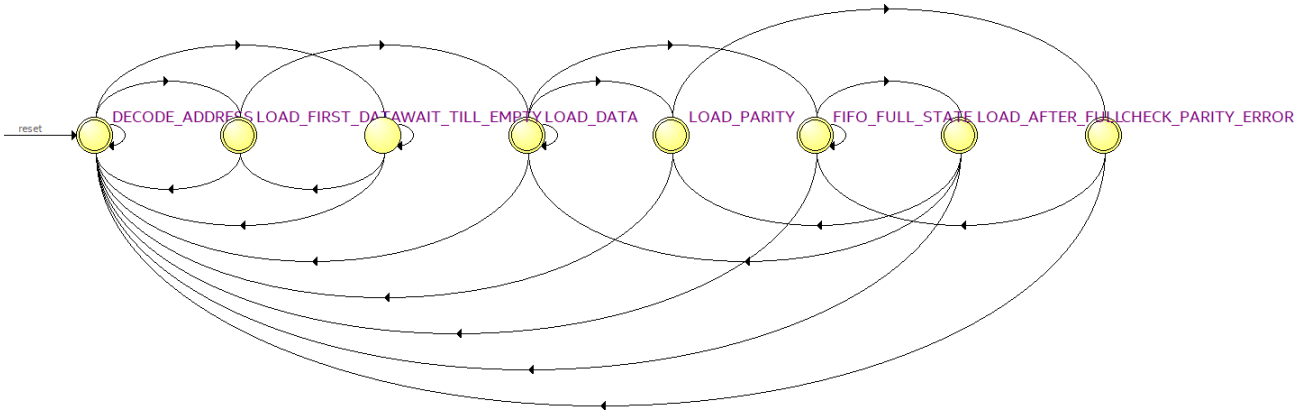
```
      resetn_tb = 1'b1;
      pkt_valid_tb = 1'b0;
      parity_done_tb = 1'b0;
      soft_reset_0_tb = 1'b0;
      soft_reset_1_tb = 1'b0;
      soft_reset_2_tb = 1'b0;
      fifo_full_tb = 1'b0;
      low_pkt_valid_tb = 1'b0;
      fifo_empty_0_tb = 1'b0;
      fifo_empty_1_tb = 1'b0;
      fifo_empty_2_tb = 1'b0;
    end
  endtask

  task DA_LFD_LD_LP_CPE_DA;
    begin
      pkt_valid_tb = 1'b1;
      data_in_tb = 2'b01;
      fifo_empty_1_tb  =  1'b1;
      #20 pkt_valid_tb = 1'b0;
      low_pkt_valid_tb = 1'b1;
      #20 fifo_empty_1_tb = 1'b0;
      low_pkt_valid_tb = 1'b0;
    end
  endtask

  task DA_LFD_LD_FFS_LAF_LP_CPE_DA;
    begin
      pkt_valid_tb = 1'b1;
      data_in_tb = 2'b10;
      fifo_empty_2_tb = 1'b1;
      #40 fifo_full_tb = 1'b1;
      pkt_valid_tb = 1'b0;
      low_pkt_valid_tb = 1'b1;
      #30 fifo_full_tb = 1'b0;
      parity_done_tb = 1'b0;
      #50 low_pkt_valid_tb = 1'b0;
      fifo_empty_2_tb = 1'b0;
    end
  endtask

  task DA_LFD_LD_FFS_LAF_LD_LP_CPE_DA;
    begin
      pkt_valid_tb = 1'b1;
      data_in_tb = 2'b00;
      fifo_empty_0_tb  =  1'b1;
      #40 fifo_full_tb = 1'b1;
      #30 fifo_full_tb = 1'b0;
      #30 pkt_valid_tb = 1'b0;
      low_pkt_valid_tb = 1'b1;
      #20 fifo_empty_0_tb = 1'b0;
      low_pkt_valid_tb = 1'b0;
    end
  endtask

  task DA_LFD_LD_LP_CPE_FFS_LAF_DA;
    begin
      pkt_valid_tb = 1'b1;
      data_in_tb = 2'b10;
      fifo_empty_2_tb = 1'b1;
      #50 pkt_valid_tb = 1'b0;
      low_pkt_valid_tb = 1'b1;
      #10 fifo_full_tb = 1'b1;
      #10 parity_done_tb = 1'b1;
      #20 fifo_full_tb = 1'b0;
      #50 parity_done_tb = 1'b0;
      low_pkt_valid_tb = 1'b0;
```

```
        fifo_empty_2_tb = 1'b0;
      end
    endtask

    task DA_WTE_LFD_LD_LP_CPE_DA;
      begin
        pkt_valid_tb = 1'b1;
        data_in_tb = 2'b00;
        fifo_empty_0_tb = 1'b0;
        #40 fifo_empty_0_tb = 1'b1;
        #20 pkt_valid_tb = 1'b0;
        low_pkt_valid_tb = 1'b1;
        #20 fifo_empty_0_tb = 1'b0;
        low_pkt_valid_tb = 1'b0;
      end
    endtask

    initial
      begin
        initialize;
        #20;
        resetn_ip;
        #20;
        data_in_tb = ADDRESS;
        soft_reset_ip(3'b010);
        #20;
        DA_LFD_LD_LP_CPE_DA;
        #100;
        DA_WTE_LFD_LD_LP_CPE_DA;
        #100;
        DA_LFD_LD_FFS_LAF_LP_CPE_DA;
        #100;
        DA_LFD_LD_FFS_LAF_LD_LP_CPE_DA;
        #100;
        DA_LFD_LD_LP_CPE_FFS_LAF_DA;
        #100;
        $finish;
      end

endmodule
```

Synthesis Circuit:

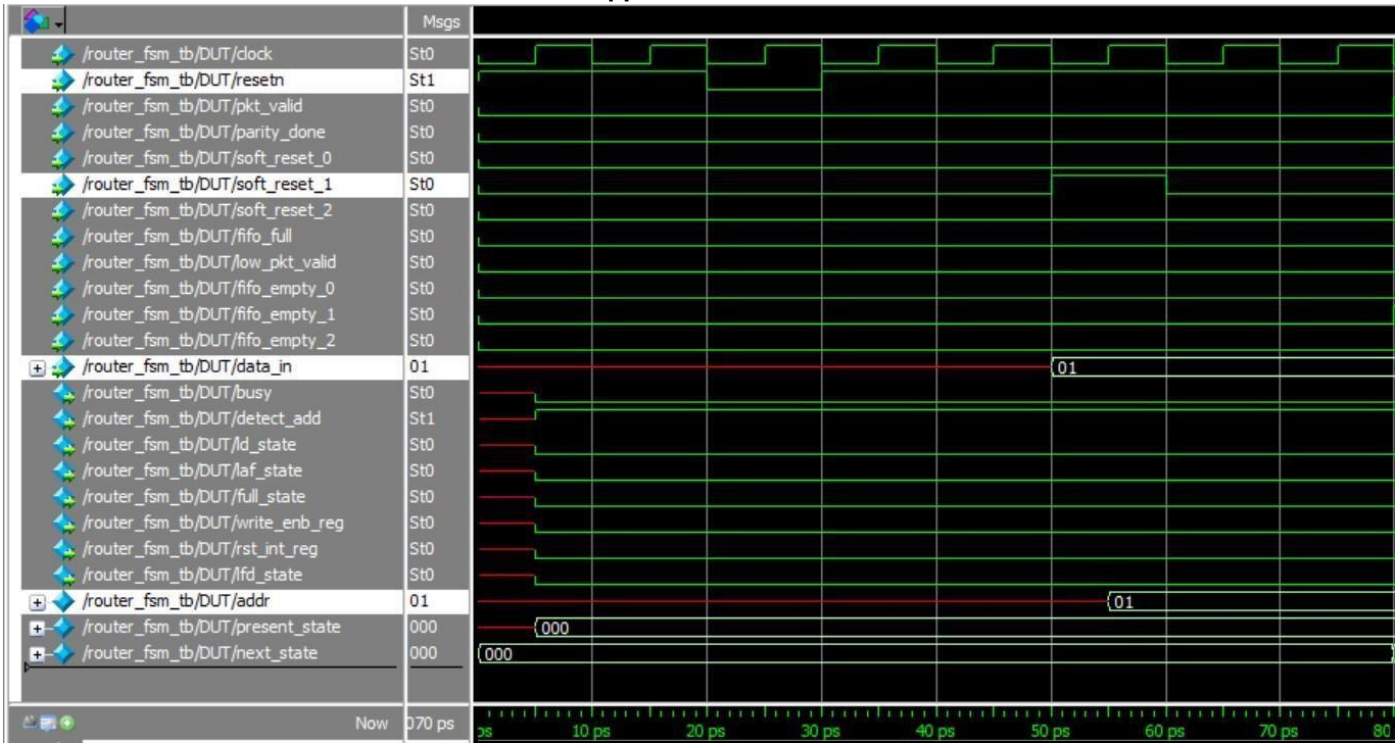


State Transition Diagram:

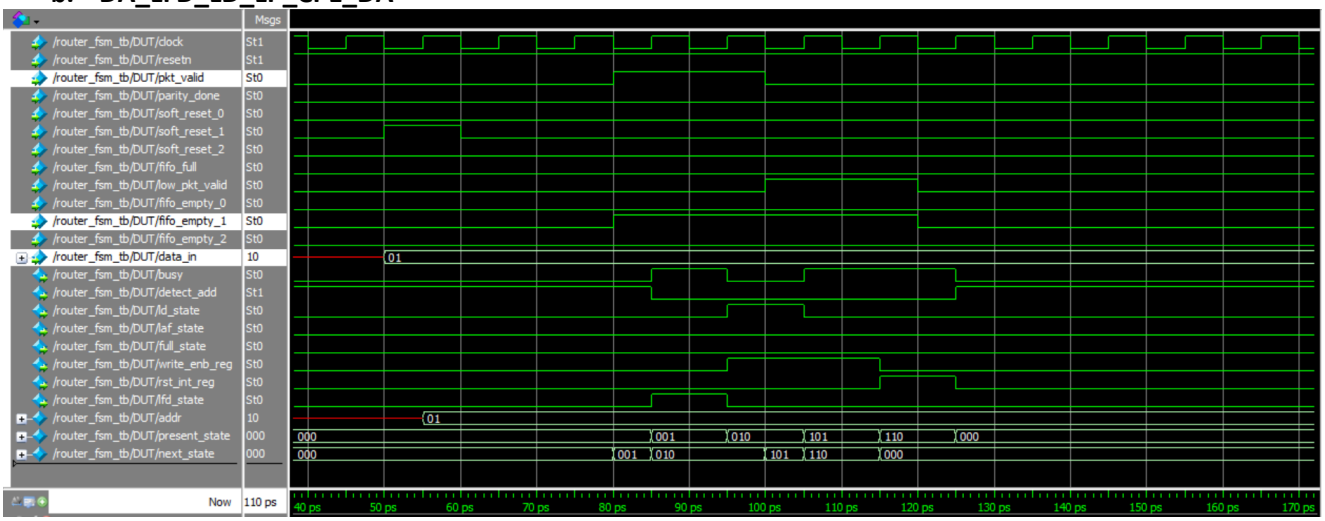


Simulation Waveform:

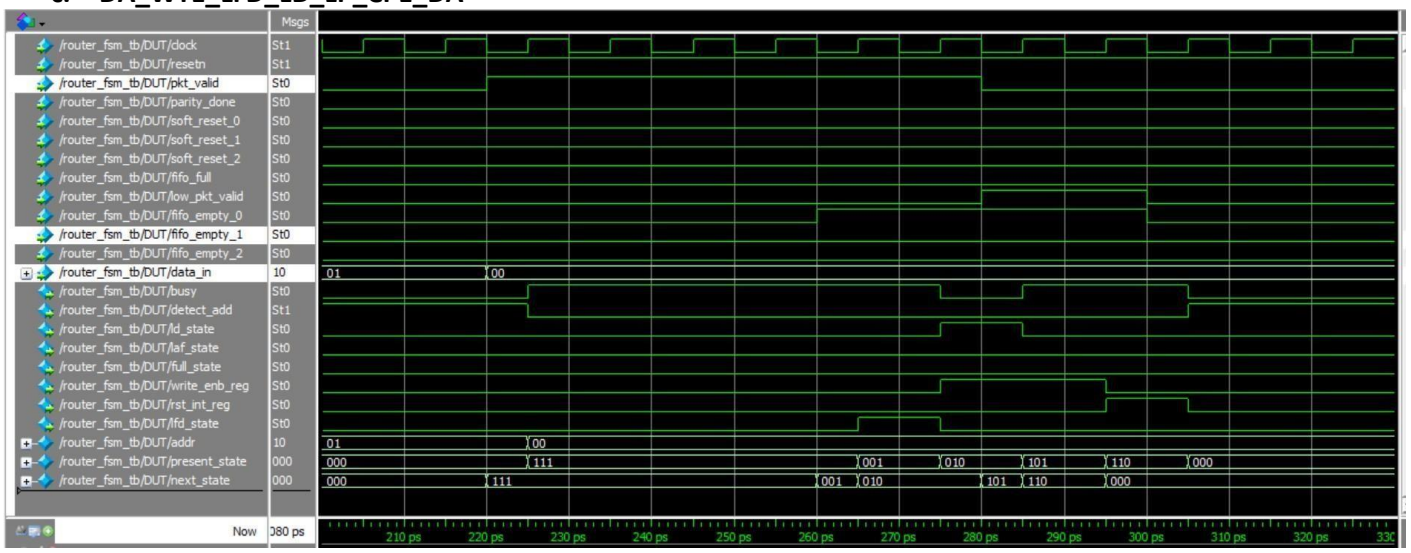
a. **When active low reset and soft reset are applied**



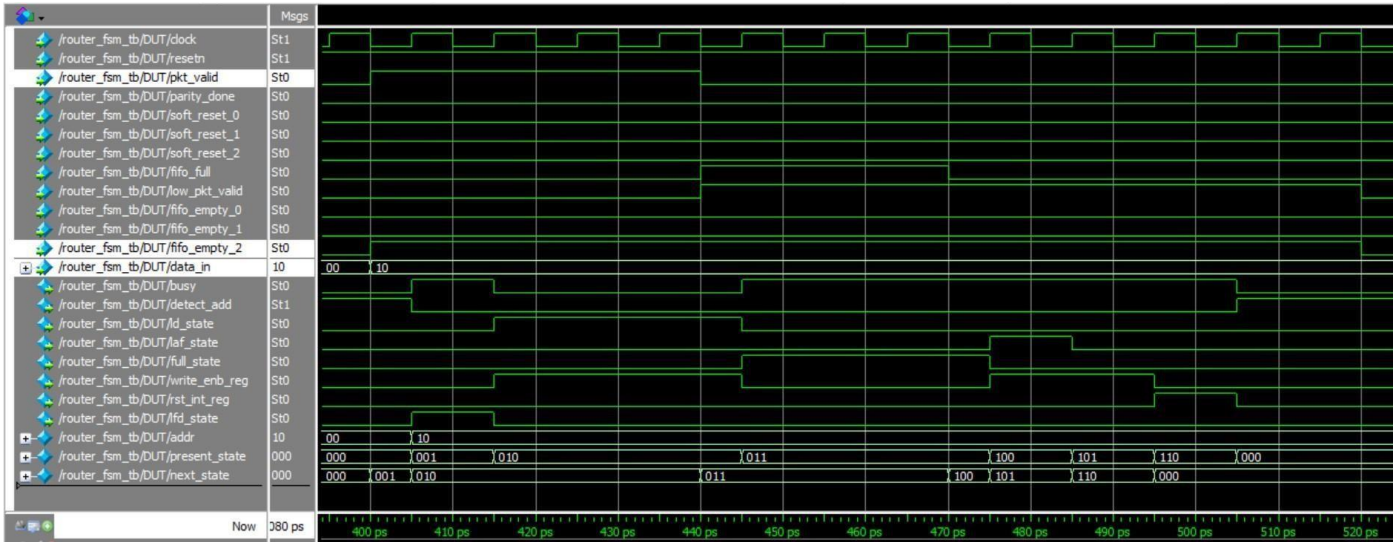
b. **DA_LFD_LD_LP_CPE_DA**



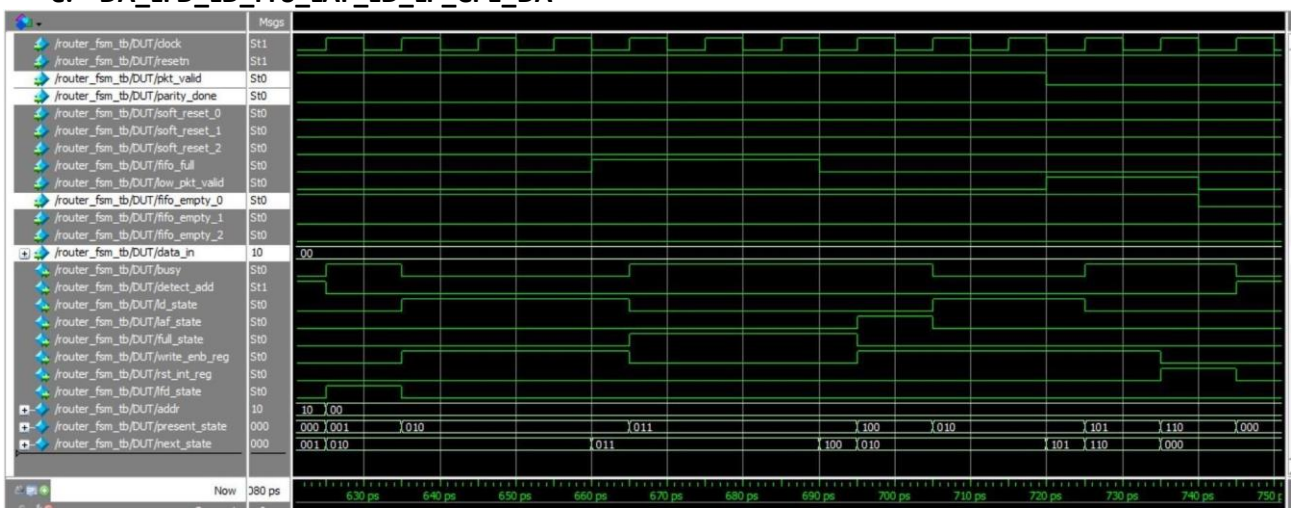
c. DA_WTE_LFD_LD_LP_CPE_DA



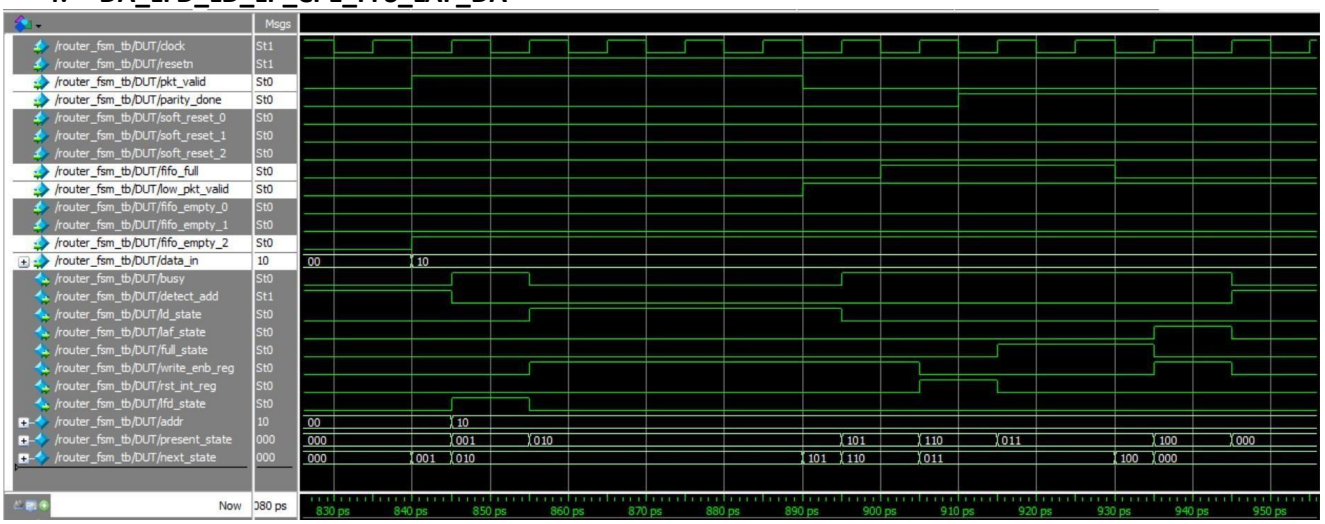
d. DA_LFD_LD_FFS_LAF_LP_CPE_DA



e. DA_LFD_LD_FFS_LAF_LD_LP_CPE_DA



f. DA_LFD_LD_LP_CPE_FFS_LAF_DA



4. Sub-Block : Router Register

Design Code:

```
module router_reg (input clock,
                           resetn,
                           pkt_valid,
                           fifo_full,
                           rst_int_reg,
                           detect_add,
                           ld_state,
                           laf_state,
                           full_state,
                           lfd_state,
                   input [7:0] data_in,
                   output reg parity_done,
                          low_pkt_valid,
                          err,
                   output reg [7:0] dout);

    reg [7:0] header_byte, fifo_full_state_byte, internal_parity, packet_parity;

    // Logic to store header byte and FIFO full state byte
    always@(posedge clock)
      begin
        if(!resetn)
          begin
            header_byte <= 0;
            fifo_full_state_byte <= 0;
          end
        else if(detect_add && pkt_valid)
          header_byte <= data_in;
        else if(ld_state && fifo_full)
          fifo_full_state_byte <= data_in;
      end

    // Logic for dout
    always@(posedge clock)
      begin
        if(!resetn)
          dout <= 0;
        else if(lfd_state)
          dout <= header_byte;
        else if(laf_state)
          dout <= fifo_full_state_byte;
        else if(ld_state && !fifo_full)
          dout <= data_in;
      end

    // Logic for parity_done
    always@(posedge clock)
      begin
        if(!resetn)
          parity_done <= 0;
        else if(detect_add)
          parity_done <= 0;
        else if((ld_state && !pkt_valid && !fifo_full) || (laf_state && low_pkt_valid
&& !parity_done))
          parity_done <= 1'b1;
      end

    // Logic for low_pkt_valid
    always@(posedge clock)
      begin
        if(!resetn)
          low_pkt_valid <= 0;
```

```
        else if(ld_state && !pkt_valid)
          low_pkt_valid <= 1'b1;
        else if(rst_int_reg)
          low_pkt_valid <= 0;
      end

    // Logic for internal_parity
    always@(posedge clock)
      begin
        if(!resetn)
          internal_parity <= 0;
        else if(detect_add)
          internal_parity <= 0;
        else if(lfd_state)
          internal_parity <= internal_parity ^ header_byte;
        else if(ld_state && pkt_valid && !full_state)
          internal_parity <= internal_parity ^ data_in;
      end

    // Logic for packet_parity
    always@(posedge clock)
      begin
        if(!resetn)
          packet_parity <= 0;
        else if((ld_state && !pkt_valid && !fifo_full) || (laf_state && low_pkt_valid
&& !parity_done))
          packet_parity <= data_in;
      end

    // Logic for err
    always@(posedge clock)
      begin
        if(!resetn)
          err <= 0;
        else if(parity_done && (internal_parity != packet_parity))
          err <= 1'b1;
        else
          err <= 0;
      end
endmodule
```

Simulation Code:

```
module router_reg_tb();
    reg
clock,resetn,pkt_valid,fifo_full,rst_int_reg,detect_add,ld_state,laf_state,full_state,l
fd_state;
    reg [7:0] data_in;
    wire parity_done,low_pkt_valid,err;
    wire [7:0] dout;

    parameter T = 10;

    integer i;

    router_reg DUT (clock,
                    resetn,
                    pkt_valid,
                    fifo_full,
                    rst_int_reg,
                    detect_add,
                    ld_state,
                    laf_state,
                    full_state,
                    lfd_state,
                    data_in,
                    parity_done,
                    low_pkt_valid,
```

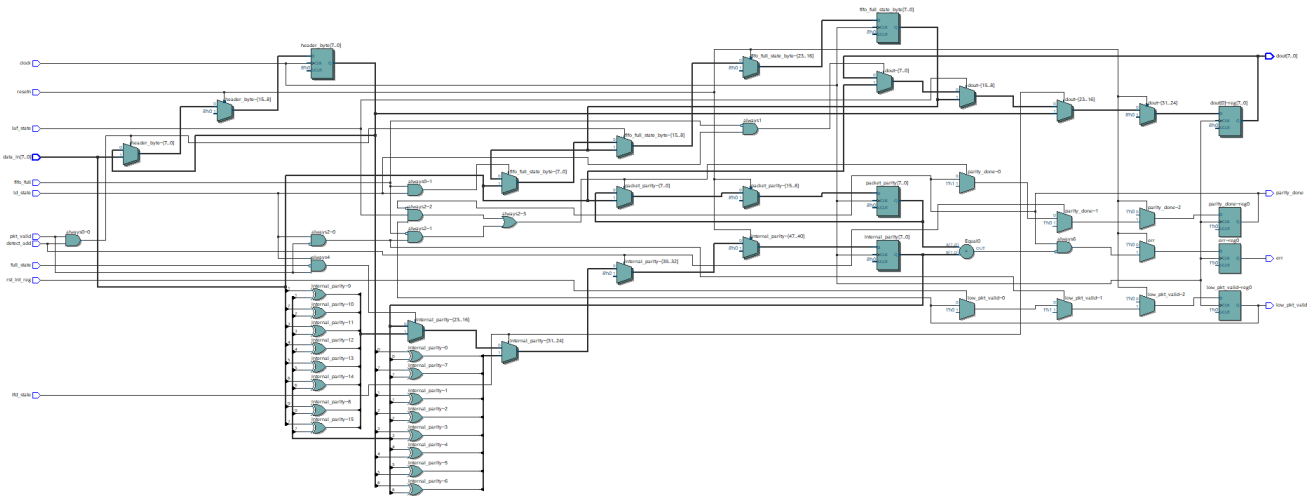
```
                err,
                dout);

  initial
    begin

      clock = 1'b0;
      forever #(T/2) clock = ~clock;
    end

  task initialize;
    begin
      resetn = 1'b1;
      pkt_valid =1'b0;
      fifo_full = 1'b0;
      rst_int_reg = 1'b0;
      detect_add = 1'b0;
      ld_state = 1'b0;
      laf_state = 1'b0;
      full_state = 1'b0;
      lfd_state = 1'b0;
      data_in = 0;
    end
  endtask

  task resetn_ip;
    begin
      @(negedge clock);
      resetn = 1'b0;
      @(negedge clock);
      resetn = 1'b1;
    end
  endtask

  task good_packet;
    reg [7:0] header, payload, parity;
    reg [5:0] payload_len;
    reg [1:0] addr;
    begin
      @(negedge clock);
      parity = 0;
      payload_len = 6'd4;
      addr = 2'd1;
      header = {payload_len,addr};
      pkt_valid =  1'b1;
      detect_add = 1'b1;
      parity = parity ^ header;
      data_in = header;
      @(negedge clock);
      detect_add  =  0;
      lfd_state = 1'b1;
      for(i=0;i<payload_len;i=i+1)
        begin
          @(negedge clock);
          lfd_state = 0;
          ld_state = 1;
          payload = {$random} % 256;
          parity = parity ^ payload;
          data_in = payload;
        end
      @(negedge clock);
      pkt_valid = 0;
      data_in = parity;
      @(negedge clock);
      ld_state = 0;
      rst_int_reg = 1'b1;
      @(negedge clock);
```

```
        rst_int_reg = 1'b0;
```

```
      detect_add = 1'b1;
    end
  endtask

  task overflow_packet;
    reg [7:0] header, payload, parity;
    reg [5:0] payload_len;
    reg [1:0] addr;
    begin
      @(negedge clock);
      parity = 0;
      payload_len = 6'd16;
      addr = 2'd2;
      header = {payload_len,addr};
      pkt_valid =  1'b1;
      detect_add = 1'b1;
      parity = parity ^ header;
      data_in = header;
      @(negedge clock);
      detect_add  =  0;
      lfd_state = 1'b1;
      for(i=0;i<15;i=i+1)
        begin
          @(negedge clock);
          lfd_state = 0;
          ld_state = 1;
          payload = {$random} % 256;
          parity = parity ^ payload;
          data_in = payload;
        end
      @(negedge clock);
      payload = 8'b11110000;
      parity = parity ^ payload;
      data_in = payload;
      fifo_full = 1'b1;
      full_state = 1'b1;
      #30 fifo_full = 0;
      full_state = 0;
      ld_state = 0;
      @(negedge clock);
      laf_state = 1'b1;
      @(negedge clock);
      laf_state = 0;
      ld_state = 1'b1;
      @(negedge clock);
      pkt_valid = 0;
      data_in = parity;
      @(negedge clock);
      ld_state = 0;
      rst_int_reg = 1'b1;
      @(negedge clock);
      rst_int_reg = 1'b0;
      detect_add = 1'b1;
    end
  endtask

  task corrupted_packet;
    reg [7:0] header, payload, parity;
    reg [5:0] payload_len;
    reg [1:0] addr;
    begin
      @(negedge clock);
      parity = 0;
      payload_len = 6'd4;
      addr = 2'd1;
      header = {payload_len,addr};
      pkt_valid = 1'b1;
```

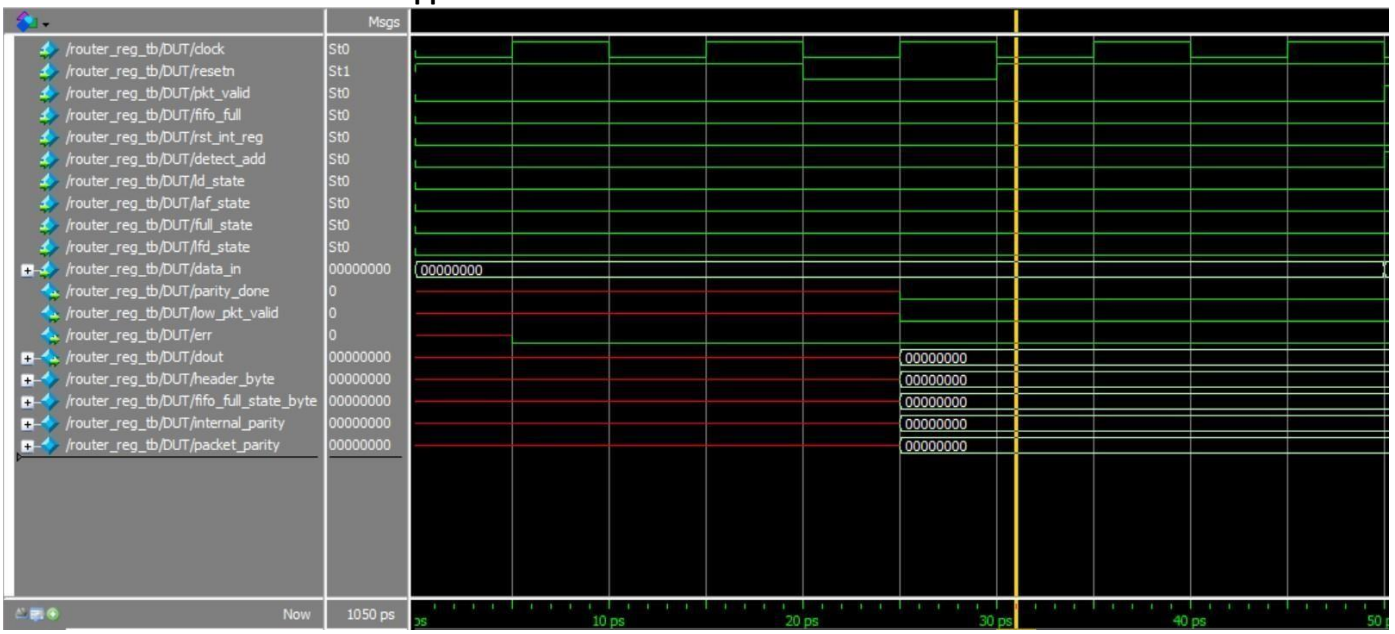
```
        detect_add = 1'b1;
        parity = parity ^ header;
        data_in = header;
        @(negedge clock);
        detect_add  =  0;
        lfd_state = 1'b1;
        for(i=0;i<payload_len;i=i+1)
          begin
            @(negedge clock);
            lfd_state = 0;
            ld_state = 1;
            payload = {$random} % 256;
            parity = parity ^ payload;
            data_in = payload;
          end
        @(negedge clock);
        pkt_valid = 0;
        data_in = ~parity;
        @(negedge clock);
        ld_state = 0;
        rst_int_reg = 1'b1;
        @(negedge clock);
        rst_int_reg = 1'b0;
        detect_add = 1'b1;
      end
    endtask

    initial
      begin
        initialize;
        #20;
        resetn_ip;
        #20;
        good_packet;
        #200;
        corrupted_packet;
        #200;
        overflow_packet;
        #200;
        $finish;
      end
endmodule
```

Synthesis Circuit:

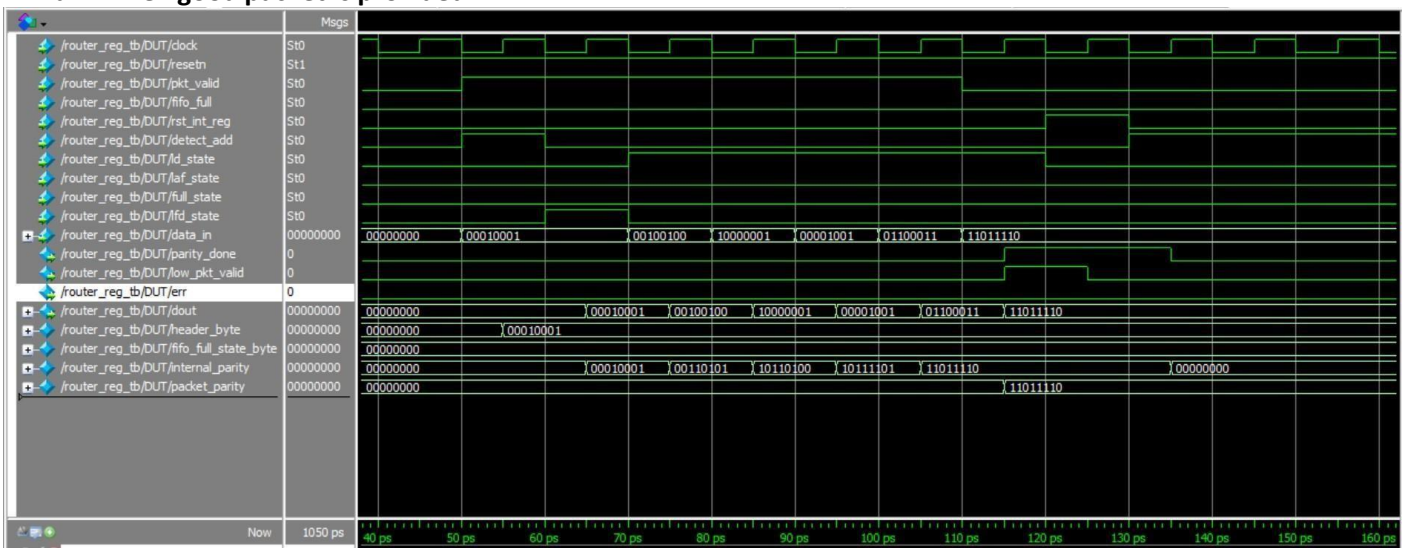


Simulation Waveform:

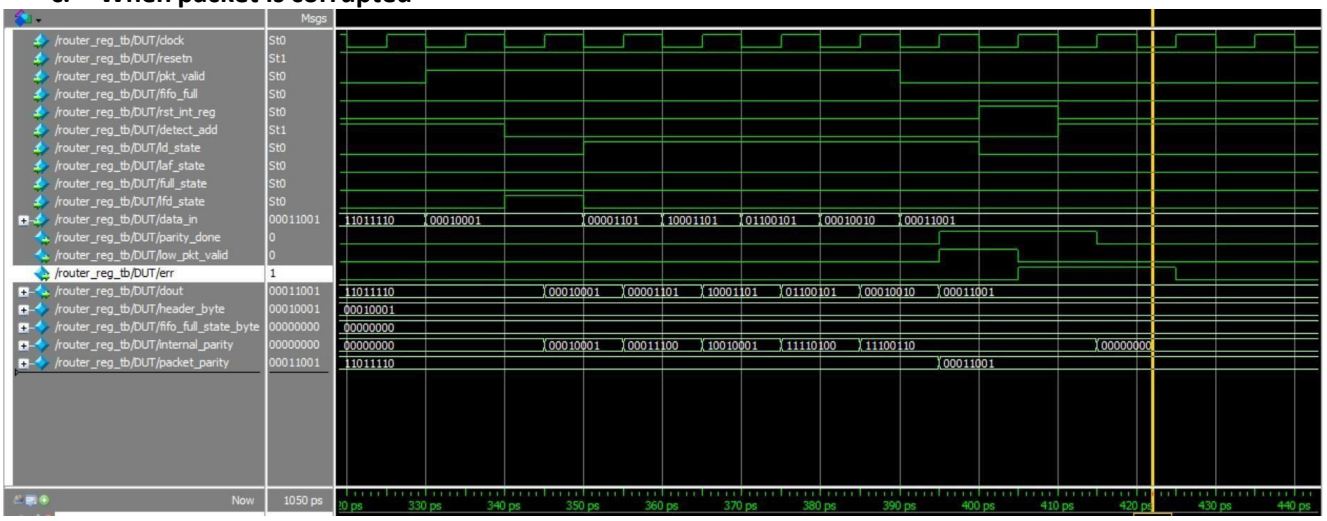
a. **When active low reset is applied**



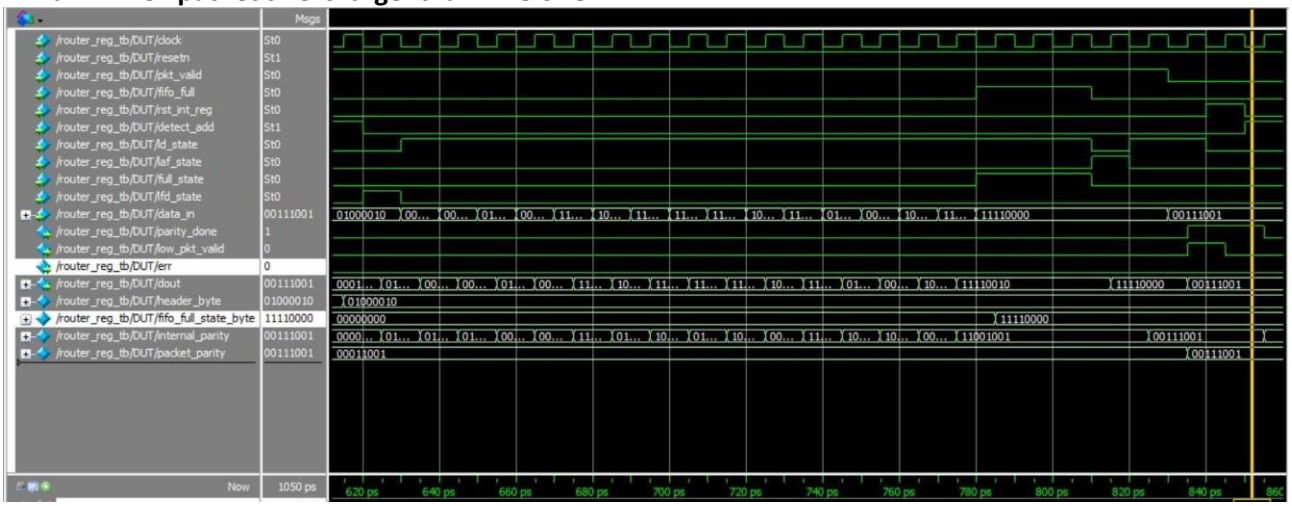
b. **When good packet is provided**



c. **When packet is corrupted**



d. When packet size is larger than FIFO size



5. Router Top Block

Design Code:

```
module router_top(input clock,
                        resetn,
                        read_enb_0,
                        read_enb_1,
                        read_enb_2,
                        pkt_valid,
                  input [7:0] data_in,
                  output vld_out_0,
                         vld_out_1,
                         vld_out_2,
                         err,
                         busy,
                  output [7:0] data_out_0,
                               data_out_1,
                               data_out_2);

    wire [2:0] write_enb;
    wire [7:0] dout;
    wire
parity_done,soft_reset_0,soft_reset_1,soft_reset_2,fifo_full,low_pkt_valid,detect_add,l
d_state,laf_state,full_state,write_enb_reg,rst_int_reg,lfd_state,empty_0,empty_1,empty_
2,full_0,full_1,full_2;

    router_fsm
FSM(clock,resetn,pkt_valid,parity_done,soft_reset_0,soft_reset_1,soft_reset_2,fifo_full
,low_pkt_valid,empty_0,empty_1,empty_2,data_in[1:0],busy,detect_add,ld_state,laf_state,
full_state,write_enb_reg,rst_int_reg,lfd_state);

    router_sync
Synchronizer(clock,resetn,detect_add,write_enb_reg,read_enb_0,read_enb_1,read_enb_2,emp
ty_0,empty_1,empty_2,full_0,full_1,full_2,data_in[1:0],fifo_full,soft_reset_0,soft_rese
t_1,soft_reset_2,vld_out_0,vld_out_1,vld_out_2,write_enb);

    router_reg
Register(clock,resetn,pkt_valid,fifo_full,rst_int_reg,detect_add,ld_state,laf_state,ful
l_state,lfd_state,data_in,parity_done,low_pkt_valid,err,dout);

    router_fifo
FIFO_0(clock,resetn,write_enb[0],soft_reset_0,read_enb_0,lfd_state,dout,empty_0,
full_0,data_out_0);

    router_fifo
FIFO_1(clock,resetn,write_enb[1],soft_reset_1,read_enb_1,lfd_state,dout,empty_1,
full_1,data_out_1);

    router_fifo
FIFO_2(clock,resetn,write_enb[2],soft_reset_2,read_enb_2,lfd_state,dout,empty_2,
full_2,data_out_2);

endmodule
```

Simulation Code:

```
module router_top_tb();
    reg clock,resetn,read_enb_0,read_enb_1,read_enb_2,pkt_valid;
    reg [7:0] data_in;
    wire vld_out_0,vld_out_1,vld_out_2,err,busy;
    wire [7:0] data_out_0,data_out_1,data_out_2;
```

```
event e1,e2;

parameter T = 10;

router_top DUT(clock,
               resetn,
               read_enb_0,
               read_enb_1,
               read_enb_2,
               pkt_valid,
               data_in,
               vld_out_0,
               vld_out_1,
               vld_out_2,
               err,
               busy,
               data_out_0,
               data_out_1,
               data_out_2);

initial
  begin
    clock = 1'b0;
    forever #(T/2) clock = ~clock;
  end

task initialize;
  begin
    resetn = 1'b1;
    read_enb_0 = 0;
    read_enb_1 = 0;
    read_enb_2 = 0;
    pkt_valid = 0;
    data_in = 0;
  end
endtask

task reset_ip;
  begin
    @(negedge clock);
    resetn = 1'b0;
    @(negedge clock);
    resetn = 1'b1;
  end
endtask

task payload_14_15(input [5:0]d1,input [1:0]a1);
  reg [7:0] header, payload, parity;
  reg [5:0] payload_len;
  reg [1:0] addr;
  integer i;
    begin
        @(negedge clock);
        wait(!busy)
        @(negedge clock);
        parity = 0;
        payload_len = d1;
        addr = a1;
        header = {payload_len,addr};
        parity = parity ^ header;
        pkt_valid = 1'b1;
        data_in = header;
        @(negedge clock);
        wait(!busy)
        for(i=0;i<payload_len;i=i+1)
          begin
            @(negedge clock);
```

```
            wait(!busy)
            payload = i;
            parity = parity ^ payload;
            data_in = payload;
          end
        wait(!busy)
        @(negedge clock);
        pkt_valid = 0;
        data_in = parity;

    end
endtask



task payload_16_17(input [5:0]d2,input [1:0]a2);
  reg [7:0] header, payload, parity;
  reg [5:0] payload_len;
  reg [1:0] addr;
  integer i;
    begin
        @(negedge clock);
        wait(!busy)
        parity = 0;
        payload_len = d2;
        addr = a2;
        header = {payload_len,addr};
        parity = parity ^ header;
        @(negedge clock);
        pkt_valid = 1'b1;
        data_in = header;
        @(negedge clock);
        wait(!busy)
        for(i=0;i<payload_len;i=i+1)
          begin
            @(negedge clock);
            wait(!busy)
            payload = i;
            parity = parity ^ payload;
            data_in = payload;
          end
        ->e1;
        wait(!busy)
        @(negedge clock);
        pkt_valid = 0;
        data_in = parity;
    end
endtask

task random_packet(input [5:0]d3,input [1:0]a3);
  reg [7:0] header, payload, parity;
  reg [5:0] payload_len;
  reg [1:0] addr;
  integer i;
    begin
        ->e2;
        @(negedge clock);
        wait(!busy)
        parity = 0;
        payload_len = d3;
        addr = a3;
        header = {payload_len,addr};
        parity = parity ^ header;
        @(negedge clock);
        pkt_valid = 1'b1;
        data_in = header;
        @(negedge clock);
        wait(!busy)
```

```
        for(i=0;i<payload_len;i=i+1)
```

```
begin
```

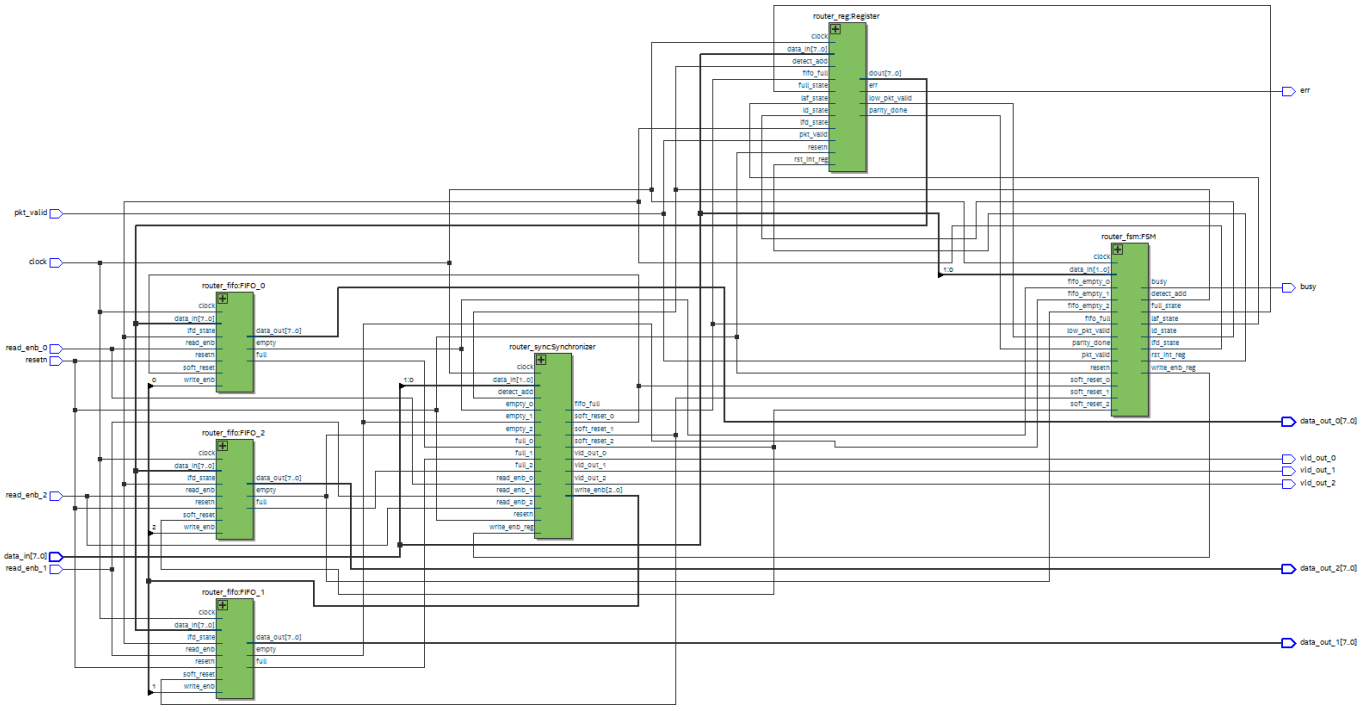
```
                @(negedge clock);
                wait(!busy)
                payload = i;
                parity = parity ^ payload;
                data_in = payload;
              end
            wait(!busy)
            @(negedge clock);
            pkt_valid = 0;
            data_in = parity;

      end
endtask



initial
  begin
   forever
    begin
    @(e1)
    begin
        @(negedge clock);
        read_enb_1 = 1'b1;
        wait(!vld_out_1)
        @(negedge clock);
        read_enb_1 = 0;
    end
    end
  end

initial
  begin
    forever
     begin
      @(e2)
      begin
        wait(vld_out_2)
        @(negedge clock);
        read_enb_2 = 1'b1;
        wait(!vld_out_2)
        @(negedge clock);
        read_enb_2 = 0;
      end
     end
  end

initial
  begin
    initialize;
    reset_ip;
    repeat (2) @(negedge clock);

    // Packet Size = 16bytes
    // DA-LFD-LD-LP-CPE-DA
    payload_14_15(6'd14,2'd0);
    @(negedge clock);
    read_enb_0 = 1'b1;
    wait(!vld_out_0)
    @(negedge clock);
    read_enb_0 = 0;
    #100;

    // Packet Size = 17bytes
    // DA-LFD-LD-LP-CPE-FFS-LAF-DA
    payload_14_15(6'd15,2'd0);
    @(negedge clock);
    wait(busy)
```

```
        @(negedge clock);
```

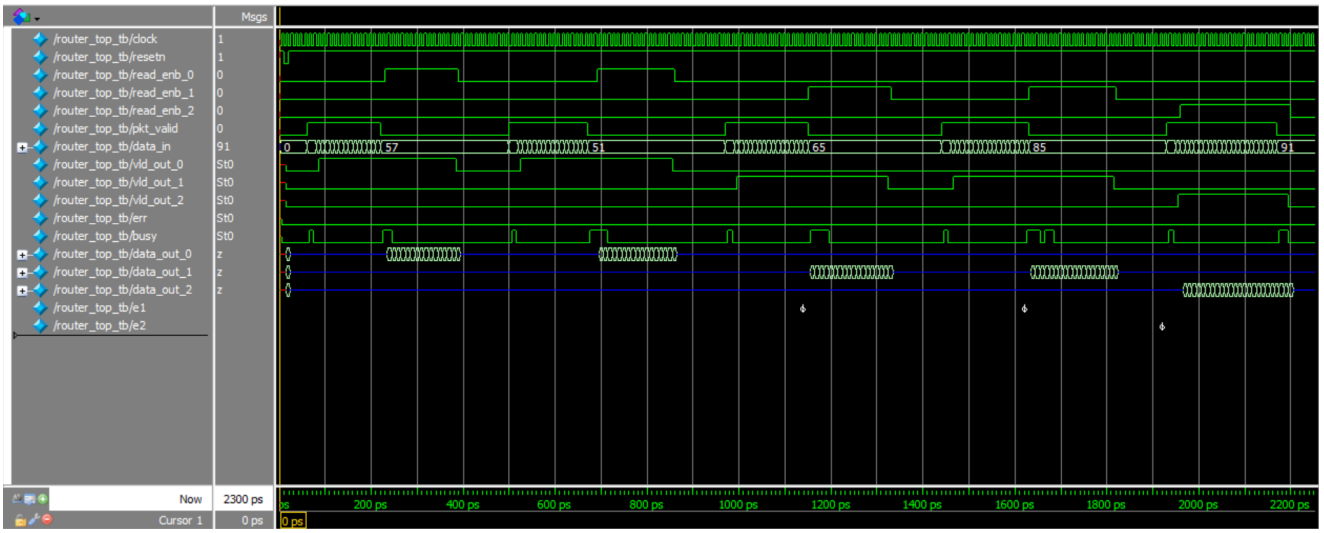
```
        read_enb_0 = 1'b1;
```

```
        wait(!vld_out_0)
        @(negedge clock);
        read_enb_0 = 0;
        #100;

        // Packet Size = 18bytes
        // DA-LFD-LD-FFS-LAF-LP-CPE-DA
        payload_16_17(6'd16,2'd1);
        wait(!vld_out_1)
        @(negedge clock);
        read_enb_1 = 0;
        #100;

        // Packet Size = 19bytes
        // DA-LFD-LD-FFS-LAF-LD-LP-CPE-DA
        payload_16_17(6'd17,2'd1);
        wait(!vld_out_1)
        @(negedge clock);
        read_enb_1 = 0;
        #100;

        // Packet Size = 24bytes
        // DA-LFD-LD-LP-CPE-DA
        random_packet(6'd22,2'd2);
        wait(!vld_out_2)
        @(negedge clock);
        read_enb_2 = 0;
        #100;

        $finish;

      end

    initial
        $monitor("Data Input = %d, data_out_0 = %d, data_out_1 = %d, data_out_2 = %d,
Busy = %b, Error = %b",
        data_in,data_out_0,data_out_1,data_out_2,busy,err);
endmodule
```

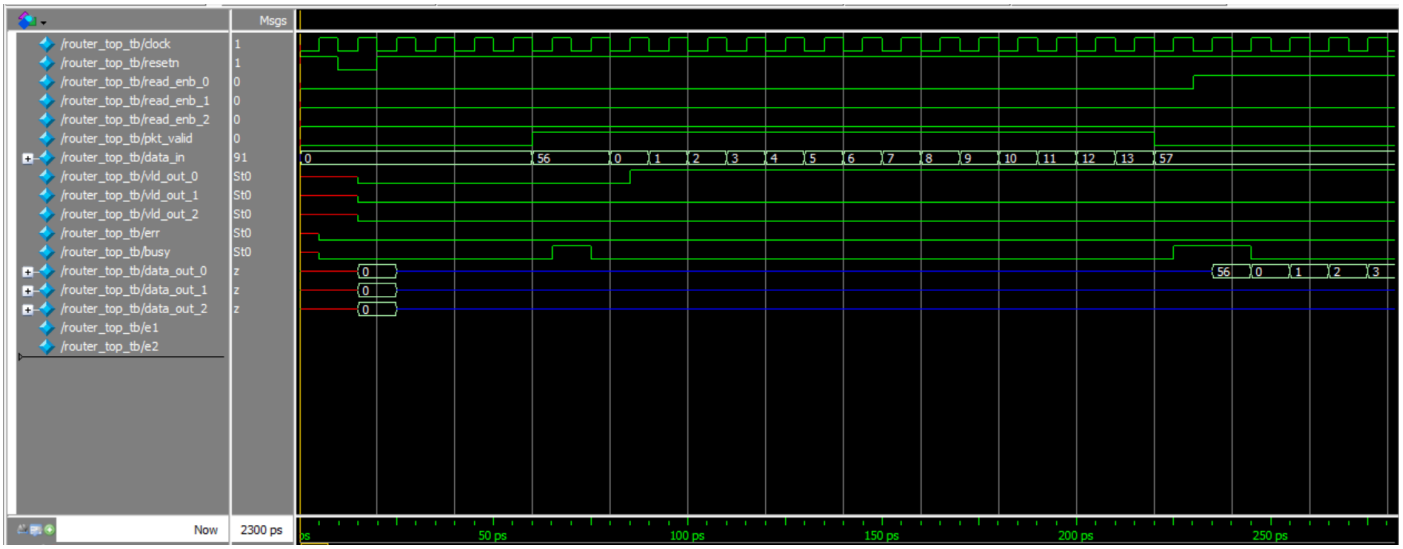
Synthesis Circuit:

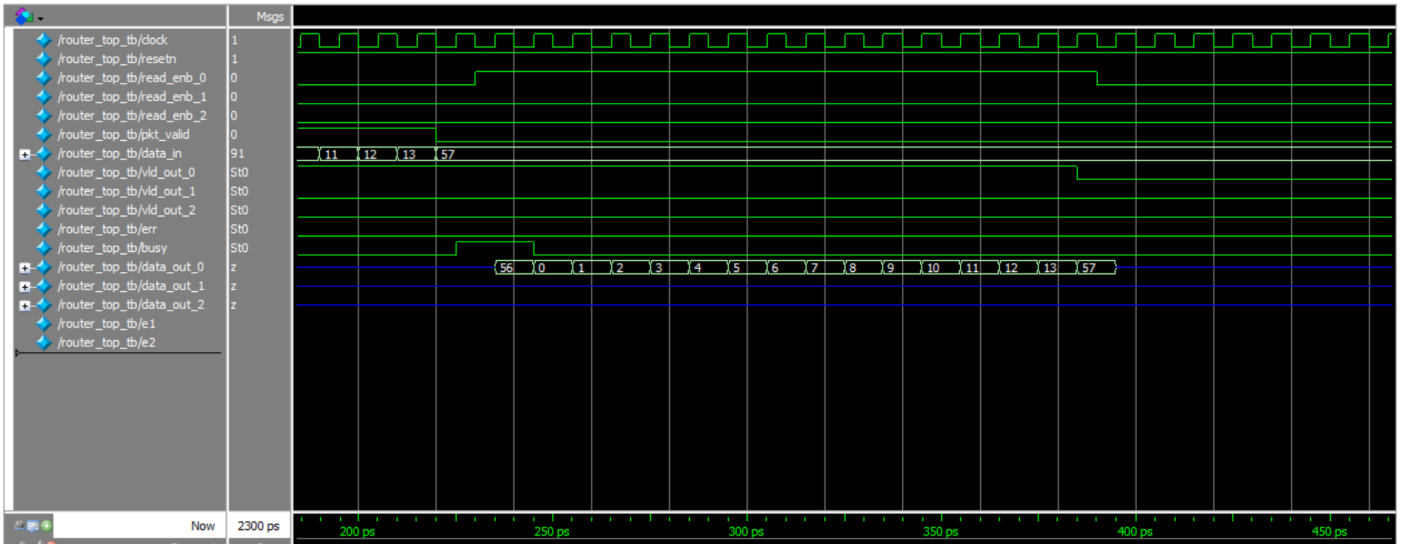


Simulation Waveform:



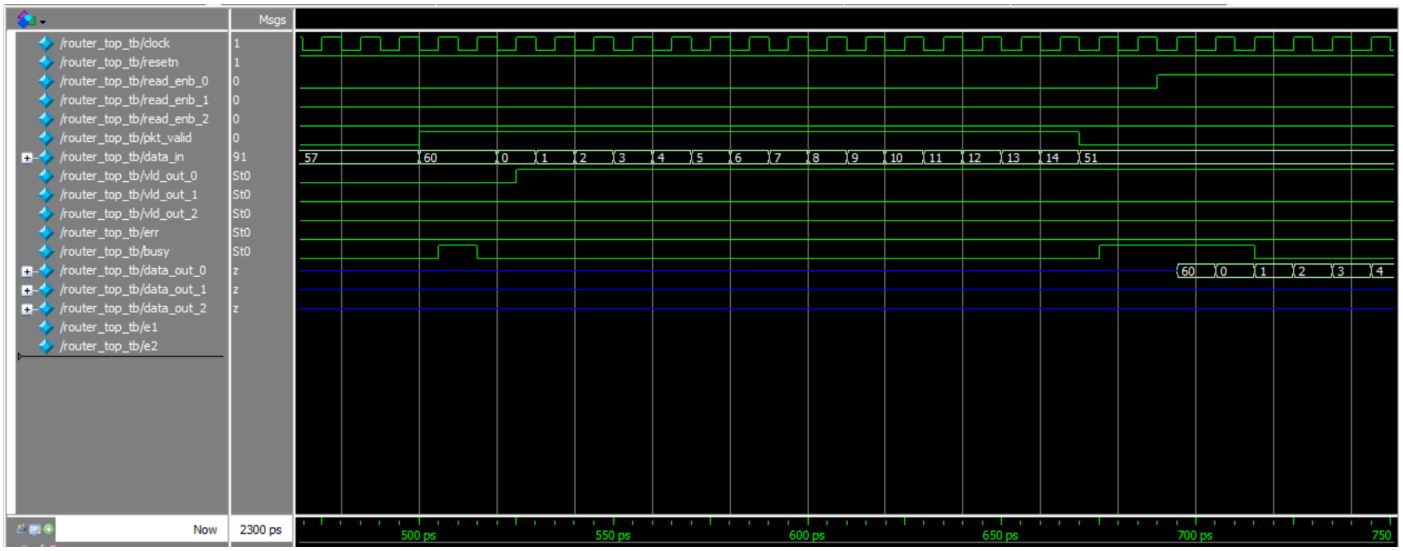
a. Packet Size = 16 bytes

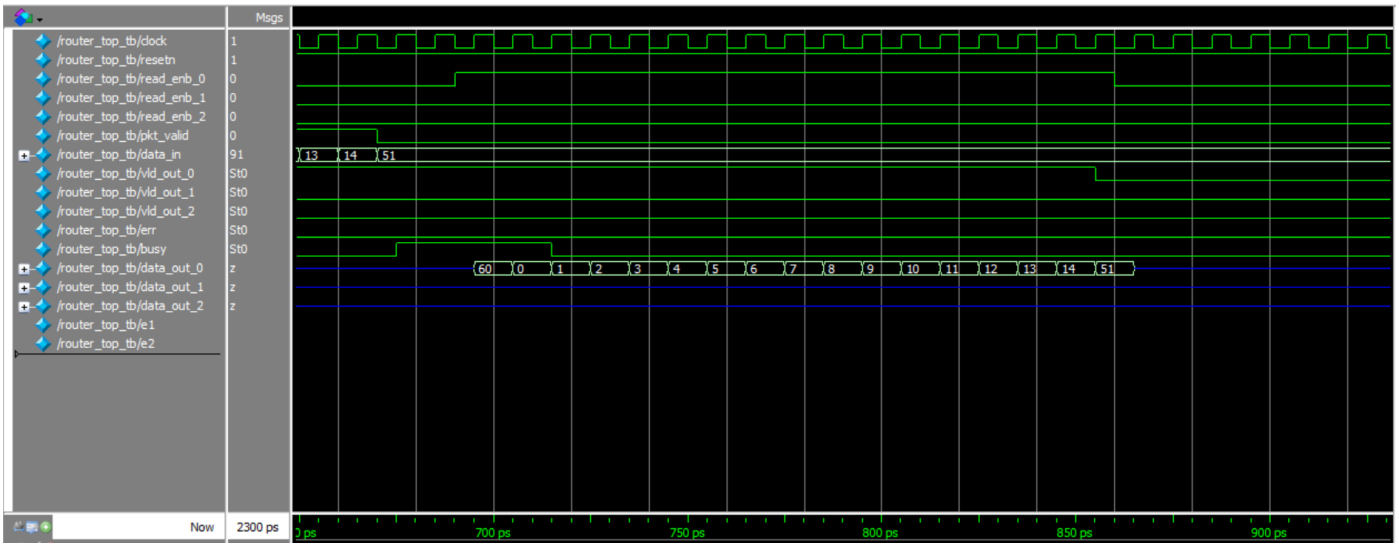





```
Data Input =   0, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  56, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  56, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 1, Error = 0
Data Input =  56, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   0, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   1, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   2, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   3, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   4, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   5, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   6, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   7, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   8, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   9, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  10, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  11, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  12, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  13, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  57, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  57, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 1, Error = 0
Data Input =  57, data_out_0 =  56, data_out_1 =   z, data_out_2 =   z, Busy = 1, Error = 0
Data Input =  57, data_out_0 =   0, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  57, data_out_0 =   1, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  57, data_out_0 =   2, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  57, data_out_0 =   3, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  57, data_out_0 =   4, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  57, data_out_0 =   5, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  57, data_out_0 =   6, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  57, data_out_0 =   7, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  57, data_out_0 =   8, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  57, data_out_0 =   9, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  57, data_out_0 =  10, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  57, data_out_0 =  11, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  57, data_out_0 =  12, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  57, data_out_0 =  13, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  57, data_out_0 =  57, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  57, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
```

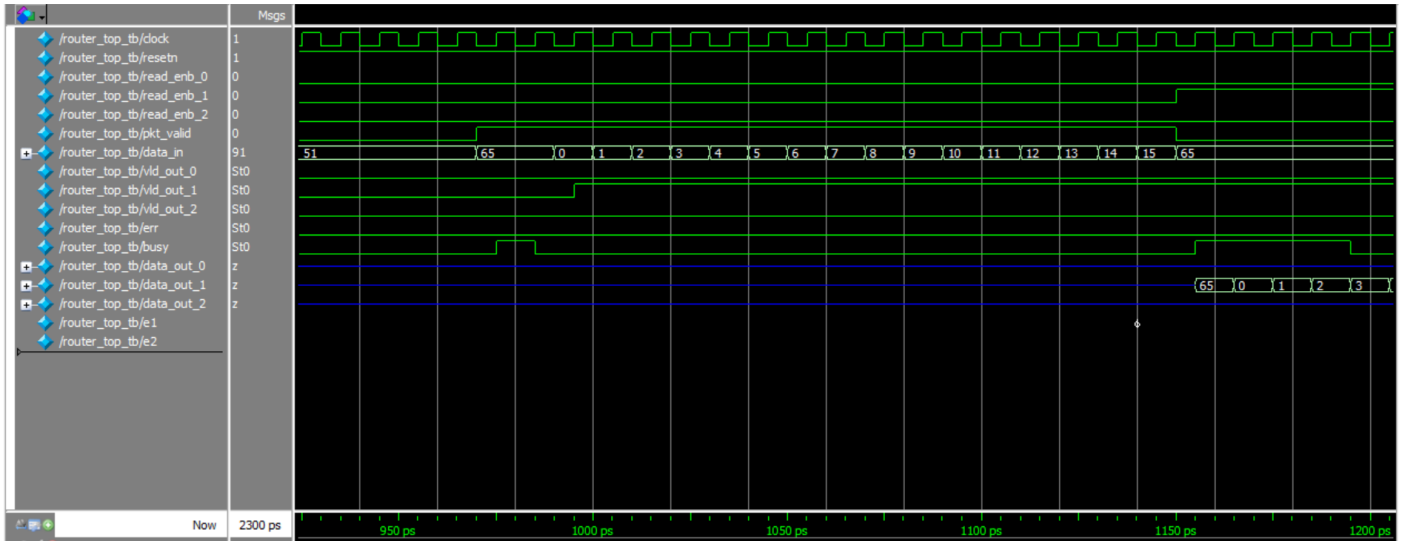
b. Packet Size = 17 bytes

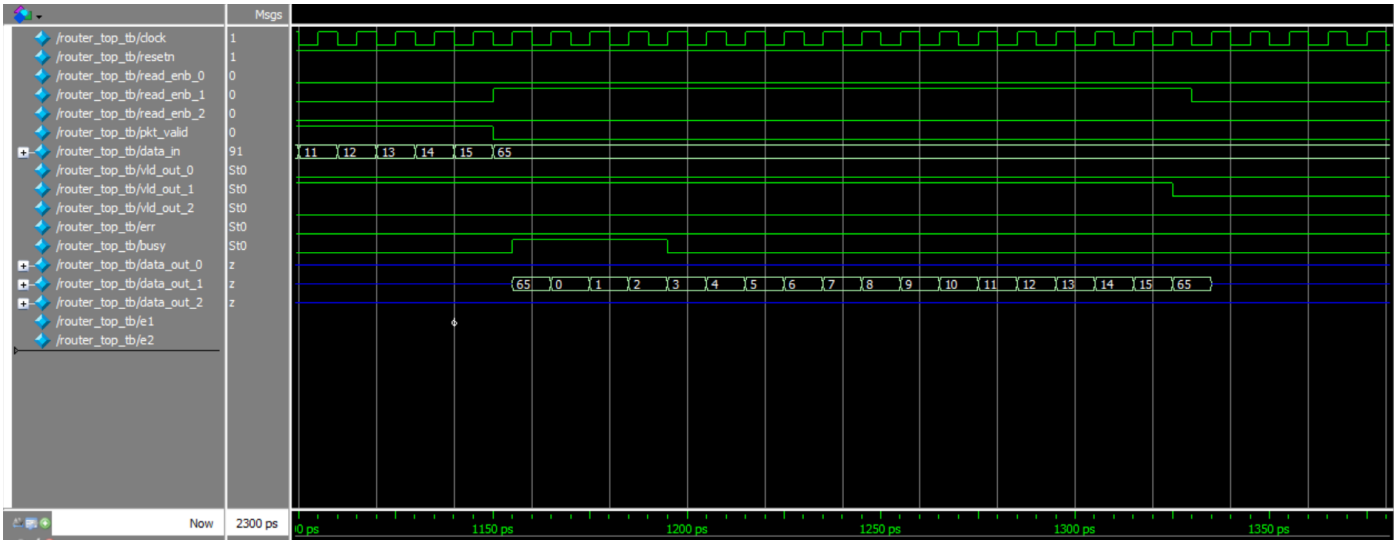





```
Data Input =  60, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   0, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   1, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   2, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   3, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   4, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   5, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   6, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   7, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   8, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   9, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  10, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  11, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  12, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  13, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  14, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  51, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  51, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 1, Error = 0
Data Input =  51, data_out_0 =  60, data_out_1 =   z, data_out_2 =   z, Busy = 1, Error = 0
Data Input =  51, data_out_0 =   0, data_out_1 =   z, data_out_2 =   z, Busy = 1, Error = 0
Data Input =  51, data_out_0 =   1, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  51, data_out_0 =   2, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  51, data_out_0 =   3, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  51, data_out_0 =   4, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  51, data_out_0 =   5, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  51, data_out_0 =   6, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  51, data_out_0 =   7, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  51, data_out_0 =   8, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  51, data_out_0 =   9, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  51, data_out_0 =  10, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  51, data_out_0 =  11, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  51, data_out_0 =  12, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  51, data_out_0 =  13, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  51, data_out_0 =  14, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  51, data_out_0 =  51, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  51, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
```

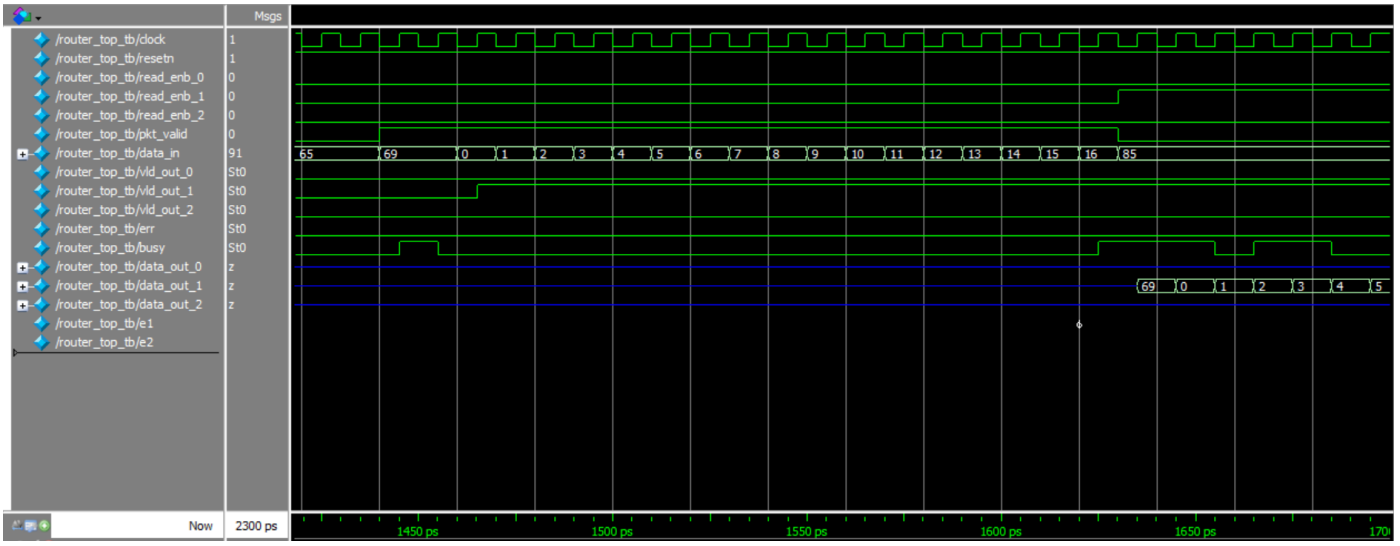
c. Packet Size = 18 bytes

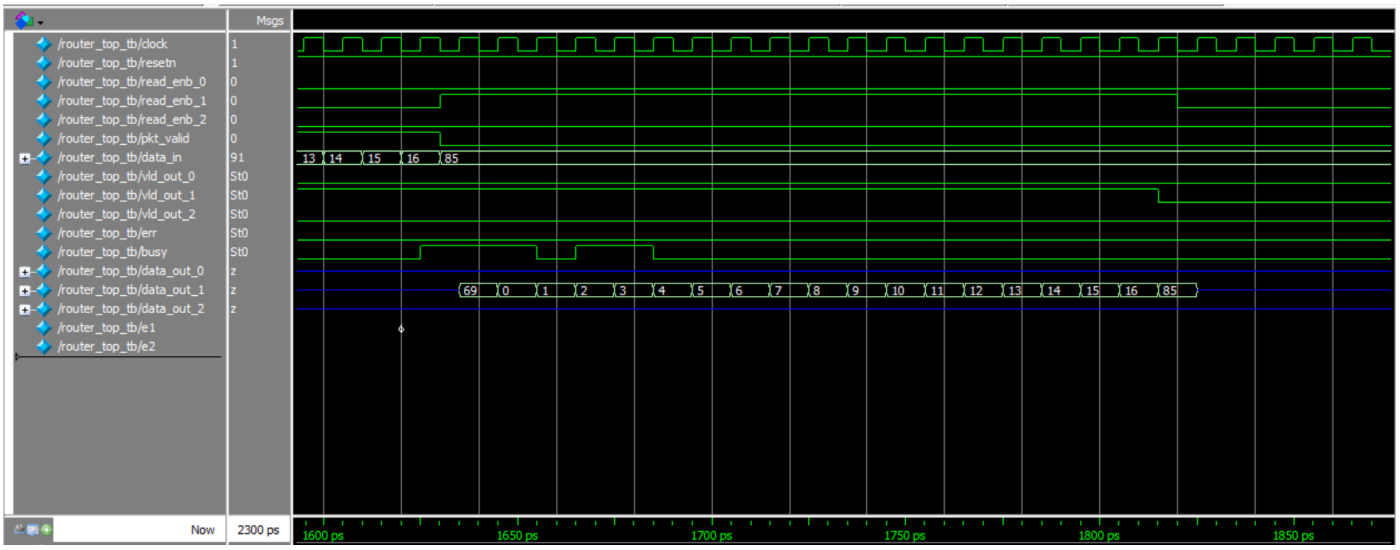





```
Data Input =  65, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   0, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   1, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   2, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   3, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   4, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   5, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   6, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   7, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   8, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   9, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  10, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  11, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  12, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  13, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  14, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  15, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  65, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  65, data_out_0 =   z, data_out_1 =  65, data_out_2 =   z, Busy = 1, Error = 0
Data Input =  65, data_out_0 =   z, data_out_1 =   0, data_out_2 =   z, Busy = 1, Error = 0
Data Input =  65, data_out_0 =   z, data_out_1 =   1, data_out_2 =   z, Busy = 1, Error = 0
Data Input =  65, data_out_0 =   z, data_out_1 =   2, data_out_2 =   z, Busy = 1, Error = 0
Data Input =  65, data_out_0 =   z, data_out_1 =   3, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  65, data_out_0 =   z, data_out_1 =   4, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  65, data_out_0 =   z, data_out_1 =   5, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  65, data_out_0 =   z, data_out_1 =   6, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  65, data_out_0 =   z, data_out_1 =   7, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  65, data_out_0 =   z, data_out_1 =   8, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  65, data_out_0 =   z, data_out_1 =   9, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  65, data_out_0 =   z, data_out_1 =  10, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  65, data_out_0 =   z, data_out_1 =  11, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  65, data_out_0 =   z, data_out_1 =  12, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  65, data_out_0 =   z, data_out_1 =  13, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  65, data_out_0 =   z, data_out_1 =  14, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  65, data_out_0 =   z, data_out_1 =  15, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  65, data_out_0 =   z, data_out_1 =  65, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  65, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
```

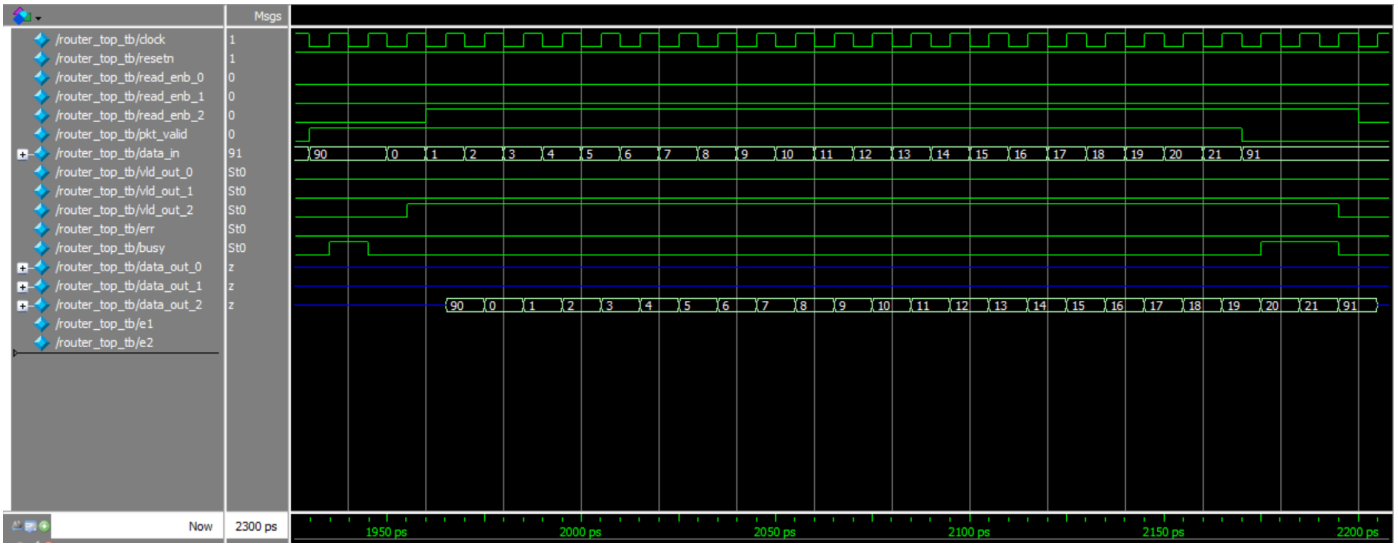
d. Packet Size = 19 bytes






```
Data Input =  69, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   0, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   1, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   2, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   3, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   4, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   5, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   6, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   7, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   8, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   9, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  10, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  11, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  12, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  13, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  14, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  15, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  16, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  16, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 1, Error = 0
Data Input =  85, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 1, Error = 0
Data Input =  85, data_out_0 =   z, data_out_1 =  69, data_out_2 =   z, Busy = 1, Error = 0
Data Input =  85, data_out_0 =   z, data_out_1 =   0, data_out_2 =   z, Busy = 1, Error = 0
Data Input =  85, data_out_0 =   z, data_out_1 =   1, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  85, data_out_0 =   z, data_out_1 =   2, data_out_2 =   z, Busy = 1, Error = 0
Data Input =  85, data_out_0 =   z, data_out_1 =   3, data_out_2 =   z, Busy = 1, Error = 0
Data Input =  85, data_out_0 =   z, data_out_1 =   4, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  85, data_out_0 =   z, data_out_1 =   5, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  85, data_out_0 =   z, data_out_1 =   6, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  85, data_out_0 =   z, data_out_1 =   7, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  85, data_out_0 =   z, data_out_1 =   8, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  85, data_out_0 =   z, data_out_1 =   9, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  85, data_out_0 =   z, data_out_1 =  10, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  85, data_out_0 =   z, data_out_1 =  11, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  85, data_out_0 =   z, data_out_1 =  12, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  85, data_out_0 =   z, data_out_1 =  13, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  85, data_out_0 =   z, data_out_1 =  14, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  85, data_out_0 =   z, data_out_1 =  15, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  85, data_out_0 =   z, data_out_1 =  16, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  85, data_out_0 =   z, data_out_1 =  85, data_out_2 =   z, Busy = 0, Error = 0
Data Input =  85, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
```

e. Packet Size = 24 bytes




```
Data Input =  90, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   0, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   1, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
Data Input =   1, data_out_0 =   z, data_out_1 =   z, data_out_2 =  90, Busy = 0, Error = 0
Data Input =   2, data_out_0 =   z, data_out_1 =   z, data_out_2 =  90, Busy = 0, Error = 0
Data Input =   2, data_out_0 =   z, data_out_1 =   z, data_out_2 =   0, Busy = 0, Error = 0
Data Input =   3, data_out_0 =   z, data_out_1 =   z, data_out_2 =   0, Busy = 0, Error = 0
Data Input =   3, data_out_0 =   z, data_out_1 =   z, data_out_2 =   1, Busy = 0, Error = 0
Data Input =   4, data_out_0 =   z, data_out_1 =   z, data_out_2 =   1, Busy = 0, Error = 0
Data Input =   4, data_out_0 =   z, data_out_1 =   z, data_out_2 =   2, Busy = 0, Error = 0
Data Input =   5, data_out_0 =   z, data_out_1 =   z, data_out_2 =   2, Busy = 0, Error = 0
Data Input =   5, data_out_0 =   z, data_out_1 =   z, data_out_2 =   3, Busy = 0, Error = 0
Data Input =   6, data_out_0 =   z, data_out_1 =   z, data_out_2 =   3, Busy = 0, Error = 0
Data Input =   6, data_out_0 =   z, data_out_1 =   z, data_out_2 =   4, Busy = 0, Error = 0
Data Input =   7, data_out_0 =   z, data_out_1 =   z, data_out_2 =   4, Busy = 0, Error = 0
Data Input =   7, data_out_0 =   z, data_out_1 =   z, data_out_2 =   5, Busy = 0, Error = 0
Data Input =   8, data_out_0 =   z, data_out_1 =   z, data_out_2 =   5, Busy = 0, Error = 0
Data Input =   8, data_out_0 =   z, data_out_1 =   z, data_out_2 =   6, Busy = 0, Error = 0
Data Input =   9, data_out_0 =   z, data_out_1 =   z, data_out_2 =   6, Busy = 0, Error = 0
Data Input =   9, data_out_0 =   z, data_out_1 =   z, data_out_2 =   7, Busy = 0, Error = 0
Data Input =  10, data_out_0 =   z, data_out_1 =   z, data_out_2 =   7, Busy = 0, Error = 0
Data Input =  10, data_out_0 =   z, data_out_1 =   z, data_out_2 =   8, Busy = 0, Error = 0
Data Input =  11, data_out_0 =   z, data_out_1 =   z, data_out_2 =   8, Busy = 0, Error = 0
Data Input =  11, data_out_0 =   z, data_out_1 =   z, data_out_2 =   9, Busy = 0, Error = 0
Data Input =  12, data_out_0 =   z, data_out_1 =   z, data_out_2 =   9, Busy = 0, Error = 0
Data Input =  12, data_out_0 =   z, data_out_1 =   z, data_out_2 =  10, Busy = 0, Error = 0
Data Input =  13, data_out_0 =   z, data_out_1 =   z, data_out_2 =  10, Busy = 0, Error = 0
Data Input =  13, data_out_0 =   z, data_out_1 =   z, data_out_2 =  11, Busy = 0, Error = 0
Data Input =  14, data_out_0 =   z, data_out_1 =   z, data_out_2 =  11, Busy = 0, Error = 0
Data Input =  14, data_out_0 =   z, data_out_1 =   z, data_out_2 =  12, Busy = 0, Error = 0
Data Input =  15, data_out_0 =   z, data_out_1 =   z, data_out_2 =  12, Busy = 0, Error = 0
Data Input =  15, data_out_0 =   z, data_out_1 =   z, data_out_2 =  13, Busy = 0, Error = 0
Data Input =  16, data_out_0 =   z, data_out_1 =   z, data_out_2 =  13, Busy = 0, Error = 0
Data Input =  16, data_out_0 =   z, data_out_1 =   z, data_out_2 =  14, Busy = 0, Error = 0
Data Input =  17, data_out_0 =   z, data_out_1 =   z, data_out_2 =  14, Busy = 0, Error = 0
Data Input =  17, data_out_0 =   z, data_out_1 =   z, data_out_2 =  15, Busy = 0, Error = 0
Data Input =  18, data_out_0 =   z, data_out_1 =   z, data_out_2 =  16, Busy = 0, Error = 0
Data Input =  19, data_out_0 =   z, data_out_1 =   z, data_out_2 =  16, Busy = 0, Error = 0
Data Input =  19, data_out_0 =   z, data_out_1 =   z, data_out_2 =  17, Busy = 0, Error = 0
Data Input =  20, data_out_0 =   z, data_out_1 =   z, data_out_2 =  17, Busy = 0, Error = 0
Data Input =  20, data_out_0 =   z, data_out_1 =   z, data_out_2 =  18, Busy = 0, Error = 0
Data Input =  21, data_out_0 =   z, data_out_1 =   z, data_out_2 =  18, Busy = 0, Error = 0
Data Input =  21, data_out_0 =   z, data_out_1 =   z, data_out_2 =  19, Busy = 0, Error = 0
Data Input =  91, data_out_0 =   z, data_out_1 =   z, data_out_2 =  19, Busy = 0, Error = 0
Data Input =  91, data_out_0 =   z, data_out_1 =   z, data_out_2 =  20, Busy = 1, Error = 0
Data Input =  91, data_out_0 =   z, data_out_1 =   z, data_out_2 =  21, Busy = 1, Error = 0
Data Input =  91, data_out_0 =   z, data_out_1 =   z, data_out_2 =  91, Busy = 0, Error = 0
Data Input =  91, data_out_0 =   z, data_out_1 =   z, data_out_2 =   z, Busy = 0, Error = 0
```

THANKYOU