SOC LAB3 Report

Name: 陳昱碩

ID: 台科大 M11107409

一、Verilog code and block diagram:

```
`timescale 1ns/1ns

module fir
       #(  parameter pADDR_WIDTH = 12,
           parameter pDATA_WIDTH = 32,
           parameter Tape_Num    = 11
        )
       (
           output  wire                     awready,
           output  wire                     wready,
           input   wire                     awvalid,
           input   wire [(pADDR_WIDTH-1):0] awaddr,
           input   wire                     wvalid,
           input   wire [(pDATA_WIDTH-1):0] wdata,
           output  wire                     arready,
           input   wire                     rready,
           input   wire                     arvalid,
           input   wire [(pADDR_WIDTH-1):0] araddr,
           output  wire                     rvalid,
           output  wire [(pDATA_WIDTH-1):0] rdata,
           input   wire                     ss_tvalid,
           input   wire [(pDATA_WIDTH-1):0] ss_tdata,
           input   wire                     ss_tlast,
           output  wire                     ss_tready,
           input   wire                     sm_tready,
           output  wire                     sm_tvalid,
           output  wire [(pDATA_WIDTH-1):0] sm_tdata,
           output  wire                     sm_tlast,

           // bram for tap RAM
           output  wire           [3:0]     tap_WE,
           output  wire                     tap_EN,
           output  wire [(pDATA_WIDTH-1):0] tap_Di,
           output  wire [(pADDR_WIDTH-1):0] tap_A,
           input   wire [(pDATA_WIDTH-1):0] tap_Do,

           // bram for data RAM
           output  wire            [3:0]    data_WE,
           output  wire                     data_EN,
           output  wire [(pDATA_WIDTH-1):0] data_Di,
           output  wire [(pADDR_WIDTH-1):0] data_A,
           input   wire [(pDATA_WIDTH-1):0] data_Do,

           input   wire                     axis_clk,
           input   wire                     axis_rst_n,
           output  wire [1:0]                ap_state
       );
// write your code here!

//state
parameter ap_idle = 0;
parameter ap_start = 1;
parameter ap_done = 2;

reg [1:0] state;
assign ap_state = state;
wire stream_prepared;
reg ap_start_sig;
reg     sm_tlast_r;
reg     ss_finish_r;
wire    ss_finish;
//state
always @( posedge axis_clk ) begin
    if ( !axis_rst_n ) begin
        state <= ap_idle;
    end
    else begin
        case(state)
            ap_idle: begin
                if(stream_prepared  && ap_start_sig)
                    state <= ap_start;

                sm_tlast_r <= 0;
            end

            ap_start: begin
                if(ss_finish && sm_tlast)
                    state <= ap_done;

                if(ss_finish  && !ctrl_tap_valid)
                    sm_tlast_r <= 1;
                else
                    sm_tlast_r <= 0;

            end

            ap_done: begin
                sm_tlast_r <= 0;
            end
            default:
                state <= ap_idle;

        endcase
    end
end


//AXI4_Stream write
reg ss_tready_r;
assign ss_tready = ss_tready_r;


reg [pADDR_WIDTH-1:0] data_WA_r;
reg [pDATA_WIDTH-1:0] data_Di_r;
reg [3:0]             data_WE_r;

assign data_WE = data_WE_r ;
wire [pADDR_WIDTH-1:0] data_WA;
assign data_WA = data_WA_r;
assign data_Di = data_Di_r;

wire [3:0]              ss_count;
reg [3:0]               ss_count_r;
assign ss_count = ss_count_r;

reg stream_prepared_r;

assign stream_prepared = stream_prepared_r;

reg     ss_read_valid_r;
wire    ss_read_valid;
assign  ss_read_valid = ss_read_valid_r;


assign  ss_finish = ss_finish_r;

//AXI4_Stream write
reg data_EN_sw_r,data_EN_sr_r ,data_EN_r_d;
wire    ss_write_valid;

always @( posedge axis_clk ) begin
    if ( !axis_rst_n ) begin
        ss_tready_r <= 0;
        ss_finish_r <= 0;

        data_WA_r <= 0;
        ss_count_r <= 0;
        stream_prepared_r <= 0;
       ss_read_valid_r <= 0;
    end
    else begin
       if (!ss_tready && ss_tvalid) begin
           case(state)
               ap_idle: begin
                   if((ss_count <= Tape_Num - 1) && !stream_prepared) begin
                       data_WE_r <= 4'b1111;
                       data_EN_sw_r <= 1;

                       data_WA_r <= (ss_count == 0) ? 0:data_WA_r + 4;
                       ss_count_r <= ss_count_r + 1;
                       data_Di_r <= 0;

                   end
                   else begin
                       stream_prepared_r <= 1;
                       ss_count_r <= 4'd10;

                       data_EN_sw_r <= 0;
                       data_WE_r <= 0;
                   end

               end

               ap_start: begin
                   if(ss_write_valid) begin
                       ss_tready_r <= 1;
                       data_WE_r <= 4'b1111;
                       data_EN_sw_r <= 1;

                       data_WA_r <= (ss_count == 4'd10) ? 0 :data_WA_r + 4;
                       ss_count_r <=(ss_count == 4'd10) ? 0 :ss_count_r + 1;
                       data_Di_r <= ss_tdata;

                       ss_read_valid_r <= 1;

                   end
                   else if (sm_tvalid)
                       ss_read_valid_r <= 0;
                   else begin
                       data_WE_r <= 0;
                       ss_tready_r <= 0;

                   end
                end
            endcase
        end
        else begin
            data_WE_r <= 4'b0;
            data_EN_sw_r <= 0;
            ss_tready_r <= 0;
            if(ss_tlast)
                ss_finish_r <= 1;


        end

    end

end

//AXI4_Stream read
reg [pADDR_WIDTH-1:0] data_RA_r;
wire [pADDR_WIDTH-1:0] data_RA;
assign data_RA = data_RA_r;



assign data_EN = data_EN_sw_r | data_EN_r_d;

always@(posedge axis_clk) begin
    data_EN_r_d <= data_EN_sr_r;
end


assign data_A =(data_EN_sw_r) ? data_WA :
       (data_EN_sr_r) ? data_RA : 0;

reg sm_tvalid_r;
assign sm_tvalid = sm_tvalid_r;

reg  [pDATA_WIDTH-1:0]  sm_tdata_r;
assign  sm_tdata = sm_tdata_r;


assign  sm_tlast = sm_tlast_r;


assign ss_write_valid = ~ ss_read_valid;



//AXI4_Stream read
```

```
reg ctrl_tap_ready_r, tap_EN_sr_r, tap_EN_r_d;
wire ctrl_tap_ready, ctrl_tap_valid, muxsel, ffen;


reg [pADDR_WIDTH-1:0] tap_RA_lr_r, tap_RA_sr_r;
wire [pADDR_WIDTH-1:0]  ctrl_tap_addr, ctrl_data_addr;

always @( posedge axis_clk ) begin
    if ( !axis_rst_n ) begin
        sm_tvalid_r <= 0;

        ctrl_tap_ready_r <= 0;

        data_EN_sr_r <= 0;
        tap_EN_sr_r <= 0;
    end
    else begin
        if (sm_tready && !sm_tvalid) begin
            case(state)
                ap_idle: begin
                end

                ap_start: begin
                    if(ss_read_valid && ctrl_tap_valid) begin
                        sm_tvalid_r <= 0;
                        data_EN_sr_r <= 1;
                        tap_EN_sr_r <= 1;

                        data_RA_r <= ctrl_data_addr;
                        tap_RA_sr_r <= ctrl_tap_addr;

                        ctrl_tap_ready_r <= 1;

                    end
                    else if (ss_read_valid && !ctrl_tap_valid) begin
                        sm_tvalid_r <= 1;
                        ctrl_tap_ready_r <= 0 ;

                    end
                end
            endcase
        end
        else begin
            sm_tvalid_r <= 0;
        end
    end
end
```

```
//caculate fir


reg ffen_d;

wire  [pDATA_WIDTH-1:0] o_ram_data, o_cof_data;

always@(posedge axis_clk) begin
    ffen_d <= ffen;

    if(!axis_rst_n)
        sm_tdata_r<=0;
    else if(sm_tvalid)
        sm_tdata_r<=0;
    else if(ffen_d)
        sm_tdata_r <= sm_tdata_r +(o_ram_data*o_cof_data);

end

//caculate fir


reg [pDATA_WIDTH-1:0] old_ram_data_r, old_cof_data_r;
wire [pDATA_WIDTH-1:0] old_ram_data, old_cof_data;
wire [pDATA_WIDTH-1:0] new_ram_data, new_cof_data;

assign o_ram_data = muxsel ?  old_ram_data : new_ram_data;
assign o_cof_data = muxsel ?  old_cof_data : new_cof_data;
assign old_ram_data = old_ram_data_r;
assign old_cof_data = old_cof_data_r;

assign new_cof_data = tap_Do;
assign new_ram_data = data_Do;

//caculate fir
always@(posedge axis_clk) begin
    if(ffen) begin
        old_ram_data_r <= new_ram_data;
        old_cof_data_r <= new_cof_data;
    end
end





assign ctrl_tap_ready = ctrl_tap_ready_r;

wire [3:0] ctrl_count;
```

```
//control fir
wire en;
reg o_valid_r, ffen_r;
reg [11:0]o_data_addr_r, o_tap_addr_r;


assign ctrl_tap_valid = o_valid_r;
assign ctrl_data_addr = o_data_addr_r;
assign ctrl_tap_addr = o_tap_addr_r;
assign ffen = ffen_r;
assign muxsel = ~ffen ;
assign en = ctrl_tap_ready & ctrl_tap_valid;
reg[3:0] count_r;
assign ctrl_count = count_r;

reg [11:0]tap_last_addr_r;

always@(posedge axis_clk) begin
    if (!axis_rst_n) begin
        o_data_addr_r <= 0;
        o_tap_addr_r <= 12'd40;
        ffen_r <= 0;
        o_valid_r <= 0;
        count_r <= 0;
    end
    else if(en) begin
        o_valid_r <= (ctrl_count == 4'd11) ? 0 : 1;

        o_data_addr_r <= (ctrl_count == 4'd11)? 0:o_data_addr_r + 4;

        o_tap_addr_r <= (ctrl_count == 4'd11) ? tap_last_addr_r :
                     (ctrl_tap_addr == 12'd40) ? 0 : ctrl_tap_addr + 4;

        tap_last_addr_r <= (ctrl_count == 0 && ctrl_tap_addr == 0) ? 12'd40 :
                        (ctrl_count == 0) ? ctrl_tap_addr - 4 : tap_last_addr_r;

        count_r <= (ctrl_count == 4'd11) ? 0 :ctrl_count + 1;

        ffen_r <= 1;
    end
    else begin
        o_valid_r <= 1;
        ffen_r <= 0;
    end
end
```

```
//AXI4_Lite write
reg awready_r, wready_r;
assign awready = awready_r;
assign wready = wready_r;

reg [pADDR_WIDTH-1:0] tap_WA_r;
reg [pDATA_WIDTH-1:0] tap_Di_r;

assign tap_WE = {4{awready & wready}};
wire [pADDR_WIDTH-1:0] tap_WA;
assign tap_WA = tap_WA_r;
assign tap_Di = tap_Di_r;

reg [pDATA_WIDTH-1:0] data_length;


//AXI4_Lite write
always @( posedge axis_clk ) begin
    if ( !axis_rst_n ) begin
        awready_r <= 0;
        wready_r <= 0;
        ap_start_sig <= 0;
    end
    else begin
        if (!awready && awvalid) begin
            if(awaddr>=12'h20 && awaddr<=12'h60) begin
                awready_r <= 1;
                tap_WA_r <= awaddr-12'h20;
            end

            else
                awready_r <= 0;
        end
        else begin
            awready_r <= 0;
        end

        if (!wready && wvalid) begin
            wready_r <= 1;
            if(awaddr>=12'h20 && awaddr<=12'h60) begin
                tap_Di_r <= wdata;
            end
            else if (awaddr==12'h10)
                data_length <= wdata;
            else if(awaddr==0 && wdata==1)
                ap_start_sig <= 1;

        end
        else begin
```

```
            wready_r <= 0;
            ap_start_sig <= 0;
        end

    end
end

//AXI4_Lite read
reg arready_r, rvalid_r;

assign arready = arready_r;
assign rvalid = rvalid_r;
reg [pDATA_WIDTH-1:0] rdata_r;
assign rdata = rdata_r ;

wire [pADDR_WIDTH-1:0] tap_RA;
assign tap_RA = (tap_EN_sr_r) ? tap_RA_sr_r : tap_RA_lr_r;




assign tap_EN = {awready & wready} | tap_EN_r_d ;

always @( posedge axis_clk ) begin
    tap_EN_r_d <= {arvalid & arready} | tap_EN_sr_r;
end

assign tap_A = ({awready & wready}) ? tap_WA :
       ({arvalid & arready} | tap_EN_sr_r) ? tap_RA : 0;


//AXI4_Lite read

always@(*) begin

    case(state)

        ap_idle: begin
            if(araddr==0 && rvalid)
                rdata_r =32'h04;
            else if(araddr==0 && rvalid && ap_start_sig)
                rdata_r =32'h01;
            else
                rdata_r = tap_Do;


        end

        ap_start: begin
```

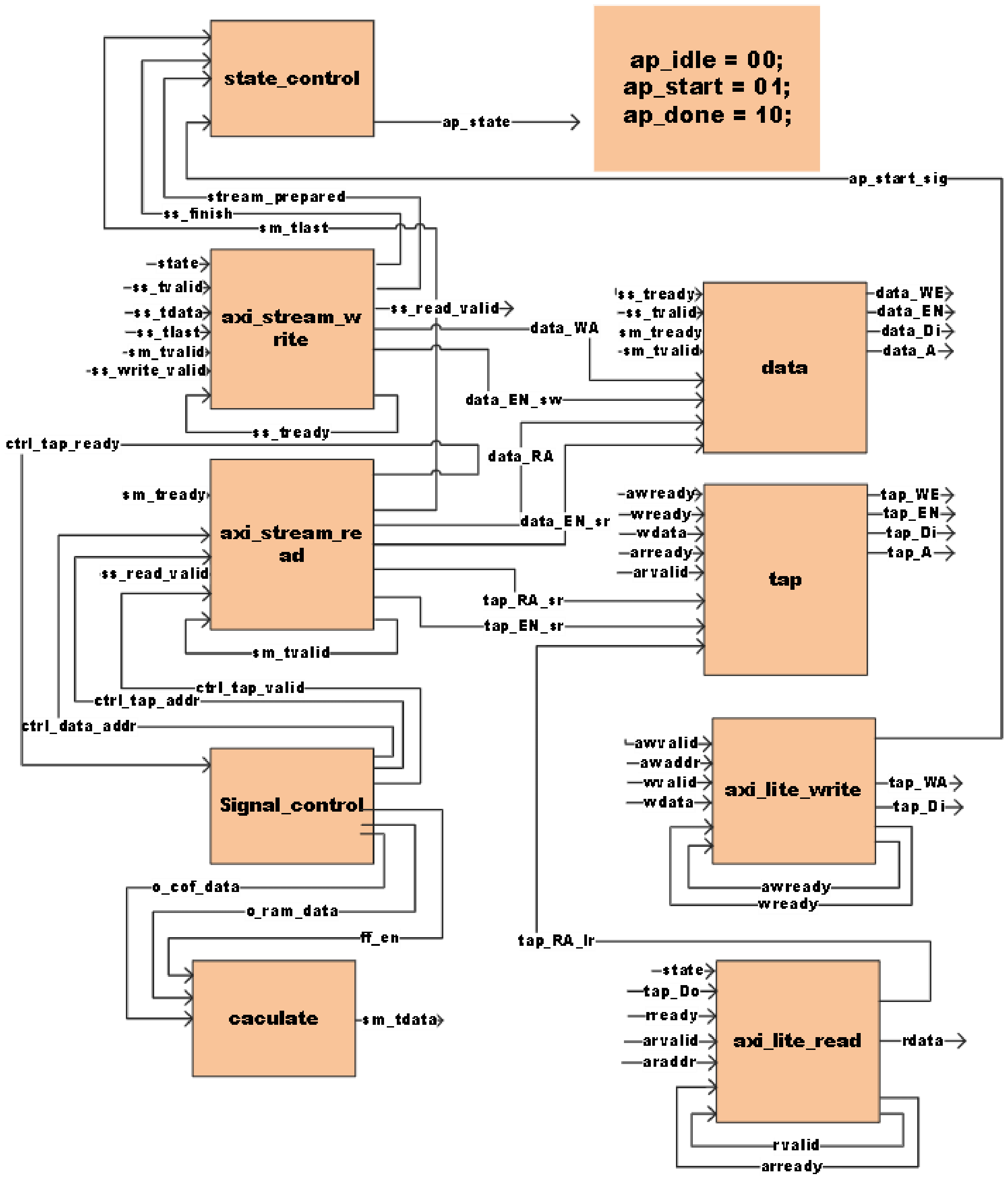
```
            if(araddr==0 && rvalid)
                rdata_r =32'h00;
            else if(awaddr==12'h10 && rvalid)
                rdata_r = data_length;
            else
                rdata_r = tap_Do;

        end

        ap_done: begin
            if(araddr==0 && rvalid)
                rdata_r =32'h06;
            else
                rdata_r = tap_Do;
        end
        default:
            rdata_r = 0;
    endcase
end

//AXI4_Lite read
always @( posedge axis_clk ) begin
    if ( !axis_rst_n ) begin
        arready_r <= 0;
        rvalid_r <= 0;
    end
    else begin
        if(!arready && arvalid && !rvalid) begin
            if(araddr>=12'h20 && araddr<=12'h60) begin
                arready_r <= 1;
                tap_RA_lr_r <= araddr-12'h20;
            end
            else if(araddr==0) begin
                arready_r <= 1;
            end
            else if (awaddr==12'h10)
                arready_r <= 1;
            else
                arready_r <= 0;

        end
        else if(arready && arvalid && !rvalid) begin
            arready_r <= 0;
            rvalid_r <= 1;
        end
        else begin
            arready_r <= 0;
            rvalid_r <= 0;
        end
    end
end
endmodule
```



二、Operations:

How to receive data-in and tap parameters and place into SRAM?

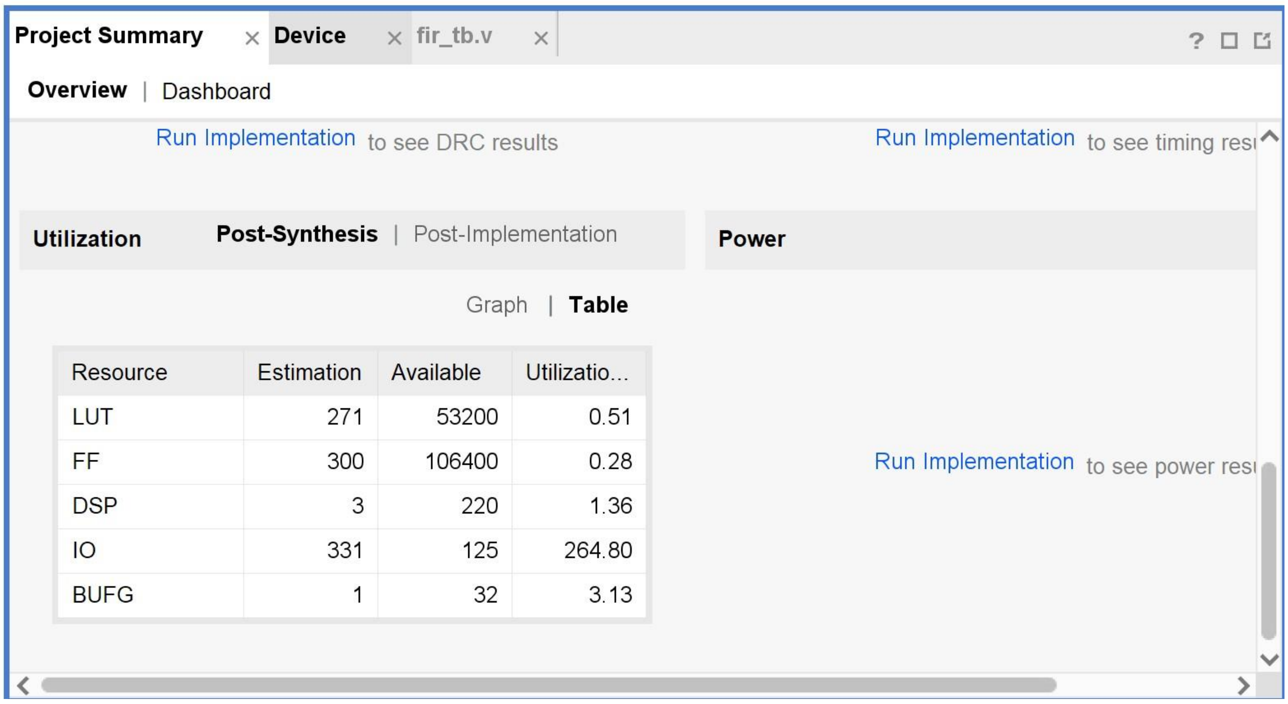
When tap_EN = 1 and tap_WE = 4'hF, wdata will be written into awaddr. Additionally, when data_EN = 1 and data_WE = 4'hF, ss_tdata will be written into the position of data_A. Thus, the memory write operation is completed.

How ap_done is generated?

When ss_tlast = 1 and sm_tlast = 1.

三、Reports:

Synthesis report



Project Summary × Device × fir_tb.v ×

Overview | Dashboard

Run Implementation to see DRC results

Run Implementation to see timing resı

Utilization Post-Synthesis | Post-Implementation

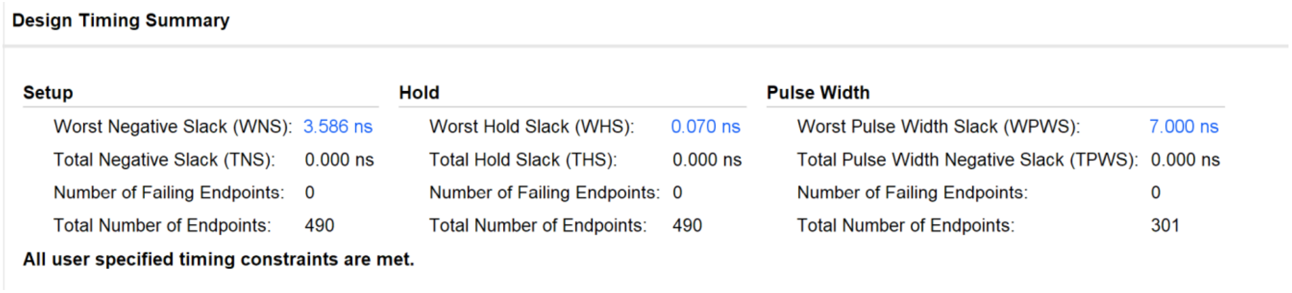
Power

Graph | Table

Resource	Estimation	Available	Utilizatio...
LUT	271	53200	0.51
FF	300	106400	0.28
DSP	3	220	1.36
IO	331	125	264.80
BUFG	1	32	3.13

Run Implementation to see power resı

Timing report



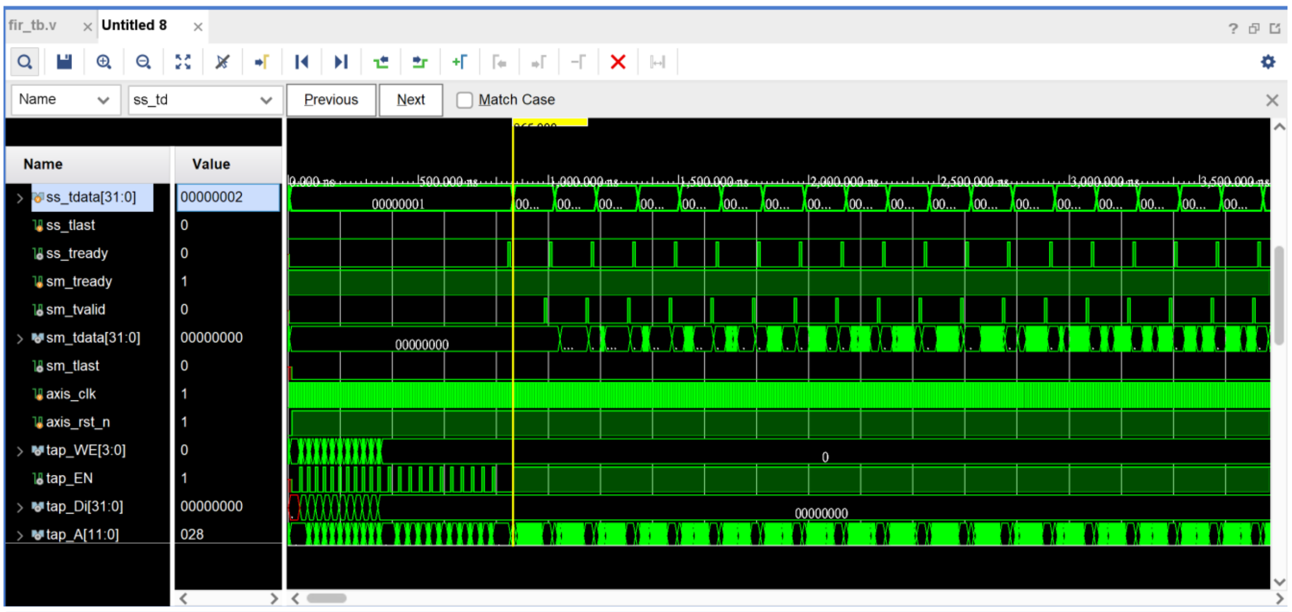
Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	3.586 ns	Worst Hold Slack (WHS):	0.070 ns	Worst Pulse Width Slack (WPWS):	7.000 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	490	Total Number of Endpoints:	490	Total Number of Endpoints:	301

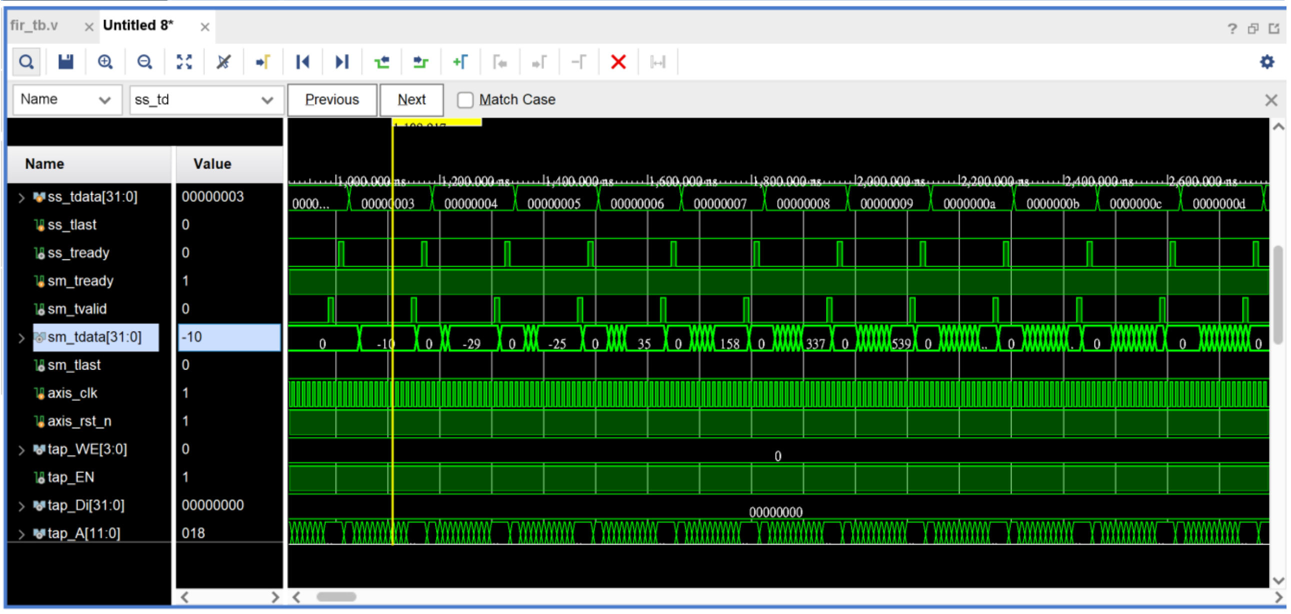
All user specified timing constraints are met.

四、Waveforms:

data stream in and out

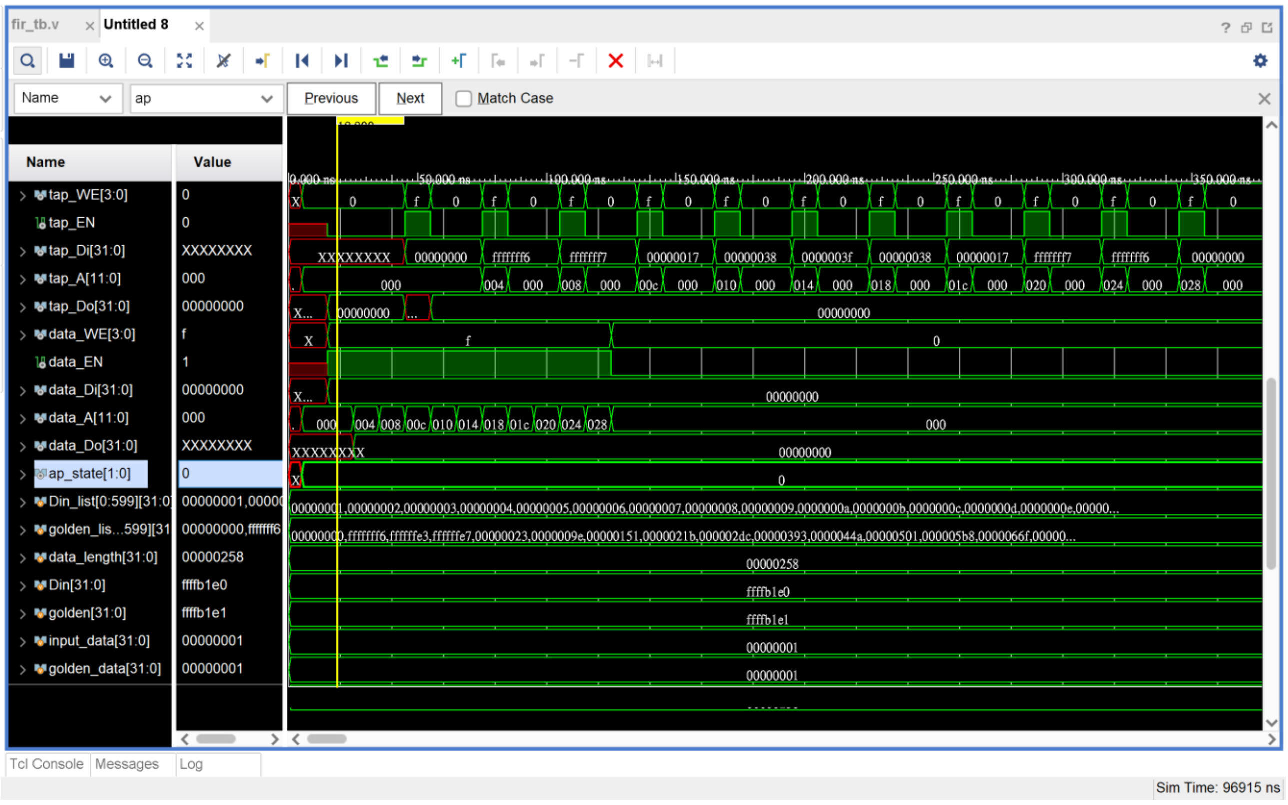


Name	Value
ss_tdata[31:0]	00000002
ss_tlast	0
ss_tready	0
sm_tready	1
sm_tvalid	0
sm_tdata[31:0]	00000000
sm_tlast	0
axis_clk	1
axis_rst_n	1
tap_WE[3:0]	0
tap_EN	1
tap_Di[31:0]	00000000
tap_A[11:0]	028



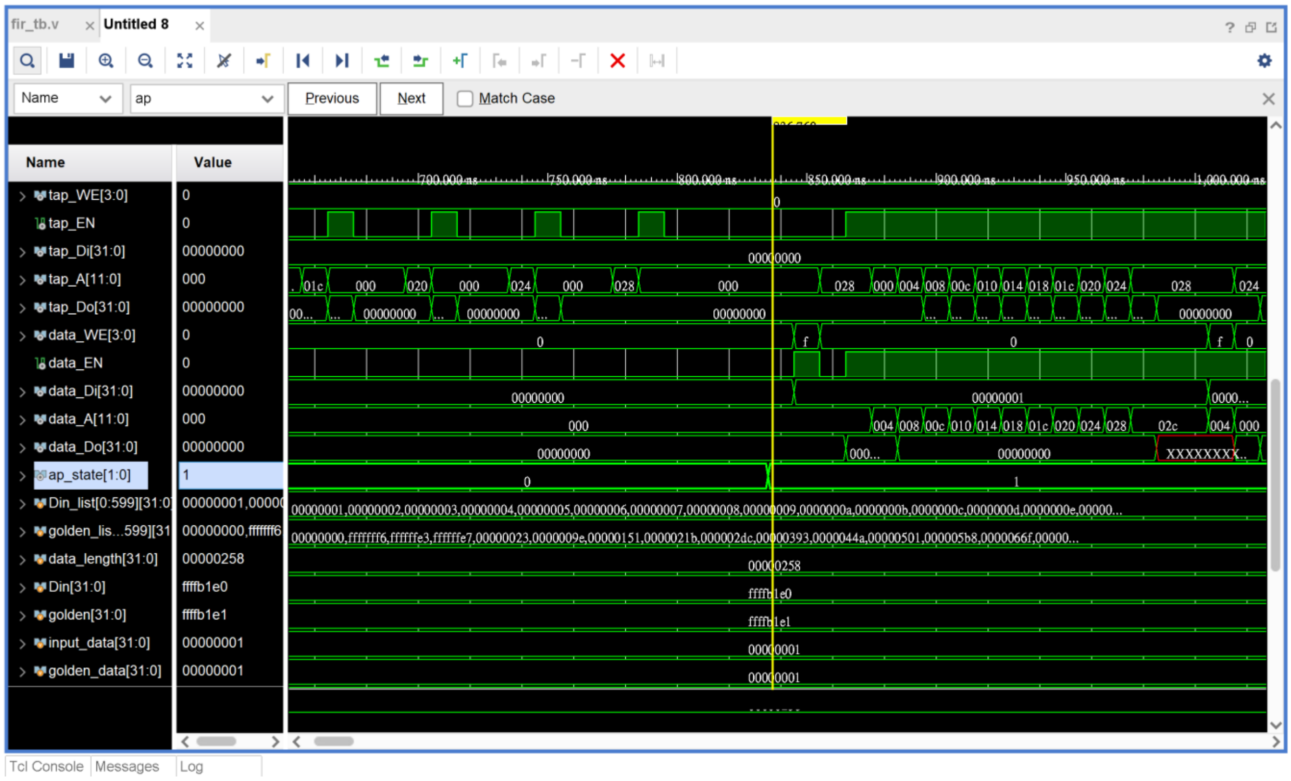
Name	Value
ss_tdata[31:0]	00000003
ss_tlast	0
ss_tready	0
sm_tready	1
sm_tvalid	0
sm_tdata[31:0]	-10
sm_tlast	0
axis_clk	1
axis_rst_n	1
tap_WE[3:0]	0
tap_EN	1
tap_Di[31:0]	00000000
tap_A[11:0]	018

ap_state = 00，AP idle



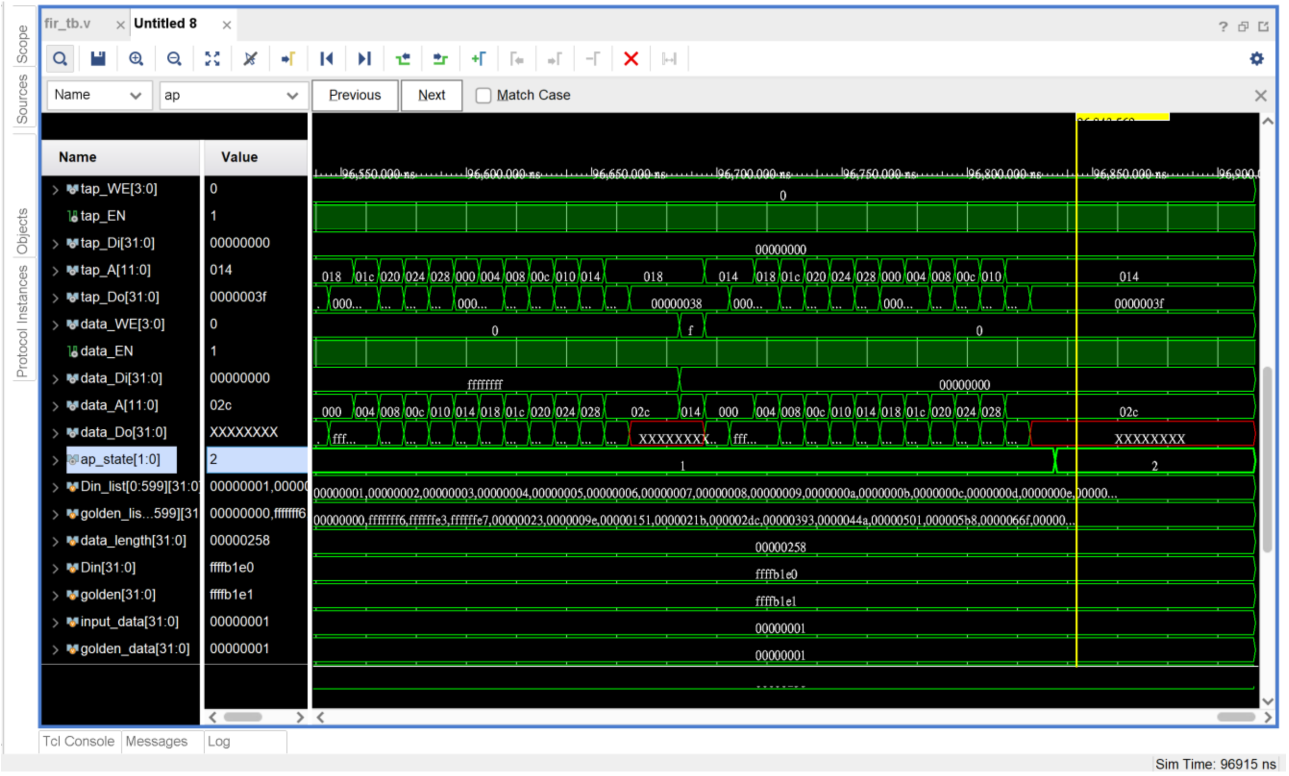
```
Name                    Value
tap_WE[3:0]             0
tap_EN                  0
tap_Di[31:0]            XXXXXXXX
tap_A[11:0]             000
tap_Do[31:0]            00000000
data_WE[3:0]            f
data_EN                 1
data_Di[31:0]           00000000
data_A[11:0]            000
data_Do[31:0]           XXXXXXXX
ap_state[1:0]           0
Din_list[0:599][31:0]   00000001,00000
golden_lis...599][31    00000000,ffffff6
data_length[31:0]       00000258
Din[31:0]               ffffb1e0
golden[31:0]            ffffb1e1
input_data[31:0]        00000001
golden_data[31:0]       00000001
```

ap_state = 01，AP start



```
Name                    Value
tap_WE[3:0]             0
tap_EN                  0
tap_Di[31:0]            00000000
tap_A[11:0]             000
tap_Do[31:0]            00000000
data_WE[3:0]            0
data_EN                 0
data_Di[31:0]           00000000
data_A[11:0]            000
data_Do[31:0]           00000000
ap_state[1:0]           1
Din_list[0:599][31:0]   00000001,00000
golden_lis...599][31    00000000,ffffff6
data_length[31:0]       00000258
Din[31:0]               ffffb1e0
golden[31:0]            ffffb1e1
input_data[31:0]        00000001
golden_data[31:0]       00000001
```

ap_state = 10，AP done

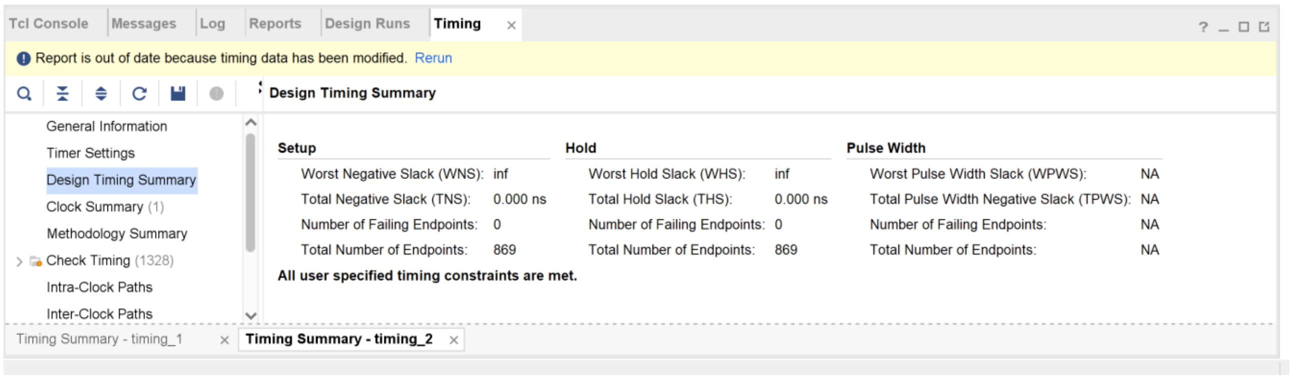


Name	Value
tap_WE[3:0]	0
tap_EN	1
tap_Di[31:0]	00000000
tap_A[11:0]	014
tap_Do[31:0]	0000003f
data_WE[3:0]	0
data_EN	1
data_Di[31:0]	00000000
data_A[11:0]	02c
data_Do[31:0]	XXXXXXXX
ap_state[1:0]	2
Din_list[0:599][31:0]	00000001,00000
golden_lis...599][31	00000000,ffffff6
data_length[31:0]	00000258
Din[31:0]	ffffb1e0
golden[31:0]	ffffb1e1
input_data[31:0]	00000001
golden_data[31:0]	00000001

Sim Time: 96915 ns

五、Problems and solution:

1. Constraint file cannot load



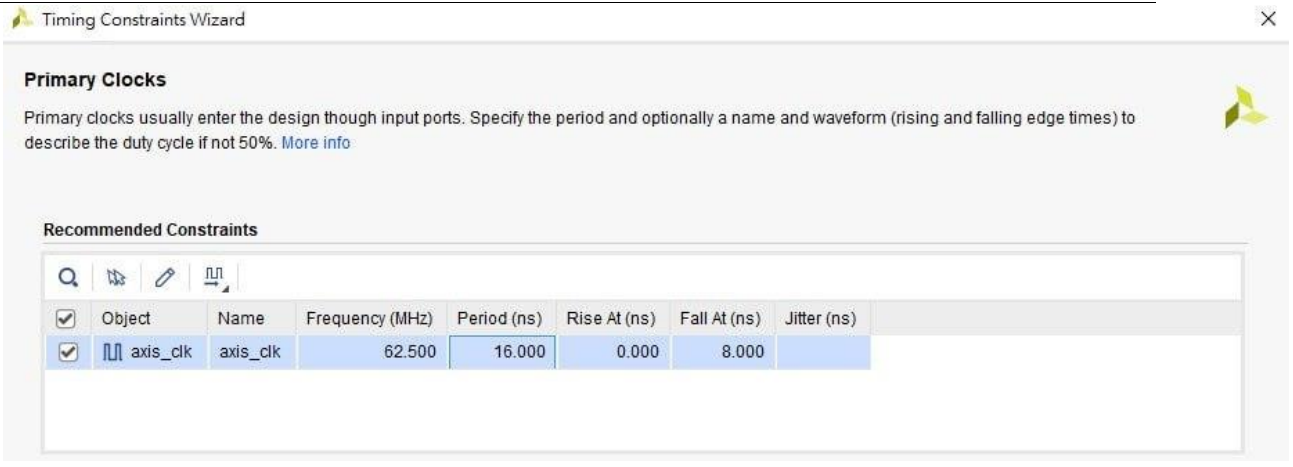
Report is out of date because timing data has been modified. Rerun

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	inf	Worst Hold Slack (WHS):	inf	Worst Pulse Width Slack (WPWS):	NA
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	NA
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	NA
Total Number of Endpoints:	869	Total Number of Endpoints:	869	Total Number of Endpoints:	NA

All user specified timing constraints are met.

如上圖若使用預設給的 Constraint 檔案會造成無法讀取而沒有 timming 資訊，解決方法為使用 Constraint wizard 即可並且數入想要的參數即可使用


Timing Constraints Wizard

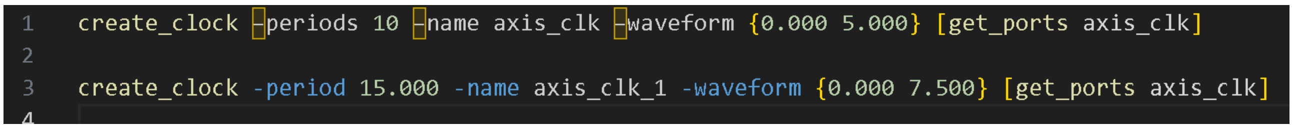
Primary Clocks

Primary clocks usually enter the design though input ports. Specify the period and optionally a name and waveform (rising and falling edge times) to describe the duty cycle if not 50%. More info

Recommended Constraints

✓	Object	Name	Frequency (MHz)	Period (ns)	Rise At (ns)	Fall At (ns)	Jitter (ns)
✓	axis_clk	axis_clk	62.500	16.000	0.000	8.000	

或者更改 periods 成 period 有機會使 Vivado 吃到檔案

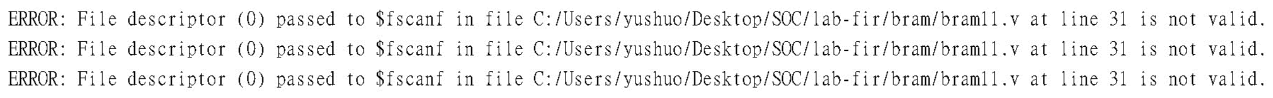


```
create_clock -periods 10 -name axis_clk -waveform {0.000 5.000} [get_ports axis_clk]

create_clock -period 15.000 -name axis_clk_1 -waveform {0.000 7.500} [get_ports axis_clk]
```

2.BRAM ERROR

如下圖如果遇到此問題很有可能是 DAT 檔案沒有吃到，使用絕對路徑即可



```
ERROR: File descriptor (0) passed to $fscanf in file C:/Users/yushuo/Desktop/SOC/lab-fir/bram/bram11.v at line 31 is not valid.
ERROR: File descriptor (0) passed to $fscanf in file C:/Users/yushuo/Desktop/SOC/lab-fir/bram/bram11.v at line 31 is not valid.
ERROR: File descriptor (0) passed to $fscanf in file C:/Users/yushuo/Desktop/SOC/lab-fir/bram/bram11.v at line 31 is not valid.
```