明德扬科技教育有限公司
异步时序之经典 CPU 接口练习
参考代码

官　网：www.mdy-edu.com

淘　宝：mdy-edu.taobao.com

QQ　群：97925396

QQ 咨询：158063679

目录

cpu_if 模块

```
module cpu_if(
        clk             ,
        rst_n           ,
        cpu_data_w      ,
        cpu_data_w_e,
        cpu_data_r      ,
        cpu_addr        ,
        cpu_rd_n        ,
        cpu_we_n        ,
        cpu_cs_n        ,
        cpu_rdy_w       ,
        cpu_rdy_w_e ,
        cfg_mode        ,
        cfg_chan        ,
        sta_fpga
        );

        //参数定义
        parameter       DATA_W =        32;
        parameter       ADDR_W =        16;

        //输入信号定义
        input                   clk             ;
        input                   rst_n           ;
        input [ADDR_W-1:0]      cpu_addr        ;
        input [DATA_W-1:0]      cpu_data_r      ;
        input                   cpu_rd_n        ;
        input                   cpu_we_n        ;
        input                   cpu_cs_n        ;
        input [15:0]            sta_fpga        ;
        output                  cpu_rdy_w       ;
output                          cpu_rdy_w_e      ;
output[ 7:0]                 cfg_mode            ;
output[31:0]                 cfg_chan            ;
output [DATA_W-1:0] cpu_data_w        ;
output                          cpu_data_w_e    ;

reg                               cpu_rdy_w         ;
reg                               cpu_rdy_w_e      ;
reg      [ 7:0]              cfg_mode            ;
reg      [31:0]              cfg_chan            ;
reg      [DATA_W-1:0]    cpu_data_w        ;
reg                               cpu_data_w_e    ;

reg                     s_rd_ff0;
reg                     s_rd_ff1;
reg                     s_rd_ff2;
reg                     s_rd_ff3;
reg                     s_rd_ff4;
reg                     s_we_ff0;
reg                     s_we_ff1;
reg                     s_we_ff2;
reg                     s_we_ff3;
reg                     s_we_ff4;
reg                   s_wr_neg;
reg                   s_rd_neg;
reg[ADDR_W-1:0] s_addr;
reg[DATA_W-1:0] s_data_r;

always@(posedge clk or negedge rst_n)begin
        if(rst_n==1'b0)begin
                cfg_mode <= 0;
        end
        else if(s_wr_neg && s_addr==16'h0001)begin
                cfg_mode <= s_data_r[7:0];
        end
end

always    @(posedge clk or negedge rst_n)begin
        if(rst_n==1'b0)begin
                cfg_chan <= 0;
        end
        else if(s_wr_neg && s_addr==16'h0000) begin
                cfg_chan <= s_data_r[31:0];
        end
```

```
end


always  @(posedge clk or negedge rst_n)begin
    if(rst_n==1'b0)begin
        s_rd_ff0 <= 1'b1;
        s_rd_ff1 <= 1'b1;
        s_rd_ff2 <= 1'b1;
        s_rd_ff3 <= 1'b1;
        s_rd_ff4 <= 1'b1;
        s_we_ff0 <= 1'b1;
        s_we_ff1 <= 1'b1;
        s_we_ff2 <= 1'b1;
        s_we_ff3 <= 1'b1;
        s_we_ff4 <= 1'b1;
    end
    else begin
        s_rd_ff0 <= cpu_rd_n;
        s_rd_ff1 <= s_rd_ff0;
        s_rd_ff2 <= s_rd_ff1;
        s_rd_ff3 <= s_rd_ff2;
        s_rd_ff4 <= s_rd_ff3;
        s_we_ff0 <= cpu_we_n;
        s_we_ff1 <= s_we_ff0;
        s_we_ff2 <= s_we_ff1;
        s_we_ff3 <= s_we_ff2;
        s_we_ff4 <= s_we_ff3;
    end
end


always  @(*)begin
    s_wr_neg = (cpu_cs_n==1'b0 && s_we_ff1==1'b0 && s_we_ff2==1'b1);
end

always  @(*)begin
    s_rd_neg = (cpu_cs_n==1'b0 && s_rd_ff1==1'b0 && s_rd_ff2==1'b1);
end

always  @(*)begin
    s_addr = cpu_addr;
end

always  @(*)begin
```

```
        s_data_r = cpu_data_r;
    end

    always    @(posedge clk or negedge rst_n)begin
        if(rst_n==1'b0)begin
            cpu_data_w <= 0;
        end
        else if(s_rd_neg) begin
            case(s_addr)
                16'h0000:begin
                                cpu_data_w <= cfg_chan;
                          end
                16'h0001:begin
                                cpu_data_w <= {24'h0,cfg_mode};
                          end
                16'h0002:begin
                                cpu_data_w <= {16'h0,sta_fpga};
                          end
                default:begin
                                cpu_data_w <= 0;
                        end
            endcase
        end
    end

    always    @(posedge clk or negedge rst_n)begin
        if(rst_n==1'b0)begin
            cpu_data_w_e <= 1'b0;
        end
        else begin
            cpu_data_w_e <= (~s_rd_ff1 && cpu_cs_n_ff1==1'b0);
        end
    end

    always    @(posedge clk or negedge rst_n)begin
        if(rst_n==1'b0)begin
            cpu_rdy_w <= 1'b1;
        end
        else if(cpu_rdy_w) begin
            if((s_rd_ff3==1'b0 && s_rd_ff4==1'b1) || (s_we_ff3==1'b0 &&
s_we_ff4==1'b1))begin
                cpu_rdy_w <= 1'b0;
            end
        end
        else begin
            if((s_rd_ff1==1'b1 && s_rd_ff2==1'b0) || (s_we_ff1==1'b1 &&
s_we_ff2==1'b0))begin
                cpu_rdy_w <= 1'b1;
            end
        end
    end

    always    @(posedge clk or negedge rst_n)begin
        if(rst_n==1'b0)begin
            cpu_rdy_w_e <= 1'b0;
        end
        else begin
            cpu_rdy_w_e <= ~cpu_cs_n;
        end
    end

endmodule
```

test_cpu_if 模块

```
`timescale 1ns/1ns
module test_cpu_if;

    parameter               DATA_W    = 32;
    parameter               ADDR_W    = 16;
    parameter               CYCLE     = 20;
    parameter               DELAY     = 1 ;

    reg                      clk              ;
    reg                      rst_n            ;
    reg     [ADDR_W-1:0]     cpu_addr         ;
    reg     [DATA_W-1:0]     cpu_data_r       ;
    reg                      cpu_rd_n         ;
    reg                      cpu_we_n         ;
    reg                      cpu_cs_n         ;
    reg     [15:0]           sta_fpga         ;
    wire                     cpu_rdy_w        ;
    wire                     cpu_rdy_w_e      ;
    wire   [ 7:0]            cfg_mode          ;
    wire   [31:0]             cfg_chan          ;
    wire     [DATA_W-1:0] cpu_data_w        ;
    wire                          cpu_data_w_e    ;


cpu_if uut(
    .clk                 (clk                 ),
    .rst_n              (rst_n              ),
    .cpu_data_w     (cpu_data_w     ),
    .cpu_data_w_e(cpu_data_w_e),
    .cpu_data_r     (cpu_data_r     ),
    .cpu_addr         (cpu_addr         ),
    .cpu_rd_n         (cpu_rd_n         ),
    .cpu_we_n         (cpu_we_n         ),
    .cpu_cs_n         (cpu_cs_n         ),
    .cpu_rdy_w       (cpu_rdy_w       ),
    .cpu_rdy_w_e (cpu_rdy_w_e ),
    .cfg_mode         (cfg_mode         ),
    .cfg_chan          (cfg_chan          ),
    .sta_fpga         (sta_fpga         )
    );


 initial begin
     clk = 0;
     forever #(CYCLE/2) clk = ~clk;
 end

 initial begin
     rst_n = 0;
     #(100*CYCLE) rst_n = 1;
 end
 reg[7:0]      tcas ;
 reg[7:0]      tcrw ;
 reg[7:0]      tcah ;
 reg[7:0]      tcwh ;
 integer        i      ;

 initial begin
     sta_fpga = $random;
 end

 initial begin
```

```
tcas = ($random)%20+10;
tcrw = ($random)%20+10;
tcah = ($random)%20+10;
tcwh = ($random)%20+200;
#DELAY;
tcas = (tcas)%20+10;
tcrw = (tcrw)%20+10;
tcah = (tcah)%20+10;
tcwh = (tcwh)%20+200;
cpu_data_r = 0;
cpu_addr      = 0;
cpu_rd_n      = 1;
cpu_we_n      = 1;
cpu_cs_n      = 1;
#(110*CYCLE);
for(i=0;i<3;i=i+1)begin
      cpu_data_r = i+3;
      cpu_addr      = i;
      cpu_rd_n      = 1;
      cpu_we_n      = 1;
      cpu_cs_n      = 0;
      #(tcas*DELAY);
      cpu_rd_n      = 1;
      cpu_we_n      = 0;
      cpu_cs_n      = 0;
      while(cpu_rdy_w)begin
            #DELAY;
      end
      #(tcrw*DELAY);
      cpu_rd_n      = 1;
      cpu_we_n      = 1;
      cpu_cs_n      = 0;
      while(cpu_rdy_w==0)begin
            #DELAY;
      end
      cpu_data_r = 0;
      cpu_addr      = 0;
      cpu_rd_n      = 1;
      cpu_we_n      = 1;
      cpu_cs_n      = 1;
      //#((tcwh-tcah)*DELAY);
      #((200-tcah)*DELAY);
end
```

```
        #(110*CYCLE);
        for(i=0;i<4;i=i+1)begin
                cpu_data_r = 0;
                cpu_addr     = i;
                cpu_rd_n      = 1;
                cpu_we_n      = 1;
                cpu_cs_n      = 0;
                #(tcas*DELAY);
                cpu_rd_n      = 0;
                cpu_we_n      = 1;
                cpu_cs_n      = 0;
                while(cpu_rdy_w)begin
                        #DELAY;
                end
                #(tcrw*DELAY);
                cpu_rd_n      = 1;
                cpu_we_n      = 1;
                cpu_cs_n      = 0;
                while(cpu_rdy_w==0)begin
                        #DELAY;
                end
                cpu_data_r = 0;
                cpu_addr     = 0;
                cpu_rd_n      = 1;
                cpu_we_n      = 1;
                cpu_cs_n      = 1;
                //#((tcwh-tcah)*DELAY);
                #((tcwh-tcah)*DELAY);
        end
    end

endmodule
```