明德扬
科技.教育

异步时序之经典CPU接口传输练习说明思路

点透学习误区　拨出设计精髓

主 讲：潘文明

明德扬科教

明德扬
科技·教育

QQ群： 97925396

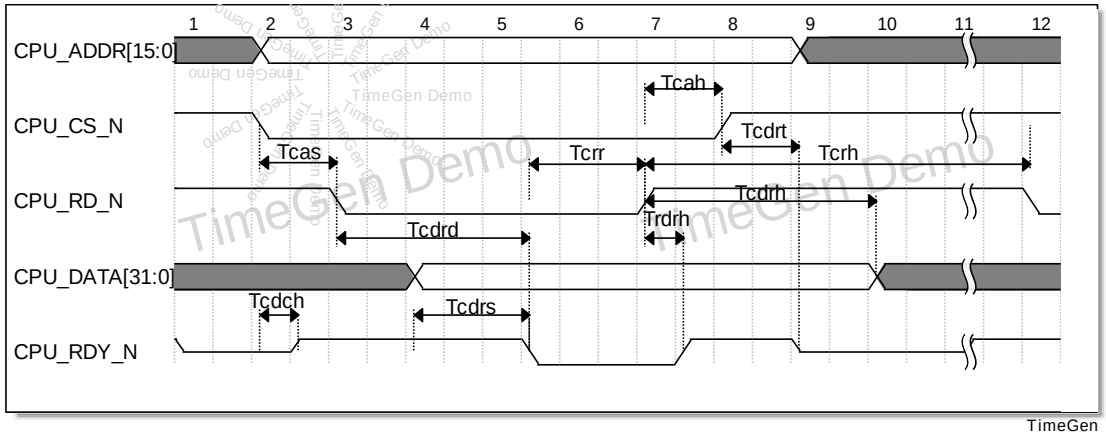
官　网：http://www.mdy-edu.com

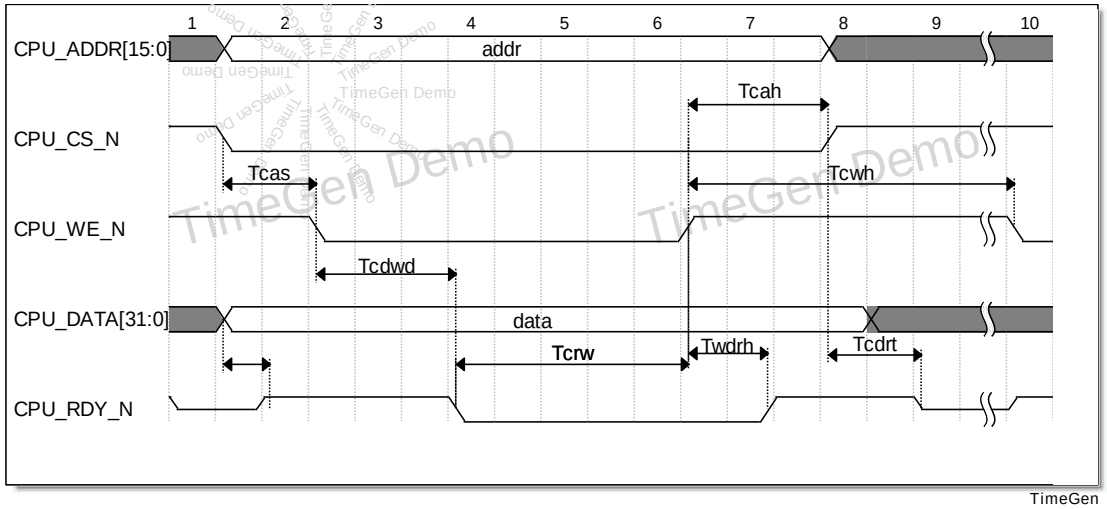
淘　宝：http://mdy-edu.taobao.com

课程大纲

1. 功能要求
2. 设计思路
3. 代码设计

一、功能要求


1
2
3
4
5
6
7
8
9
10
11
12
CPU_ADDR[15:0]
CPU_CS_N
CPU_RD_N
CPU_DATA[31:0]
CPU_RDY_N
Tcah
Tcdrt
Tcas
Tcrr
Tcrh
Tcdrh
Trdrh
Tcdrd
Tcdch
Tcdrs
TimeGen Demo
TimeGen


1
2
3
4
5
6
7
8
9
10
CPU_ADDR[15:0]
addr
CPU_CS_N
CPU_WE_N
CPU_DATA[31:0]
data
CPU_RDY_N
Tcah
Tcas
Tcwh
Tcdwd
Tcrw
Twdrh
Tcdrt
TimeGen Demo
TimeGen

一、功能要求

本练习设计 cpu_if 模块。该模块具有如下功能

1. CPU 通过读时序，可以读到 FPGA 内部的寄存器
   a 读地址为 0 时，将 cfg_chan 的值返回给 CPU。
   b 读地址为 1 时，将 cfg_mode 的值返回给 CPU 低 8 位，其他位补 0。
   c 读地址为 2 时，将 sta_fpga 的值返回给 CPU 低 16 位，高位补 0。
   d 读地址为其他时，返回给 CPU 的值为 0。
2. CPU 通过写时序，可以配置 FPGA 内部的寄存器
   a 写地址为 0 时，将 CPU 总线上的值赋给 cfg_chan。
   b 写地址为 1 时，将 CPU 总线上的低 8 位赋给 cfg_mode。
   c 写地址为其他时，无任何变化。

一、功能要求

二、 寄存器列表

RW：可读可写

RO： 只读

WO：只写

RC： 读清

Bits	Field	ADDR	Access Type	Description	Reset Val
31：0	cfg_chan	16'h0000	RW	FPGA 的工作通道	0

Bits	Field	ADDR	Access Type	Description	Reset Val
31:8	RESERVED	16'h0001	RO	保留	0
7：0	cfg_mode	16'h0001	RW	FPGA 的工作模式	0

Bits	Field	ADDR	Access Type	Description	Reset Val
31:16	RESERVED	16'h0002	RO	保留	0
15：0	sta_fpga	16'h0002	RO	FPGA 的工作状态	0

一、功能要求

信号名	I/O	位宽	说明
clk	I	1	工作时钟，50MHz
rst_n	I	1	复位信号
cpu_data_w	O	32	写数据总线，FPGA 往 CPU 送的数据。连接到三态门上
cpu_data_w_e	O	1	写数据使能，连接到三态门上。该值为 1 时，表示打开三态门，将 cpu_data_w 的值送到总线上。
cpu_data_r	I	32	从数据总线三态门获取的数据
cpu_addr	I	16	地址总线
cpu_rd_n	I	1	读指示
cpu_wr_n	I	1	写指示
cpu_cs_n	I	1	片选指示
cpu_rdy_w	O	1	状态指示数据信号，连接到三态门上
cpu_rdy_w_e	O	1	写状态指示使能信号，连接到三态门上。该值为 1 时，表示打开三态门，将 cpu_rdy_w 的值送到总线上。
cfg_mode	O	8	发给其他模块使用的工作模式指示信号。
cfg_chan	O	32	发给其他模块使用的工作通道指示信号
sta_fpga	I	16	从其他模块过来的工作状态指示信号

三、设计思路—cfg_mode

1. CPU通过写时序，可以配置FPGA内部的寄存器
2. 写地址为1时，将CPU总线上的值赋给cfg_mode。

1. 假设s_wr_neg=1表示写有效
2. 假设s_addr表示写地址
3. 假设s_data_r表示写数据（三态门读到的数据）

```
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
```

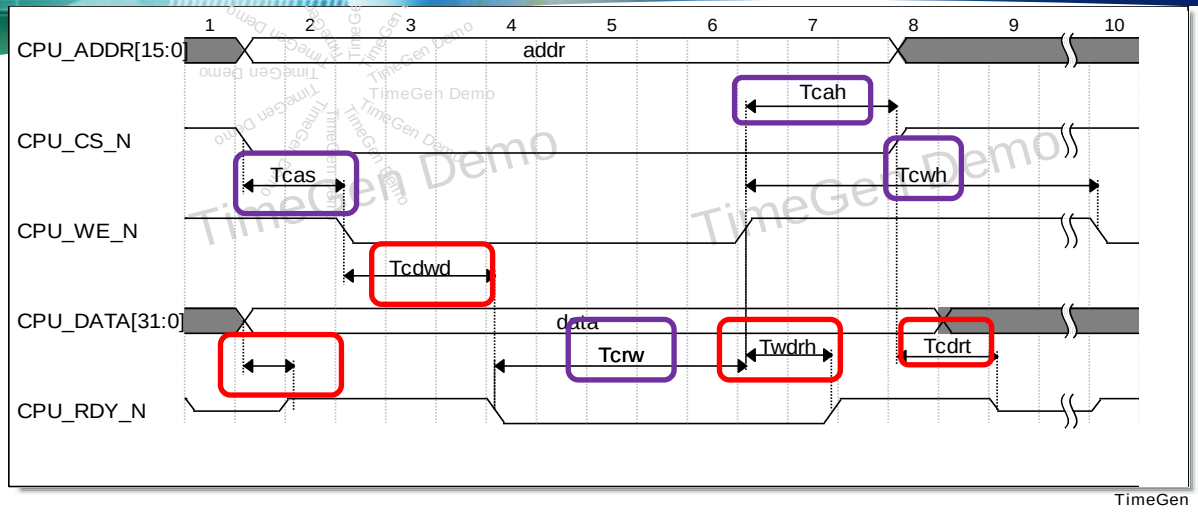
三、设计思路—cfg_chan

1. CPU通过写时序，可以配置FPGA内部的寄存器
2. 写地址为0时，将CPU总线上的值赋给cfg_chan。

1. s_wr_neg=1表示写有效
2. s_addr表示写地址
3. s_data_r表示写数据

```
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
```

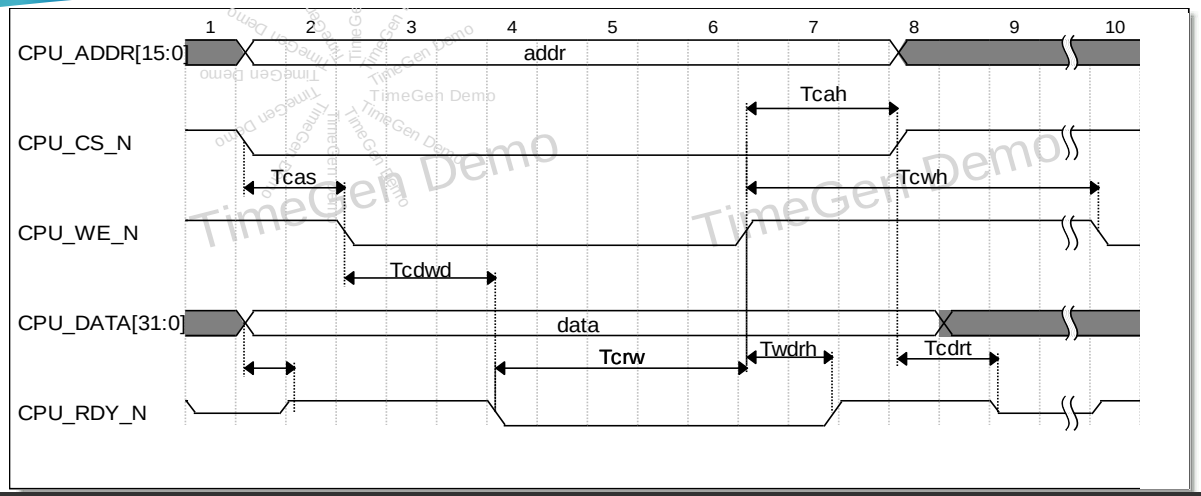
二、写时序



1. Tcas：cs=0到we=0至少有10ns
2. Tcrw：rdy=0到we=1至少有10ns
3. Tcah：we=1到cs=1至少有10ns
4. Tcwh：两次写之间至少有200ns

1. Tcdch：cs=0到rdy=1要在0~60ns范围
2. Tcdwd：we=0到rdy=0在80~200ns
3. Tcdrs：数据在总线至rdy=0必须大于10ns
4. Twdrh：we=1到rdy=1在10~80ns范围
5. Tcdrt：cs=1到rdy=z在0~100ns范围

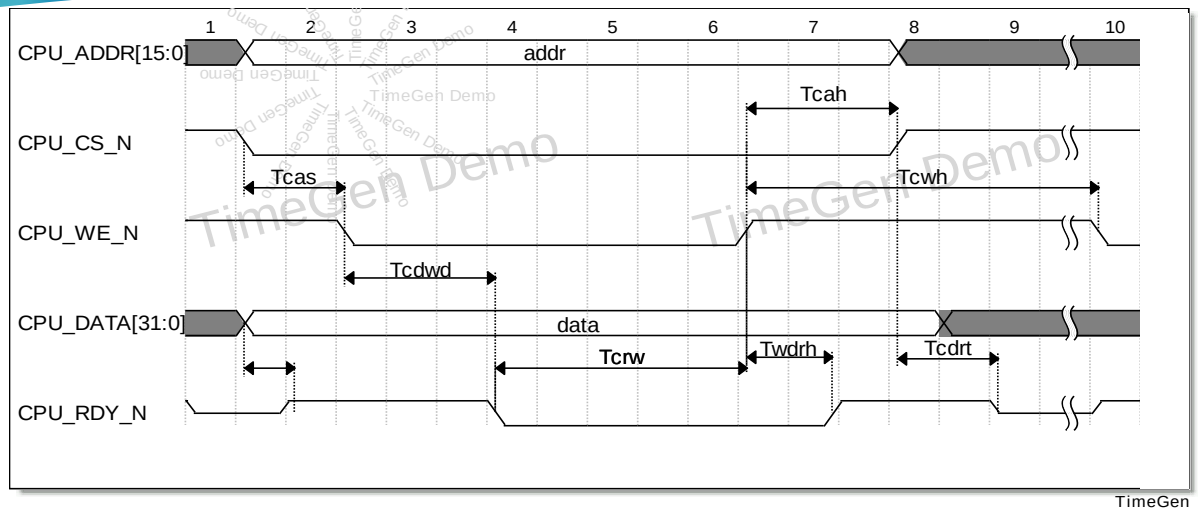
二、写时序cpu_rdy_w




```
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
```

1. 由cpu_we_n触发
2. 由于cpu_we_n是异步信号，因此需要打两拍，we_ff1
3. 由于cpu_we_n=0时间大于1个时钟，因此检测下降沿
4. Tcdwd值为80~200ns（we_n在时钟沿后一点点拉低）：等待时钟上升沿（20ns）+出现延时两拍（40ns）=60ns
5. Tcdwd值为80~200ns（ we_n在时钟沿前一点点拉低）：出现延时两拍（40ns）=40ns
6. 因此采到we_ff1的下降沿后，需要再延时两拍（共4拍），we_ff3的下降沿再拉低，使得周期80ns~200ns
7. 由于Twdrh为0~80，用we_ff1的上升沿都满足要求
8. 读和写的时序参数是一样的

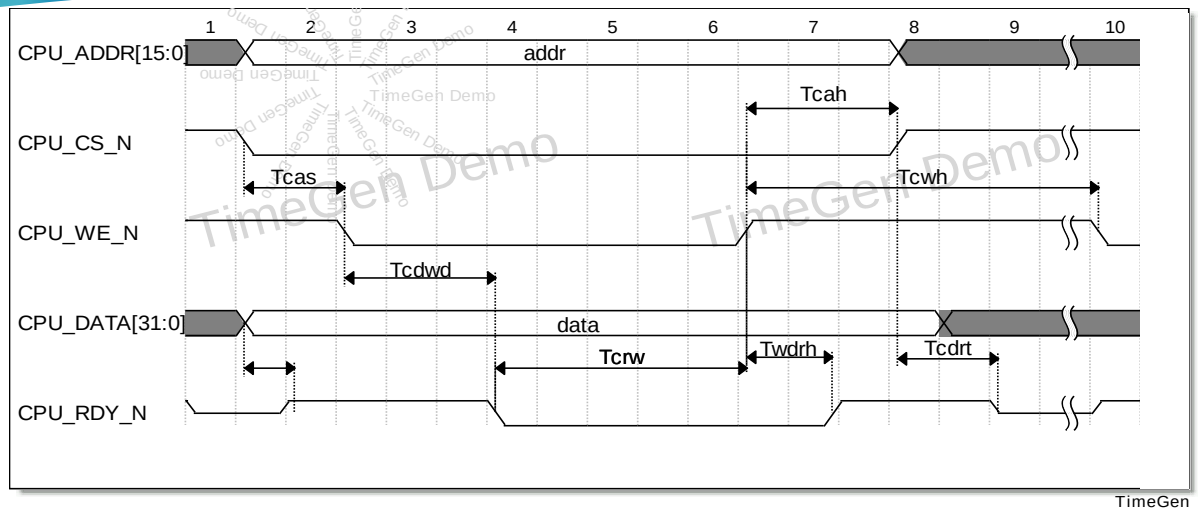
二、写时序cpu_rdy_w_e



1. 由cpu_cs_n触发；
2. 由于cpu_cs_n是异步信号，因此需要打两拍，cs_ff1
3. 由于Tcdch为0~60，Tcdrt为0~100，如果以cs_ff1为条件判断，则延时可能会60ns，不符合Tcdch
4. 其实由于cpu_rdy_w_e连接到三态门后输出，因此不存在亚稳态问题
5. 读和写的参数是一样的
6. 因此时序逻辑 cpu_rdy_w_e <= ~cpu_cs_n ;

```
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
```

二、写时序s_wr_neg

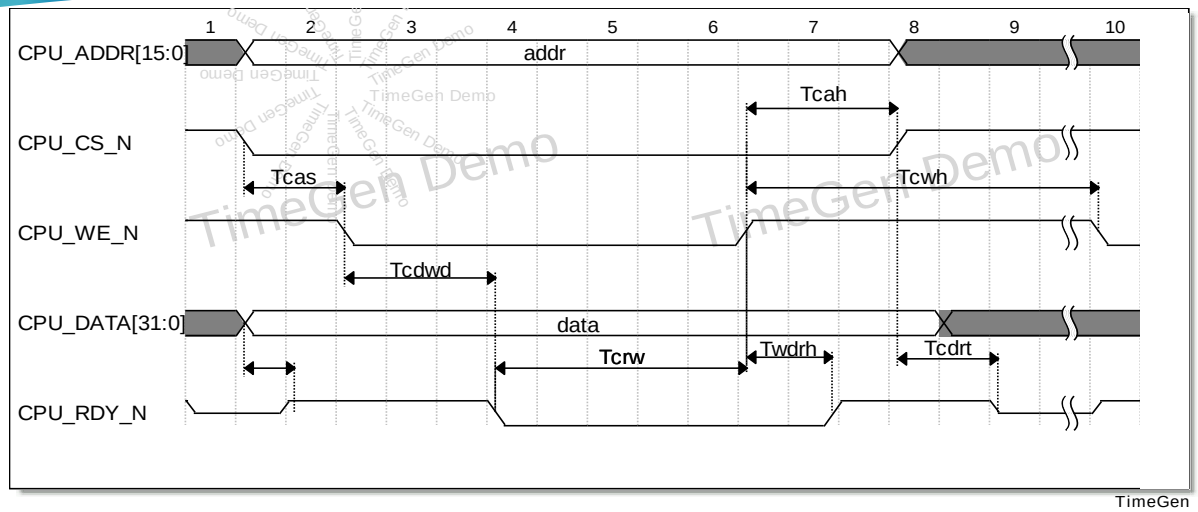


1. 由于是异步信号，需要打两拍,we_ff1；

2. cpu_we_n的下降沿有效，并且cpu_cs_n为0；

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

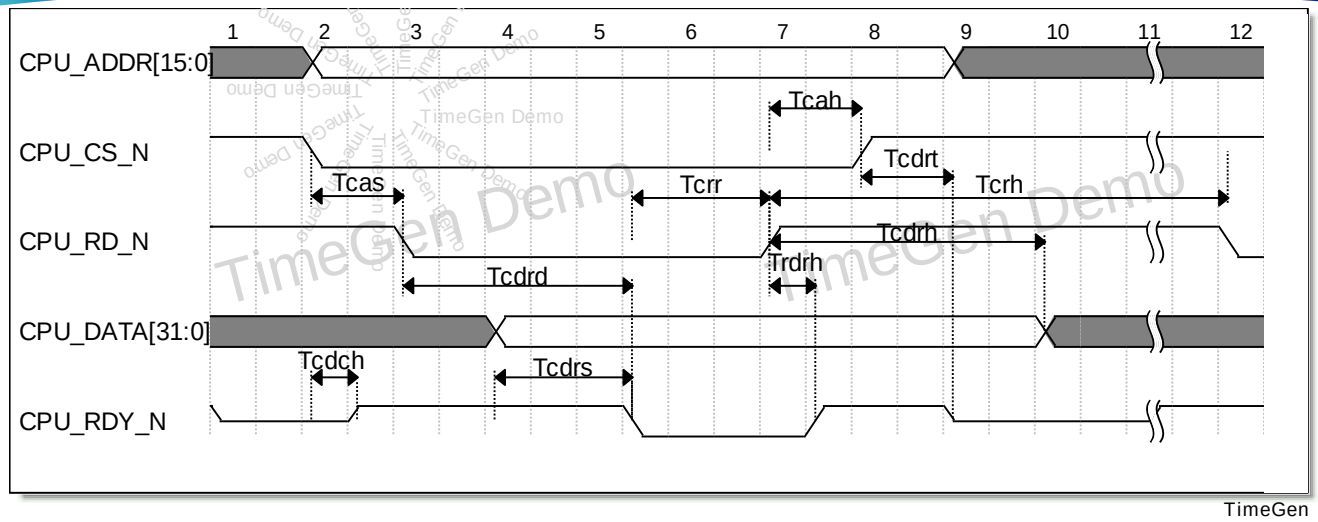
二、写时序s_addr s_data_r



1. s_wr_neg有效时，地址和数据在总线上稳定不变；

```
always  @(*)begin
    s_addr = cpu_addr;
end
always  @(*)begin
    s_data_r = cpu_data_r;
end
```

二、写时序cpu_data_w

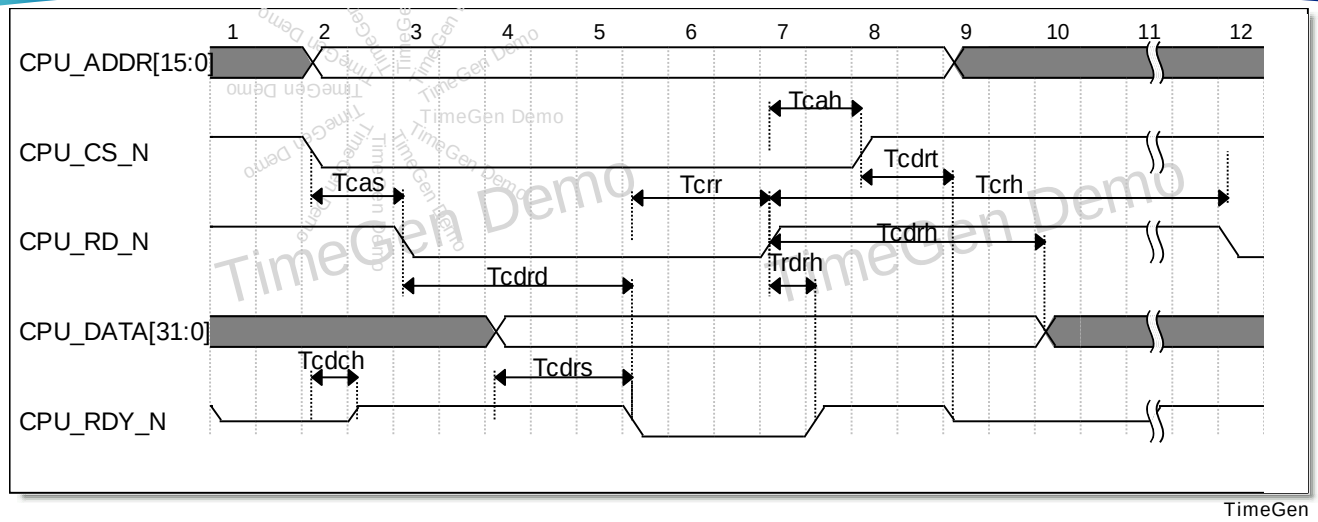


1. 在rdy变低之前，要将数据送到总线；
2. Tcdrs至少为10ns，最大没限制
3. 由于rdy是在rd_ff3时才拉底的，那么必须在rd_ff2或rd_ff1时把值送到总线
4. 最简单，检查到rd_ff1下降沿，即赋值。假设为s_rd_neg
5. 当地址为0时，送cfg_chan的值；当地址为1时，送cfg_mode的值；当地址为3时，送sta_fpga；当其他时，送0.

二、写时序cpu_data_w

```
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
```

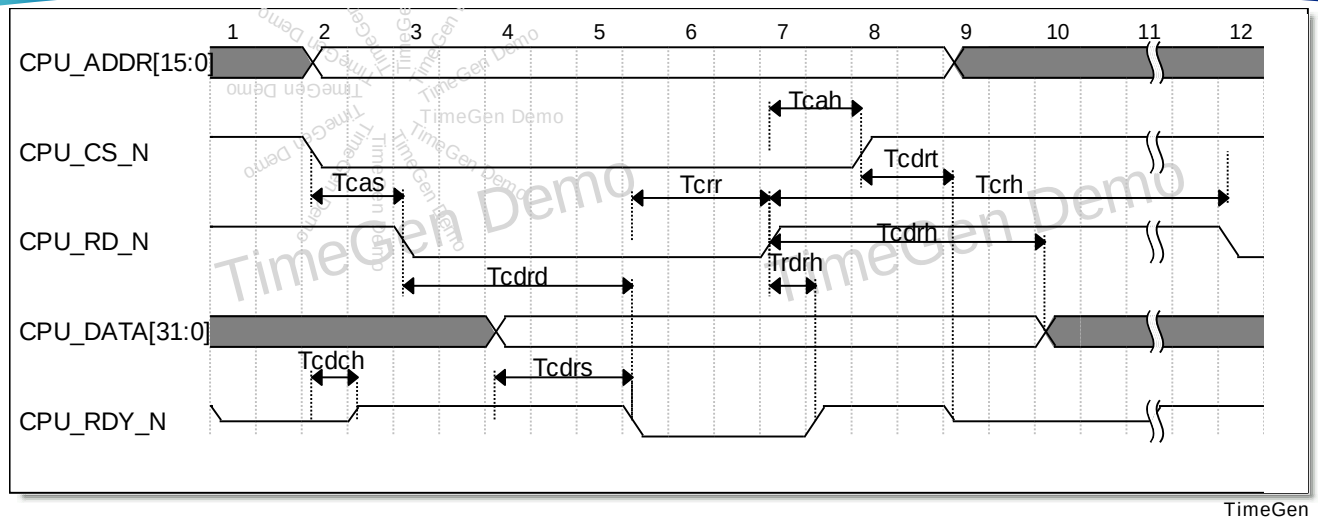
二、写时序cpu_data_w_e



1. 数据准备好，即可送出去；
2. 当检测到s_rd_neg有效时，值为1；
3. Tcdrh为30~120ns；
4. s_rd_ff1作条件时，在40~60之间
5. 要判断cpu_cs_n，以免误操作
6. 由于Tcah最少为10，因此要cs_ff1

```
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
```

二、写时序s_rd_neg



1. 由于是异步信号，需要打两拍,rd_ff1；

2. cpu_rd_n的下降沿有效，并且cpu_cs_n为0；

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

三、总结

1. 希望能掌握查看时序图的方法，并按照参数要求进行设计；

2. 仍然强调一个一个信号进行设计，将功能进行分解；

3. 测试时，要注意重点测试满足参数的边界点。

4. 每个企业的规范标准不同，更严格地如cpu_cs_n都需要打2拍后再使用。

明德扬科教

明德扬
科技·教育

QQ群： 97925396

官　网：http://www.mdy-edu.com

淘　宝：http://mdy-edu.taobao.com

Thank You !