# 哈尔滨理工大学

## 计算机科学与技术学院

# 课程设计报告

（2021-2022 第二学期）

课　　程　高级数字 IC 设计

题　　目　基于 AHB 总线的 SRAM 控制器的设计

专　　业　集成电路设计与集成系统

班　　级　集成 19-2

学　　生　黄羽铧

学　　号　1914020208

指导教师　杨兵

2022 年 6 月

# 目录

# 基于 AHB 总线的 SRAM 控制器的设计

## 1．前言

本设计为 AHB 接口的 SRAM 设计。为了初步对 AHB 总线进行了解与学习而进行的设计，主要为了设计 SRAM 与 AHB 总线通信的控制器，实现 SRAM 存储器与 AHB 总线的数据信息交换。

## 2.工作原理和整体结构

本设计一共分三个模块，分别是顶层模块 sram_top，从设备 interface 模块 sram_slave_if 与比较常见的 SRAM 模块 sram_core。本次设计中主要涉及的是 SRAM 从设备的 interface 的设计。sram_top 顶层模块主要负责把前面两个模块例化并连接起来。SRAM 在整个系统中作为缓存，SRAM 控制器实现 SRAM 存储器与 AHB 总线的数据信息交换，一端连接 AHB 总线，另一端连接 SRAM，将 AHB 总线上的读写操作转换成标准的 SRAM 读写操作。

SRAM 的工作原理很简单，SRAM 控制单元根据接收到的总线控制信号，将这些信号处理转化为 SRAM 存储器可以识别的信号，发送到 SRAM 存储器；之后，将经过地址译码的物理地址传送到存储器的地址总线，并将数据路径处理的数据送到 SRAM 存储器的数据总线。最后，SRAM 存储器进行相应的读写访问。如果是写操作，SRAM 控制单元的任务就完成了，SRAM 存储器已将数据信息按照要求写入。如果是读操作，SRAM 控制单元需要接收返回的读数据，将其送到数据路径，由数据路径将信息传输给 AHB 总线，最终实现总线对 SRAM 的读操作。

在设计过程中发现需要注意的是，AHB 的总线传输数据位宽有 8/16/32 这几种传输模式，在本设计中，采用了 4 个 SRAM，每个 SRAM 对应一个字。

## 3.模块设计

设计中采用的 SRAM 具有内嵌的 MBIST，使用的是从网上下载的 IP。设计的 SRAM 控制器一端连接 AHB，另一端连接 SRAM 存储器。控制器设计目标是实现 SRAM 存储器与 AHB 总线的数据信息交换。

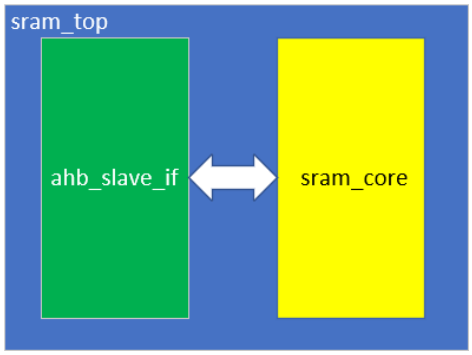



Figure 3.1 结构示意图

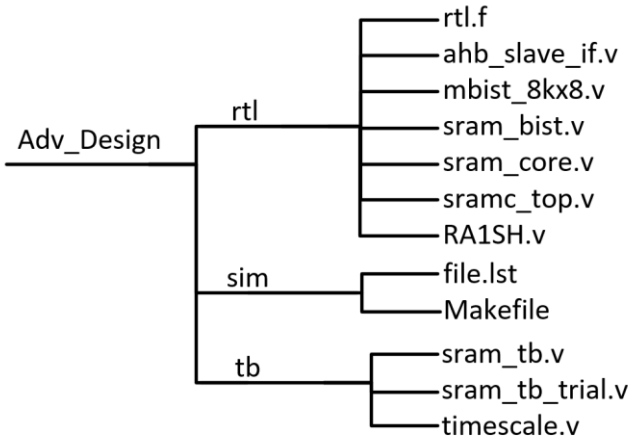



Figure 3.2 文件目录结构

## 3.1 AHB 总线控制 slave (ahb_slave_if.v)

SRAM 与控制器连接的接口，从 AHB 传输过来的数据信号都会经过总线控制单元的转换处理。来自官方手册的接口图如下所示：

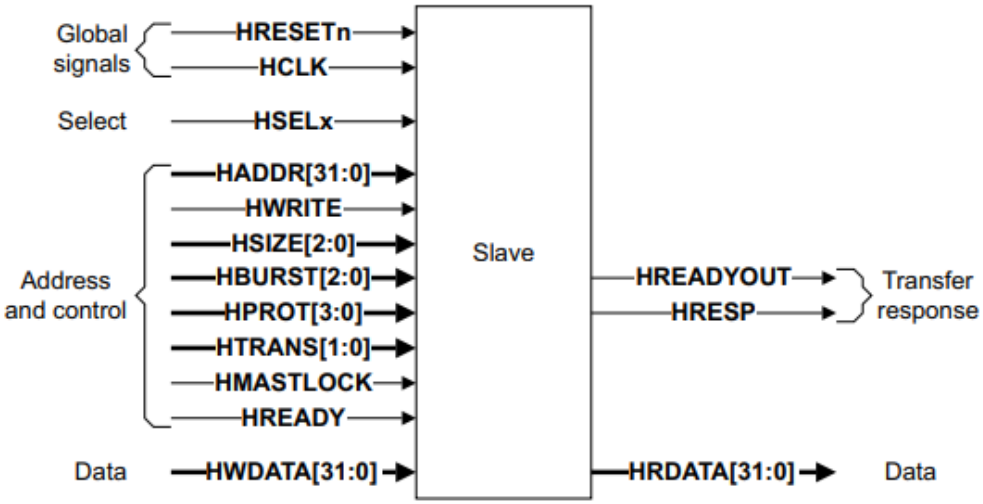



Figure 3.3 来自 arm 官方手册的 Figure1-3 对于 slave 的示例图

图中当 HSEL 信号有效时，总线控制单元与 SRAM 存储器控制单元之间可以进行信息传输，实现对 SRAM 的读写操作。总线控制单元做的就是接收来自 AHB 总线的地址和数据信息等等，将这些信息处理后送到 SRAM 单元，读写地址信息送到地址路径，写数据送到数据路径，并接收路径返回的数据。

## 3.2 SRAM 控制器（sramc_top.v）

SRAM 控制单元根据接收到的总线控制信号，将这些信号转化为 SRAM 存储器可以识别的信号，发送到 SRAM 存储器，之后将经过译码的物理地址传送到存储器的地址总线，

并将数据路径处理的数据送到 SRAM 存储器的数据总线，SRAM 存储器根据控制信号完成相应的读写访问操作。为了设计报告观感，模块代码放入下一小节。

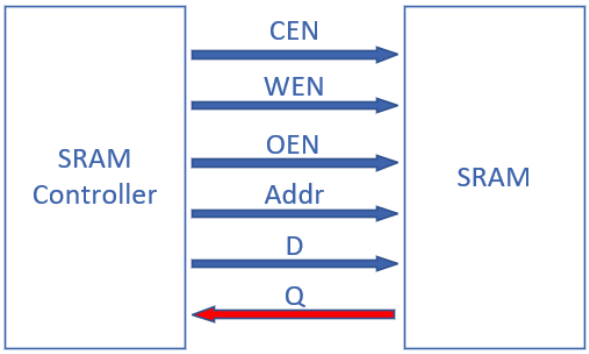



Figure 3.4 SRAM 控制器模块的示意图

## 3.3 SRAM

之前在课上学习的时候有学到，对于 SRAM 这种比较简单的结构来说，其数量，容量等参数是影响电路的功耗与性能的主要因素。本设计采用的是 8k x 8 的 SRAM。根据不同地址控制器会选择不同块的 SRAM。同时 SRAM 中分为上下两个库，库 0 包含 sram_bist 0~3，地址范围是 0 到 31；库 1 包括 sram_bist 4~7，地址范围是 32 到 63。

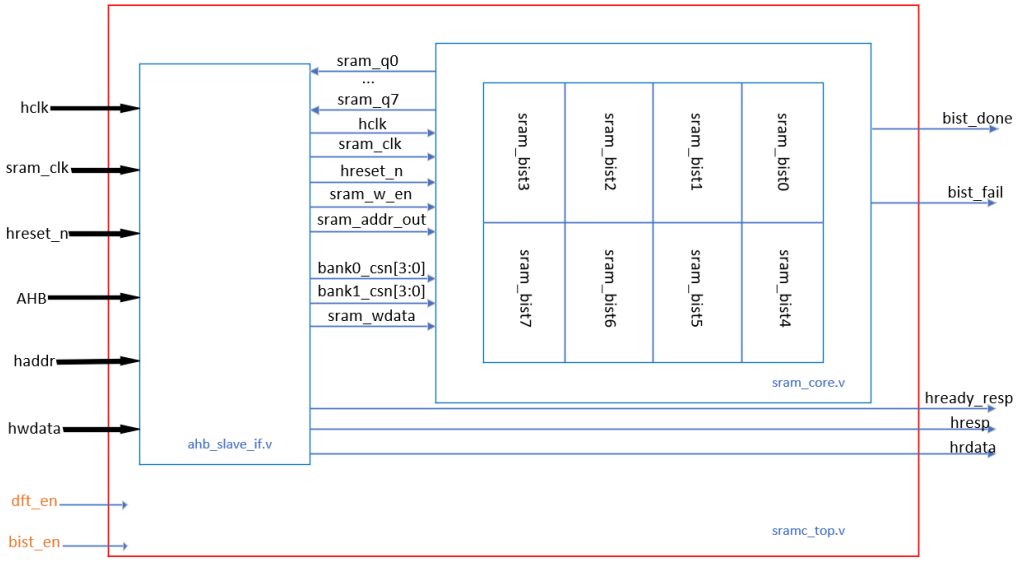



Figure 3.5 总体信号示例

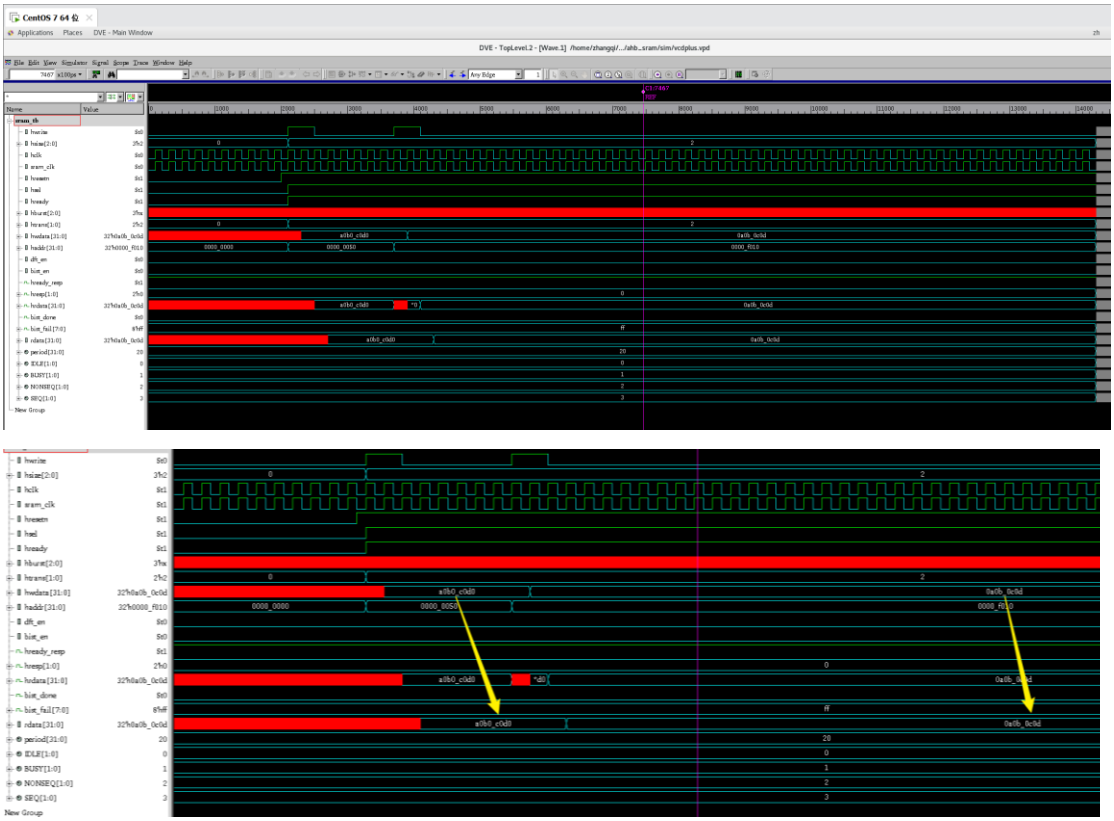
这样设计可以使得在片选中未被选中的 RAM 片功耗很低，优化了相关 PPA 指标。

# 4.RTL 设计

SRAM 控制器部分关键代码（ahb_sramc）

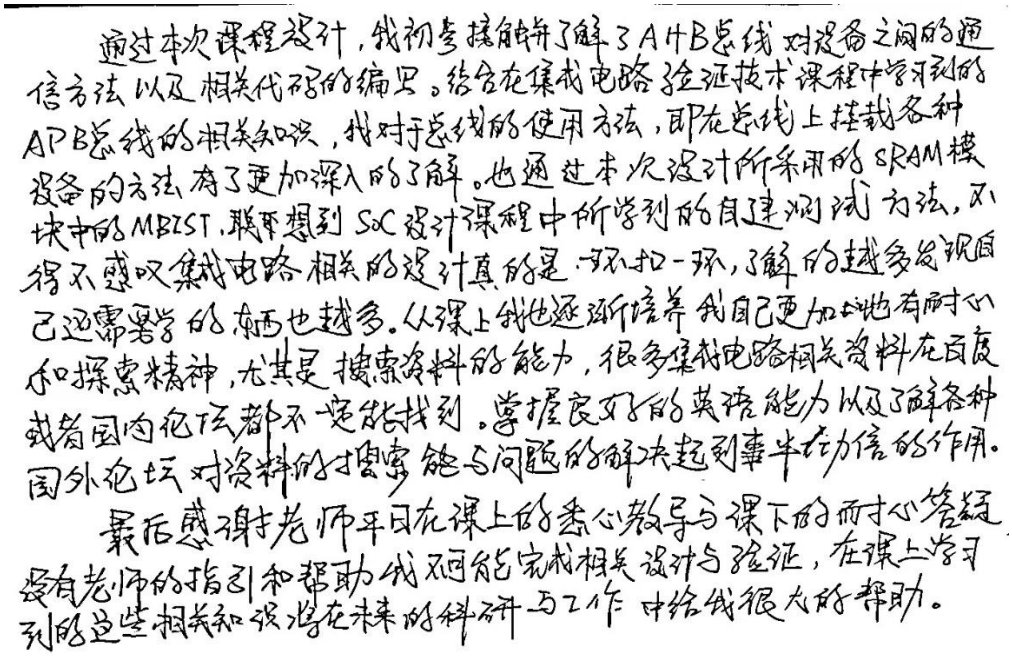
```
    /*选用单周期读写 SRAM 且 SRAM 一直处于“OK”状态*/
    assign hready_resp =1’b1;
    assign hresp =2’b00;

    /*数据从 SRAM 传输到 AHB 总线 */
    assign hrdata = sram_data_out

    /* 根据 AHB 总线协议，只有当 TRANS 信号为 NONSEQ 或者 SEQ 时，数据读写才有效，因此，需要保证 SRAM 读写时，TRANS 信号有效。SRAM 进行写操时，hwrite 信号为高；SRAM 进行读操作时，hwrite 信号为低 */
    assign sram_write = ((htrans_r == NONSEQ) || (htrans_r == SEQ)) && hwrite_r;
    assign sram_read = ((htrans_r == NONSEQ) || (htrans_r == SEQ)) && (!hwrite_r);

    /*写使能信号低有效*/
    assign sram_w_en = !sram_write;

/*每个 bank 有 4 片 SRAM，每一片 SRAM 有一个片选信号  */
always@(hsize_sel or haddr_sel)
begin
   if(hsize_sel == 2'b10)
        sram_csn = 4'b0;

/*一次片选两片 SRAM */
   else if(hsize_sel == 2'b01)
     begin
       if(haddr_sel[1] == 1'b0)
            sram_csn = 4'b1100;
       else
          sram_csn = 4'b0011;
     end
/*选择一片 SRAM，具体到 4 片中的某一片*/
    else if(hsize_sel == 2'b00)
      begin
       case(haddr_sel)
           2'b00 : sram_csn = 4'b1110;
           2'b01 : sram_csn = 4'b1101;
           2'b10 : sram_csn = 4'b1011;
           2'b11 : sram_csn = 4'b0111;
           default : sram_csn = 4'b1111;
         endcase
      end
    else
        sram_csn = 4'b1111;
end
```

# 5.仿真结果和分析



设计能够正确地把 AHB 总线输入的数据存到 SRAM 中，并且能够成功正确地读取出来。通过波形图可以看到设计中的各种信号都能够正常工作，设计目标的功能也成功实现，通过波形图可以看到控制器对于 SRAM 的地址的操作以及数据的操作。

# 6.总结




（草稿）通过本次课程设计，我初步接触并了解了 AHB 总线与设备之间的通信方法以及相关代码的编写。结合在集成电路验证技术课程中学习到的 APB 总线的相关知识，我对于总线的使用方法，即在总线上挂载各种设备的方法有了更加深入的了解。也通过本次设计所采用的 SRAM 模块中的 MBIST，联想到 SoC 课程中所学到的自建测试方法，不得不感叹集成电路相关的设计真的是一环扣一环，了解的越多发现自己还需要学的东西也越多。从课上我也逐渐培养我自己更加的有耐心和探索精神，尤其是搜索资料的能力，很多集成电路相关资料在百度或者国内论坛都不太好搜索，掌握良好的英语能力以及了解各种国外论坛能对与资料的搜索与问题的解决起到事半功倍的作用。

最后感谢老师在平日上课或课下的悉心教导，没有老师的指引和帮助我不可能完成相关设计，在课上学习到的这些相关知识将在未来的科研与工作中给我很大的帮助。

# 参考文献

[1]. ARM AMBA 5 AHB Protocol Specification AHB5, AHB-Lite Copyright © 2001, 2006, 2010, 2015 ARM Limited or its affiliates. All rights reserved.

[2]. Modeling and analysis of the AMBA bus using CSP and B January 2007 Concurrent Systems Engineering Series 65:379-398

[3]. IEEE Standard for Universal Verification Methodology Language Reference Manual Developed by the Design Automation Standards Committee of the IEEE Computer Society Approved 4 June 2020 IEEE SA Standards Board

# 附录

代码附录

```
module ahb_slave_if(

   //input signals
   input              hclk,
   input              hresetn,
   input              hsel,
   input              hwrite,
   input              hready,
   input    [2:0]    hsize,
   input [1:0]    htrans,
   input [2:0]    hburst,      // hard code -> parameter
   input [31:0] hwdata,
   input [31:0] haddr,

   //output signals
   output         hready_resp,
   output [1:0]  hresp,
   output [31:0]hrdata,

   //sram output
   input [7:0]      sram_q0, // 8bits
   input [7:0]      sram_q1,
   input [7:0]      sram_q2,
   input [7:0]      sram_q3,
   input [7:0]      sram_q4,
   input [7:0]      sram_q5,
   input [7:0]      sram_q6,
   input [7:0]      sram_q7,


   output         sram_w_en,          // 0:write, 1:read
   output [12:0]sram_addr_out,
   output [31:0] sram_wdata,         //写 sram 数据
   output [3:0]  bank0_csn,          //四字节可以单独写入
   output [3:0]  bank1_csn
   );

   //------------------------------------------------------
   //internal registers used for temp the input ahb signals
   //------------------------------------------------------
   //temperate all the AHB input signals
```

```
reg              hwrite_r;
reg [2:0]    hsize_r ;
reg [2:0]    hburst_r;
reg [1:0]    htrans_r;
reg [31:0] haddr_r;

reg [3:0]    sram_csn;

//------------------------------------------------------
//Internal signals
//------------------------------------------------------
//"haddr_sel " and "hsize_sel" used to generate banks of
//sram: "bank0_sel" and "bank1_sel".
wire [1:0] haddr_sel;
wire [1:0] hsize_sel;
wire bank_sel;

wire sram_csn_en;                 //sram chip select enable

wire sram_write;                  //sram write enable signal from AHB bus
wire sram_read;                   //sram read enable signal from AHB bus
wire [15:0] sram_addr;         //sram address from AHB bus
wire [31:0] sram_data_out; //data read from sram and send to AHB bus

parameter         IDLE      = 2'b00,
                    BUSY      = 2'b01,
                    NONSEQ = 2'b10,
                    SEQ       = 2'b11;

//------------------------------------------------------------
//    Combinatorial portion
//------------------------------------------------------------

//assign the response and read data of the ahb slave
//In order to implement the sram function-writing or reading
//in one cycle, the value of hready_resp is always "1".
assign hready_resp = 1'b1; // Singal Cycle
assign hresp    = 2'b00;         // OK

//------------------------------------------------------------
//sram data output to AHB bus
//------------------------------------------------------------
assign      hrdata = sram_data_out; //组合逻辑读，CPU 读 SRAM，地址有效则立即读
```

```
//Choose the right data output of the two banks(bank0, bank1) according
//to the value of bank_sel. If bank_sel = 1'b1, bank0 sleceted, or
//bank1 selected.
assign    sram_data_out = (bank_sel) ?
                                    {sram_q3, sram_q2, sram_q1, sram_q0} :
                                    {sram_q7, sram_q6, sram_q5, sram_q4} ;

//Generate sram write and read enable signals.
assign sram_write = ((htrans_r == NONSEQ) || (htrans_r == SEQ)) && hwrite_r;
assign sram_read =    ((htrans_r == NONSEQ) || (htrans_r == SEQ)) && (!hwrite_r);
assign sram_w_en = !sram_write;

//generate sram address
//SRAM 总寻址 64K 0x0--0xffff
assign sram_addr = haddr_r [15:0];//64K 8*8K
assign sram_addr_out = sram_addr[14:2]; // word

//Generate bank select signals by the value of sram_addr[15].
//Each bank(32kx32) comprises of four sram block(8kx8), and
//the width of the address of the bank is 15 bits(14~0), so
//the sram_addr[15] is the minimun of the next bank. If its
//value is "1", it means the next bank is selcted.
assign sram_csn_en = (sram_write || sram_read);
//低 32K bank0  高 32K bank1
assign bank_sel    = (sram_csn_en && (sram_addr[15] == 1'b0)) ? 1'b1 : 1'b0;
assign bank0_csn = (sram_csn_en && (sram_addr[15] == 1'b0)) ? sram_csn : 4'b1111;
assign bank1_csn = (sram_csn_en && (sram_addr[15] == 1'b1)) ? sram_csn : 4'b1111;

//signals used to generating sram chip select signal in one bank.
assign haddr_sel = sram_addr[1:0];
assign hsize_sel = hsize_r [1:0];

//-------------------------------------------------------
//data from ahb writing into sram
//-------------------------------------------------------
assign sram_wdata = hwdata;

//-------------------------------------------------------
//Generate the sram chip selecting signals in one bank.
//The resluts show the AHB bus write or read how many data
//once a time: byte, halfword or word.
//-------------------------------------------------------
always@(hsize_sel or haddr_sel) begin
   if(hsize_sel == 2'b10)//32bit
```

```
        sram_csn = 4'b0;
    else if(hsize_sel == 2'b01) begin//16bit
        if(haddr_sel[1] == 1'b0) //little-endian
            sram_csn = 4'b1100;
        else
            sram_csn = 4'b0011;
    end
    else if(hsize_sel == 2'b00) begin//8bit
        case(haddr_sel)
            2'b00 : sram_csn = 4'b1110;
            2'b01 : sram_csn = 4'b1101;
            2'b10 : sram_csn = 4'b1011;
            2'b11 : sram_csn = 4'b0111;
            default : sram_csn = 4'b1111;
        endcase
    end
    else
        sram_csn = 4'b1111;
end

//------------------------------------------------------------
//    Sequential portion
//------------------------------------------------------------

//tmp the ahb address and control signals
always@(posedge hclk , negedge hresetn) begin
    if(!hresetn) begin
        hwrite_r    <= 1'b0    ;
        hsize_r     <= 3'b0    ;
        hburst_r    <= 3'b0    ;
        htrans_r    <= 2'b0    ;
        haddr_r     <= 32'b0 ;
    end
    else if(hsel && hready) begin //hsel 要片选，否则信号一直在翻转，可能会功能错误，并且功耗大
        hwrite_r    <= hwrite ;//写 SRAM 时，把控制信号寄存。因为写操作时，要把地址打一拍，和数据对齐
        hsize_r     <= hsize    ;
//      hburst_r    <= hburst ;//AHB 中 master 会把 burst 传输所有地址传递过来，AXI 只传递起始地址。此处用处不大。减少一个 REG
        htrans_r    <= htrans ;
        haddr_r     <= haddr    ;
    end else begin
        hwrite_r    <= 1'b0    ;
```

```
            hsize_r     <= 3'b0    ;
            hburst_r    <= 3'b0    ;
            htrans_r    <= 2'b0    ;
            haddr_r     <= 32'b0 ;
        end
    end

endmodule
```