6 外部 SRAM 实验

6.1. 背景介绍

本章我们将通过 AHB 总线访问外部 SRAM，首先简单的了解一下 AHB 总线和 SRAM。

6.1.1. AHB 总线简介

AHB 总线规范是 AMBA 总线规范的一部分，AMBA 总线规范是 ARM 公司提出的，由于其规范严谨、功能丰富、总线效率高被大多数 SOC 设计采用。完整的 AHB 总线由四部分组成：

1. AHB 主设备 Master：发起依次读/写操作；某一时刻只允许一个主设备使用总线。
2. AHB 从设备 Slave：响应一次读/写操作；通过地址映射来选择使用哪一个从设备。
3. AHB 仲裁器 Arbiter：允许某一个主设备控制总线。
4. AHB 译码器 Decoder：通过地址译码来决定选择哪一个从设备。

其基本的组成框图如下图 6. 1 所示。

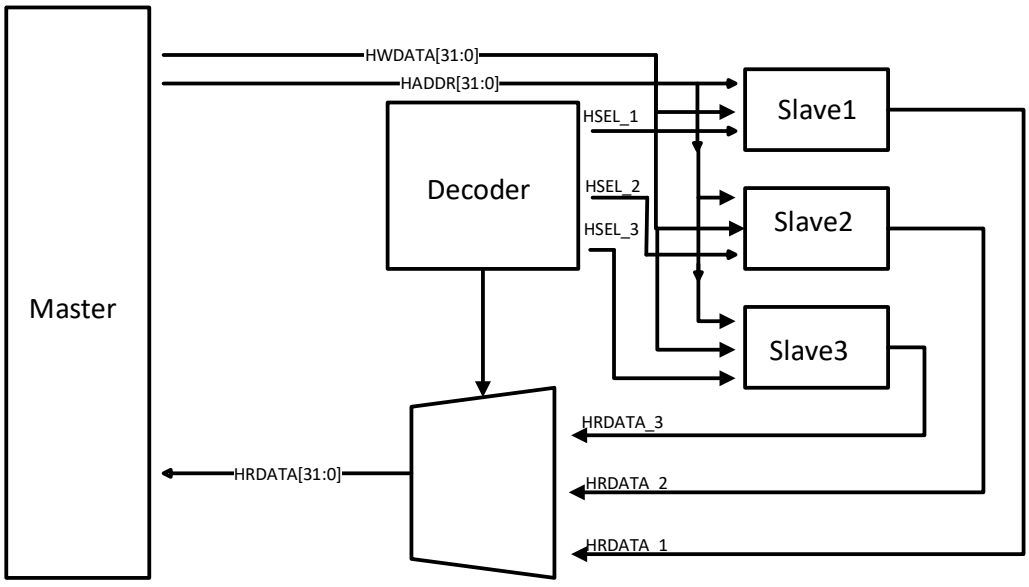

图 6. 1 AHB 总线组成框图

6.1.2. AHB 操作概述

当需要占用总线的时候，总线的 Master 向 Arbiter 发出请求，Arbiter 授权给指定的 Master。任一时间周期只允许一个 Master 接入总线，对其指定的 Slave 进

行读写操作。

总线授权的 Master 向 AHB 传输的步骤：首先发出地址和控制信号，然后提供地址信息、传输方向、带宽和 burst 类型。总线将会统一规划 Slave 的地址，译码器根据地址和控制信号确定哪个 Slave 与 Master 进行数据通信。为了避免出现三态总线，AHB 将读写总线分开，写数据总线将用于从 Master 到 Slave 的数据传输，读数据总线用于从 Slave 到 Master 的数据传输。每笔传输包括一个地址和控制周期，一个或者多个数据周期。地址和控制周期不能被扩展，因此 Slave 必须在一个周期内采样地址信号。数据周期可以通过 HREADY 信号扩展，当 HREADY 为低时，给传输加入等待状态以使 Slave 获得额外的时间来提供或采样数据，另外 Slave 通过可以响应信号 HRESP 反映传输状态。

AHB 支持批量式数据传送，可以自动递增地址。递增地址的方式分为：持续递增与回绕传送。一般情况下 Master 完成完整的 burst 传输，Arbiter 才会授权给其他的 Master 接入总线，然而为避免过大的判决延迟，Arbiter 也可能打断 burst 传输。在这种情况下 Master 必须再次接入总线以进行中断的 burst 剩余部分的传输。

6.1.3. AHB 信号描述

对于 AHB 信号描述如下表 6- 1 所示。

表 6- 1AHB 信号描述表

信号名	含义	源	I O	描述
HCLK	总线时钟	clock source	各 module	时钟为所有总线传输提供基准频率。所有信号时序都和 HCLK 的上升沿有关
HRESETn	复位	reset controller	各 module	总线复位信号，低电平有效，用于复位系统和总线
HSEL	从选择	Decoder	slave	每隔从机都有自己独立的从机选择信号，并且该信号可以表示当前传输的是否为选中的从机。
HADDR[31:0]	地址总线	Master	decoder;mux to slave;arbiter	32 位系统地址总线
HTRANS[1:0]	传送状态	Master	mux to slave	当前传输状态。
HSIZE[2:0]	传送带宽	Master	mux to slave	每一个 transfer 传输的数据大小，以字节为单位，最高支持 1024 位。
HWRITE	传送方向	Master	mux to slave	1 为写，0 为读
HWDATA[31:0]	写数据总线	Master	mux to slave	写数据总线，用来在写操作期间将数据从主机传送到从机，建议最小的数据总线宽度为 32 位。
HREADY	传送完成	Slave	mux to master; arbiter	该信号为高电平时，表示主机和所有的从机传输已完成
HREADYOUT	传送完成	Slave	mux to mast	高电平表示传输已完成，低电平表示可以

			er; arbiter	继续传输
HRDATA[31:0]	读数据总线	Slave	mux to master	读数据总线，通过译码选择通道
HRESP	传送响应	Slave	mux to master; arbiter	Slave 发给 Master 的总线传输状态。00：OKAY，传输完成；01：ERROR：传输错误；10：RETRY，传输未完成，请求主设备重新开始一个传输，Arbiter 会继续使用通常的优先级；11：SPLIT，传输未完成，请求主设备分离一次传输，Arbiter 会调整优先级方案以便其他请求总线的主设备可以访问总线。
HBURST[2:0]	批量传送	Master	mux to slave	表示传输是否组成了突发的一部分。支持 4 个，8 个，16 个节拍的突发传输，突发传输可以使增量或回环。
HPROT[3:0]	保护控制	Master	mux to slave	保护控制信号，需要 slave 带保护功能，一般不用

6.1.3.1. AHB 控制信号说明

1. 传送状态 HTRANS[1:0]

在 AHB 总线上，M 的传送状态可以由 HTRANS[1:0]来表示，关于这两位的说明如下表 6- 2 所示。

表 6- 2 传送状态位说明表

HTRANS[1:0]	类型	含义	描述
00	IDLE	空闲状态	此时控制 AHB 总线的 M（Master 得到允许后的信号），M 以 IDLE 表示没有数据传送请求。S 必须忽略此时的传送（已收到的地址与其它控制信号），并立刻以零等待 OKAY 信号来对 M 进行响应
01	BUSY	忙状态	M 以此信号表示正在处理数据，此时 SLAVE 要响应 OKAY 的信号（零等待状态），此时 SLAVE 会忽略已收到的地址和其它控制信号。一般在一个批量传送的中间，表示 M 发起一次批量传送，但是下一个传送不能立即进行，这样就使 AHB M 能在传送的中间插入空闲周期，这时候的地址和控制信号必须反映下一个传送数据的情况。作为 AHB 从设备必须忽略此时的传送，并以零等待的 OKAY 状态来响应 M。
10	NONSEQ	非连续状态	表示当前是单笔数据或突发传送的第一笔数据的传送，此时所送出的地址及控制信号与上一笔的传送无关。
11	SEQ	连续状态	表示突发传送剩余的数据传送时顺序传送，地址及控制信号与前面的操作相关。地址等于上一笔地址加控制信号 HSIZE 字节，控制信号与上一笔数据相同。

2. 批量传送 HBURST[2:0]

每次传送的数据大小由 HBURST[2:0]定义，说明如下表 6- 3 所示。

表 6- 3 批量传送状态位说明表

HBURST[2:0]	类型	描述

000	SINGLE	单笔数据传送
001	INCR	不定长的递增方式的批量传送
010	WRAP4	4 个数据的回绕方式的批量传送
011	INCR4	4 个数据的递增方式的批量传送
100	WRAP8	8 个数据的回绕方式的批量传送
101	INCR8	8 个数据的递增方式的批量传送
110	WRAP16	16 个数据的回绕方式的批量传送
111	INCR16	16 个数据的递增方式的批量传送

AHB 从设备通常支持 SINGLE，INCR，INCR4，WRAP8，INCR8，WRAP16，INCR16。AHB 对传送范围规定不超过 1KB，递增传送的大小为任意数目，但不可超过 1KB 的范围。

3. 传送方向 HWRITE

HWRITE 拉高时（写）,M 必须对写入动作初始化，数据会由 M 放到 HWDATA[31:0]总线上。HWRITE 拉低时（读）,M 会对读取动作初始化，被寻址到的 S 会将数据放到 HRDATA[31:0]总线上。

4. 传送大小 HSIZE[2:0]

传输数据大小由 HSIZE[2:0]控制，表示每次传送的字节数目，对其说明如下表 6- 4 所示。

表 6- 4 传输数据大小状态位说明表

HSIZE[2:0]	大小	描述
000	8 位	字节
001	16 位	半字
010	32 位	字
011	64 位	-
100	128 位	4-字数据线
101	256 位	8-字数据线
110	512 位	-
111	1024 位	-

HSIZE[2:0]和 HBURST[2:0]两个信号可合起来定义回绕地址的范围。例如：HSIZE[2:0]=32 位，HBURST[2:0]长度为 4 字节，则回绕地址必须对齐在 16 字节。通常的 AHB 从设备是 32 位数据线，所以它支持 8 位，16 位和 32 位 3 种大小。

5. 保护控制 HPROT[3:0]

提供总线访问的附加信息，主要是给那些希望执行某种保护级别的模块使用的。这个信号指示当前传输是否为预取指令或者数据传输，同时也表示传输是某种保护模式访问还是用户模式访问，对带存储器管理端元的总线主机而言，这些信号也用来指示当前传输时高速缓存（cache）还是缓冲的（buffer）。

6.1.4. AHB 模块接口

1. 从接口框图

从接口的框图如下图 6. 2 所示。

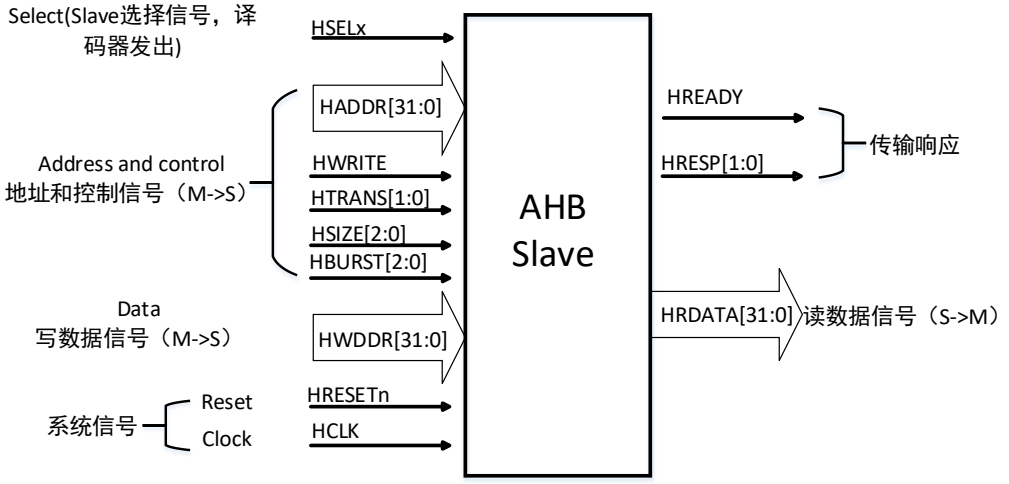

图 6. 2 从接口的框图

AHB 总线中的仲裁器决定哪一个 Master 拥有总线使用权之后，会将这个 Master 的数据地址、控制信号及预写入 Slave 的数据选出，并且送至每一个 Slave，所选出的数据地址会经由 AHB 译码器产生唯一的 HSELx 使能信号来启动 Slave 进行数据传送。Master 启动一个数据传送之后，被使能的 Slave 会发出 HREADY 信号来决定是否延长当前的数据传送，若 HREADY 为 0，表示此笔数据的传递必须被延迟，若 HREADY 为 1，表示能够完成此笔数据的传递。

由上图可以发现，Slave 除了用 HREADY 信号来告知此笔数据是否需要额外的延迟时间之外，还会通过 HRESP[1:0]信号响应当前传送的情形，具体的情形可以参看表 6- 1 中的内容。

2. 主接口框图

主接口的框图如图 6. 3 所示。

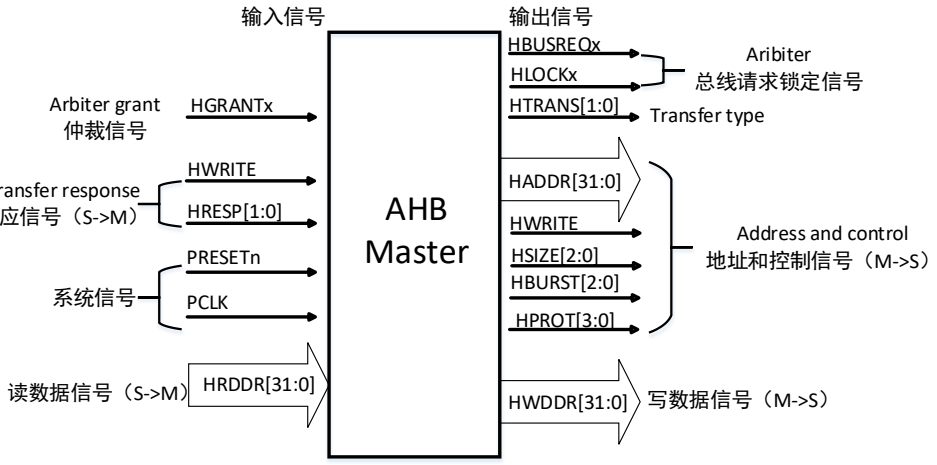

图 6. 3 主接口框图

每一次的数据传送可以分为四种型态，Master 用 HTRANS[1:0]信号来决定此次传送数据的型态，具体的传送型态可以参看表 6- 2 中的内容。

6.1.5. SRAM 简介

我们开发板上使用的 SRAM 芯片类型为 XM8A51216，它是由深圳星忆深圳星忆存储科技有限公司生产的一颗 16 位宽 512K（512*16，即 1M 字节）容量的 CMOS 静态内存芯片。具有以下几个特点：

1. 高速。具有最高访问速度 10/12ns。
2. 低功耗。
3. TTL 电平兼容。
4. 全静态操作。不需要刷新和时钟电路。
5. 三态输出。
6. 字节控制功能。支持高/低字节控制。

XM8A51216 的功能框图如下图 6. 4 所示。

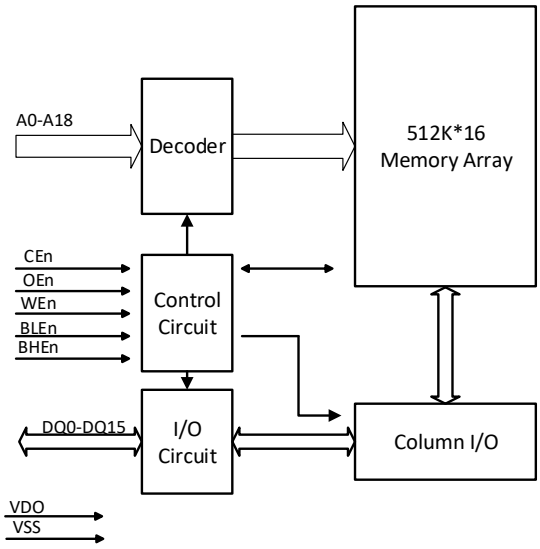

图 6. 4 XM8A51216 功能框图

上图中的 A0~A18 为地址线，总共 19 根地址线（即 2^19=512K，1K=1024）；DQ0~15，总共 16 根数据线，CEn 是芯片使能信号，低电平有效；OEn 是输出使能信号，低电平有效；WEn 是写使能信号，低电平有效；BLEn 和 BHEn 是高字节控制和低字节控制信号。

如果要写入设备的话，需要将芯片使能（CE）和写使能（WE）输入为低电平。如果字节低使能（BLE）为低，则来自 I/O 引脚（DQ0 至 DQ7）的数据被写入地址引脚（A0 至 A18）上指定的位置。如果字节高使能（BHE）为低电平，则来自 I/O 引脚（DQ8 至 DQ15）的数据将写入地址引脚（A0 至 A18）上指定的位置。要从器件读取的话，需要将芯片使能（CE）和输出使能（OE）设为低电

平，同时写入使能（WE）设为高电平。如果字节低使能（BLE）为低，则地址引脚指定的存储器位置中的数据出现在 DQ0 至 DQ7 上。如果字节高使能（BHE）为低，则来自存储器的数据出现在 DQ8 到 DQ15 上。

6.2. 实验介绍

SA5Z-30 内部集成两组 AHB 总线接口，通过 AHB 总线接口可以使 FPGA 外围设备与内部嵌入式 Cortex M3 硬核处理器实现数据通信。本次实验中将使用 AHB 总线拓展接口 0，将外部的 SRAM 作为从机挂接到 AHB 总线上，CM3 作为主机对 SRAM 进行操作。

6.3. 建立 HQFPGA 工程

将已有的 HQ 工程复制到存放本次实验代码的文件夹下，并且对其进行修改，操作方式参考实验二中 2.3 节的内容。

6.4. FPGA 侧代码设计

6.4.1. 复制 AHB 总线操作文件

找到我们提供的 SRAM 例程中的 ahb2sram 文件夹（SRAM/CM3_System/SRAM/rtl），将该文件夹复制至自己的工程文件夹中，如下图 6. 5 所示。

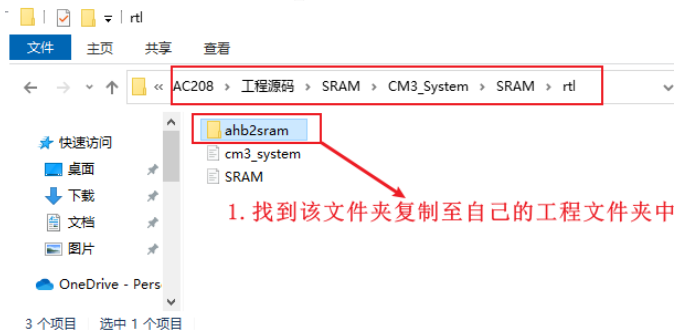

图 6. 5 复制 AHB 总线操作文件至工程文件中

6.4.2. HQ 软件中添加 AHB 总线操作模块

打开 HQFPGA 软件，点击工程属性，添加模块文件，如下图 6. 6 所示。

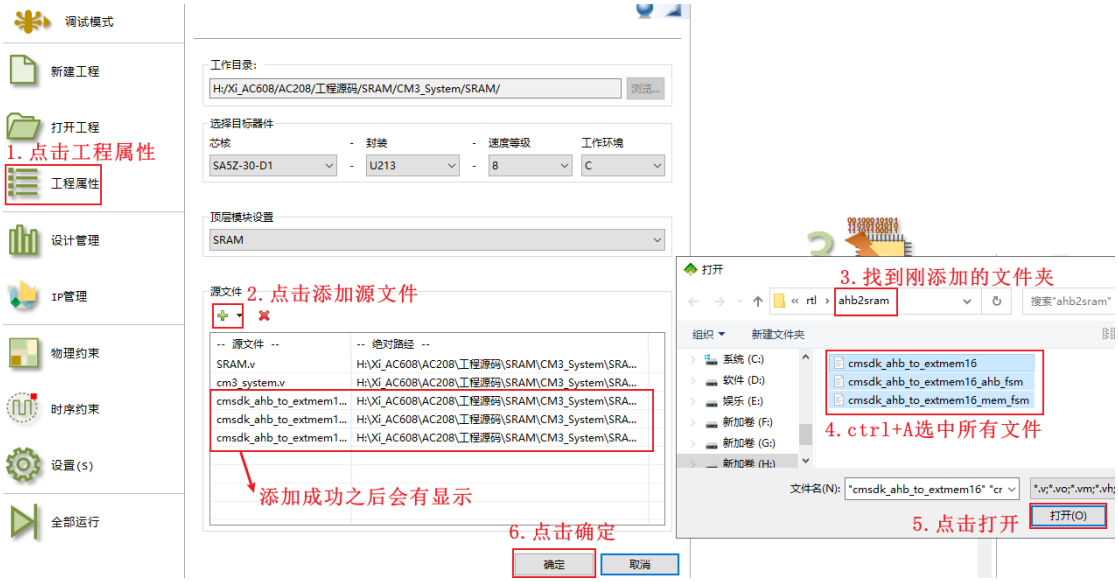

图 6. 6HQ 软件中添加模块文件

6.4.3. 修改 CM3_System 模块代码

点击设计管理，进入设计管理器，在 cm3_system 模块中添加 SRAM 端口，如下图 6. 7 所示。

```
module cm3_system(
    input    wire                MAIN_CLK,        //CM3时钟
    output   wire                CLKOUT,
    output   wire                PCLK,
    output   wire                MTX_CLK,
    input    wire                MTXRSTN,         //CM3复位

    input    wire                swdclk,          //debug接口
    inout    wire                swddio,          //debug接口
    inout[31:0]                  GPIO,            //32位通用GPIO

    //外部静态存储器SRAM端口
    output [18:0]SRAM_ADDR,
    inout [15:0]SRAM_DQ,
    output SRAM_nWE,
    output SRAM_nOE,
    output SRAM_nCE,
    output SRAM_nLB,
    output SRAM_nUB
    );
```

图 6. 7 添加外部 SRAM 端口

6.4.4. 例化 AHB 总线操作模块

在 cm3_system 中添加 AHB 总线操作模块，并将其端口提供给外部 SRAM 使用，并对 SRAM 的的地址信号和数据信号进行处理，需要添加的代码如下所示。

```
wire [19:0]SRAM_ADDR_Byte;  //以字节为单位的 SRAM 地址
assign SRAM_ADDR = SRAM_ADDR_Byte[19:1];     //19 位 SRAM，地址去掉最低位
wire SRAM_nDQoe;
```

```
wire [15:0]SRAM_DQo;
assign SRAM_DQ = SRAM_nDQoe?16'hzzzz:SRAM_DQo;
cmsdk_ahb_to_extmem16
#(
  .AW (20)
)
u_ahb_to_extmem16 (
  .HCLK          (PLL_OUT), //时钟
  .HRESETn       (MTXRSTN), //复位
  .HSEL          (TARGEXP0HSEL), // AHB inputs，设备选择
  .HADDR         (TARGEXP0HADDR[20-1:0]), //地址
  .HTRANS        (TARGEXP0HTRANS), //传输控制信号
  .HSIZE         (TARGEXP0HSIZE),  //传输大小
  .HWRITE        (TARGEXP0HWRITE), //写控制
  .HWDATA        (TARGEXP0HWDATA), //写数据
  .HREADY        (TARGEXP0HREADYMUX), //传输完成
  .HREADYOUT     (TARGEXP0HREADYOUT), // AHB Outputs，设备准备信号
  .HRDATA        (TARGEXP0HRDATA), //读取到的数据
  .HRESP         (TARGEXP0HRESP), //设备响应
   // 配置信号
  .CFGREADCYCLE      (2),   // Read cycle
  .CFGWRITECYCLE     (0),  // Write cycle
  .CFGTURNAROUNDCYCLE(0),  // Turn around cycle
  .CFGSIZE           (1),     // Size (0 = 8-bit, 1 = 16-bit)
   // 连接外部存储器
  .DATAIN        (SRAM_DQ),      // data input
  .ADDR          (SRAM_ADDR_Byte),        // address output
  .DATAOUT       (SRAM_DQo),     // data output
  .DATAOEn       (SRAM_nDQoe),     // output enable (active low)
  .WEn           (SRAM_nWE),         // write control (active low)
  .OEn           (SRAM_nOE),         // read control  (active low)
  .CEn           (SRAM_nCE),         // Chip Enable   (active low)
  .LBn           (SRAM_nLB),         // Lower Byte    (active low)
  .UBn           (SRAM_nUB)          // Upper Byte    (active low)
);
```

在上述代码中，我们可以看到在将 SRAM 的地址连接至 AHB 总线上的时候并不是直接相连接，而是去掉地址最低位之后连接至 AHB 端口，下面就其这样操作进行说明。

首先，我们使用到的 SRAM 芯片一共有 19 根地址线，16 根数据线，故其存储空间应该为 2^{19}*16bit=2^{20}*8=2^{20}byte=1024KB。可以看到如果需要寻址 1024KB 的存储空间，需要 20 根数据线，而 SRAM 的引脚却只有 19 根的数据线，真正

的存储空间只有 512KB。其实这是因为 SRAM 芯片的数据总线是 16 位的数据位宽，即一次传输 2 个字节的数据，也就是说一个地址空间其实对应着 2 个字节的数据，那么 SRAM 真正的寻址空间就为 512KB，19 根的数据线足够。这样在连接至 AHB 总线上的时候，并不需要将 20 根的地址线全部连接至 SRAM 芯片上，只需要给出高 19 位的地址，然后，SRAM 芯片即根据这 16 根地址找到对应的地址空间，将该地址空间的 2 字节发送到数据总线上，然后根据 AHB 总线连接到 SRAM 芯片的地址最低位 SRAM_ADDR_Byte[0]，由该位的值来决定取高字节还是低字节。

由上诉分析得出一个规律：AHB 控制端口的地址线到底是 A0 、A1 还是 A2 连接存储芯片的 A0，取决于存储系统的数据总线位宽，如果是 8 位则 A0→A0，如果是 16 位则 A1→A0，如果是 32 位则 A2→A0。

6.5. 物理管脚约束

在设计管理器中点击设计文件，打开 ac208.upc 文件，启动对于 SRAM 的引脚分配，如下图 6. 8 所示。修改完成之后，保存文件。

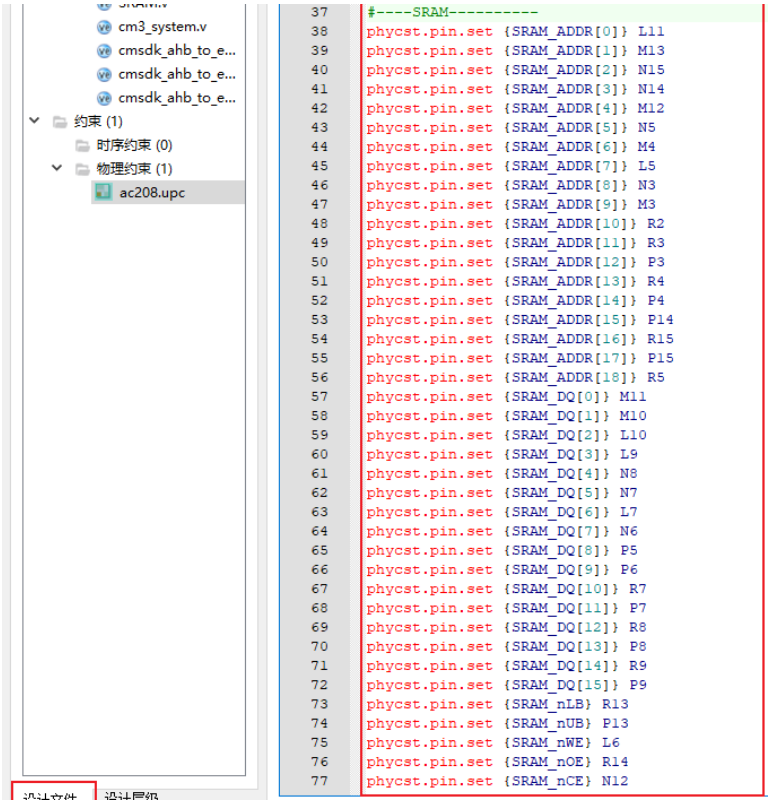

图 6. 8 分配 SRAM 的引脚

6.6. 编译设计

点击“全部运行”按钮，对设计进行全编译并生成 bin 文件。操作步骤如下图 6. 9 所示。

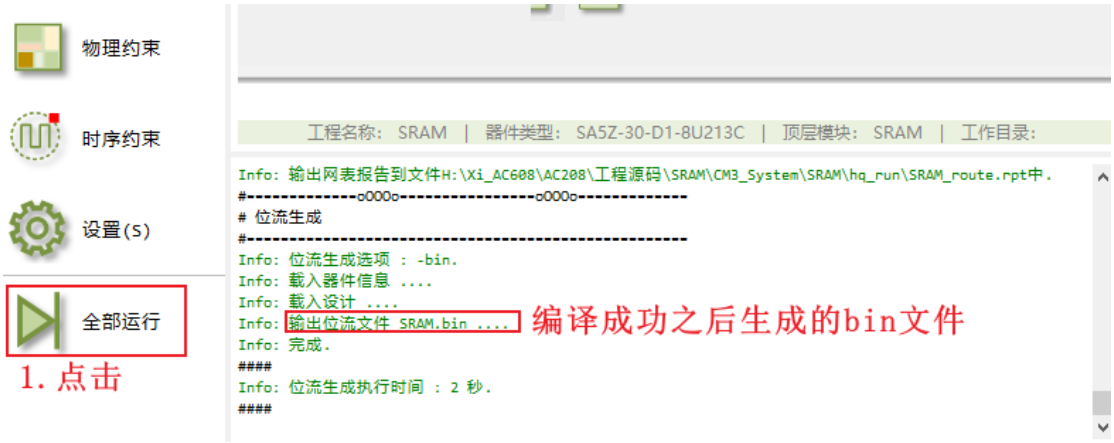

图 6. 9 编译运行整个文件

6.7. 建立 MDK 工程

将已有的 MDK 工程复制至本次工程的文件下，并对其进行修改，操作方式参考实验二中 2.8 节的内容。

6.8. 软件设计

实验功能：完成对外部 SRAM 的写操作，并且通过串口打印写入的数据。

实验代码：在代码中定义一个超大数组 testsram，我们指定该数组定义在外部 SRAM 的起始地址，本次实验中使用的是 AHB 扩展端口 0，对于 AHB 两个扩展口的说明如下表 6- 5 所示。

表 6- 5AHB 扩展端口说明表

Type	Strat	End	Peripheral	Size	Subsystem connection	Comment	Brietgbiaonnd
Peripherals	0xA000_0000	0xBFFF_FFFF	AHB Expansion0	512MB	TARGEXP0	connected to FPGA fabric	-
	0xC000_0000	0xDFFF_FFFF	AHB Expansion1	512MB	TARGEXP1	connected to FPGA fabric	-

根据上表中的内容，定义数组的起始地址为 0xA000_0000(__attribute__((at(CM3DS_MPS2_TARGEXP0_BASE)))。然后初始化串口，将串口的波特率设置为 9600，然后通过循环向数组中写入 250000 个数据，再将写入 SRAM 的数据通过串口打印出来。代码如下所示：

```
#include "CM3DS_rcc.h"
#include "CM3DS_gpio.h"
#include "CM3DS_uart.h"
```

```
#include "CM3DS_MPS2.h"
#include "CM3DS_spi.h"
#include "uart.h"
uint32_t testsram[250000] __attribute__((at(CM3DS_MPS2_TARGEXP0_BASE)));
int main()
{
    uint32_t ts;
    Uart_Init(CM3DS_MPS2_UART0,9600);//初始化串口
    for(ts=0;ts<250000;ts++)
        testsram[ts]=ts;//预存测试数据
    for(ts=0;ts<250000;ts++)
        printf("data[%d]:%d\r",ts,testsram[ts]);
    return 0;
}
```

6.9. 板级验证

6.9.1. 实验所需硬件

(1) AC208 开发板
(2) FPGA 下载器：XIST USB Cable
(3) CM3 仿真器：DAP Link
(4) 电源线一根

6.9.2. 硬件连接

硬件连接参考实验一。

6.9.3. 下载文件至目标板

下载文件的方式参考实验一。

6.9.4. 功能演示

打开串口助手，将波特率设置为 9600，此时我们可以看到串口连续打印写入 SRAM 的数据，如下图 6. 10 所示。

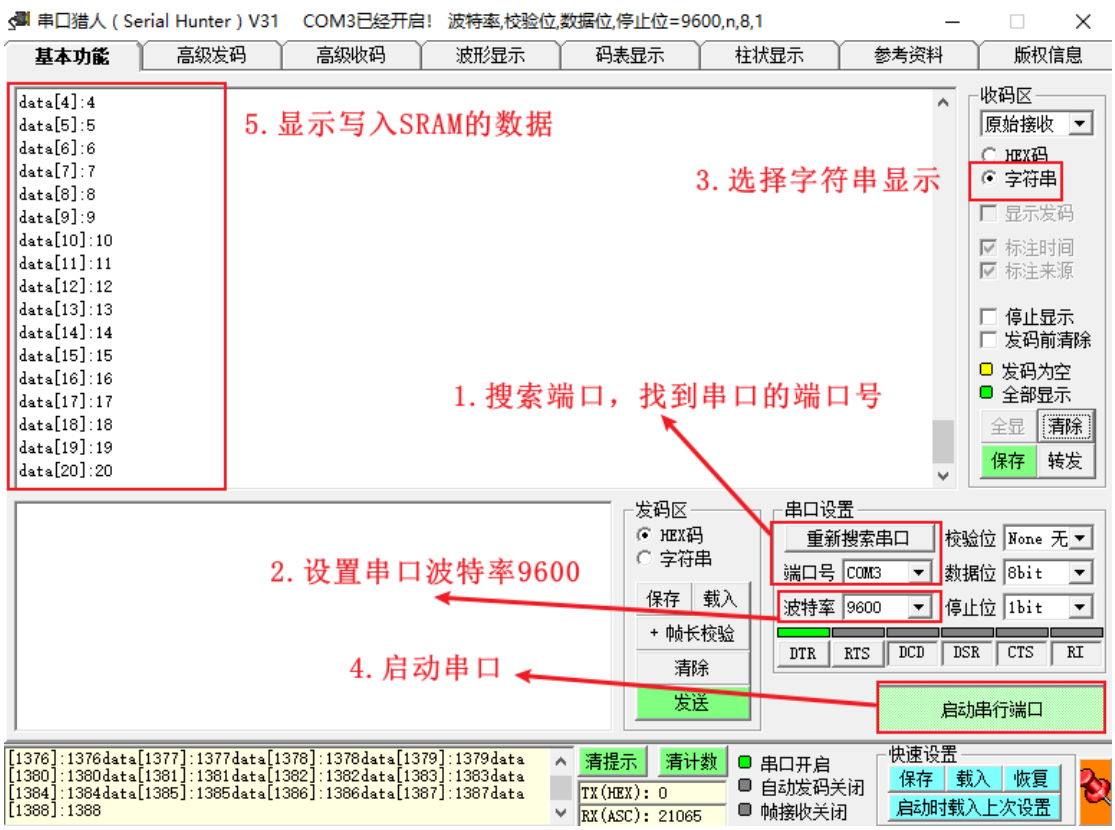

图 6. 10 串口显示写入 SRAM 的数据

6.10. 思考与总结

本次实验通过 AHB 总线成功实现了向 SRAM 中写入数据的操作，需要注意的是，并不是直接将 SRAM 的地址信号直接连接至 AHB 总线模块上，而是定义一个 20 位的以字节为单位的 SRAM 地址 SRAM_ADDR_Byte，SRAM_ADDR_Byte[19:1]对应着 SRAM 的地址信号，SRAM_ADDR_Byte[0]的值决定取高字节还是低字节的数据。