13 PDS 软件中 RAM 存储器 IP 配置和使用

工程源码	
相关视频课程	

章节导读

RAM 的英文全称是 Random Access Memory，即随机存取存储器，它可以随时把数据写入任一指定地 址的存储单元，也可以随时从任一指定地址中读出数据，其读写速度是由时钟频率决定的。RAM 主要用来存放程序及程序执行过程中产生的中间数据、运算结果等。

13.1 实验任务

本章我们将对 PDS 软件生成的 RAM IP 核进行读出测试，并向大家介绍“DRM Based Simple Dual Port RAM ”IP 核的使用方法。

13.2工程建立

新建 PDS 工程，首先在菜单栏里选择“Tools”然后单击 “IP Compiler”选项，如图 13-1 所示。

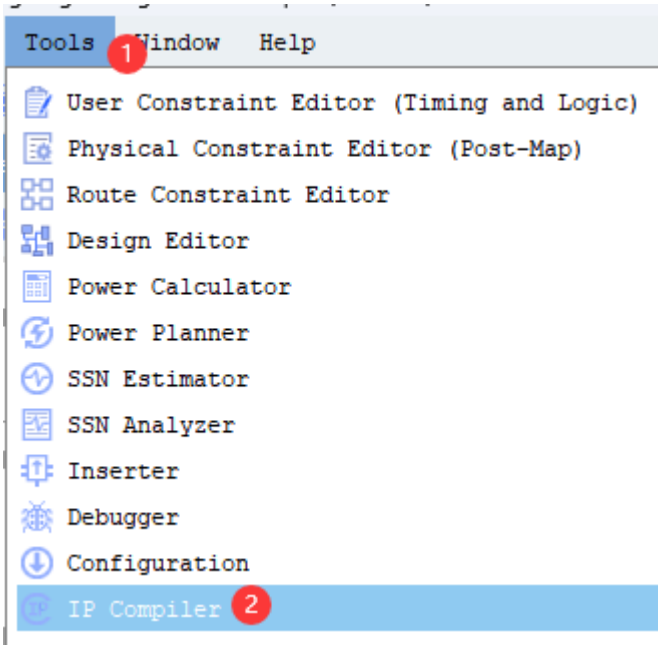

图 13-1“IP Compiler”页面

点击图 13-1 中的“IP Compiler”后会跳转图 13-2 页面。如图 13-2 所示

Memory IP 分为两种，一种是“Distributed RAM”，另一种是“DRM”。

“Distributed RAM”IP 是紫光同创基于 FPGA 片内资源设计的 IP，适用于全系列 FPGA 产品，用户可以通过公司 PDS 软件完成 IP 模块的配置和生成。“Distributed RAM”IP 包含以下五种 IP：

1. Distributed FIFO：分布式 FIFO;
2. Distributed ROM：分布式 ROM;
3. Distributed Shift Register：分布式移位寄存器;
4. Distributed Simple Dual Port RAM：分布式简单双端口 RAM;
5. Distributed Single Port RAM：分布式单口 RAM;

“DRM”IP 通过对 DRM 的级联调用实现 RAM/ROM/FIFO 等 IP 设计，“DRM”IP 包含以下五种 IP：

1. DRM Based Dual Port RAM：基于 DRM 的双端口 RAM;
2. DRM Based FIFO：基于 DRM 的 FIFO;
3. DRM Based ROM：基于 DRM 的 ROM;
4. DRM Based Simple Dual Port RAM：基于 DRM 的简单双端口 RAM;
5. DRM Based Single Port RAM：基于 DRM 的单端口 RAM;

本章实验我们选择基于 DRM 的简单双端口 RAM;

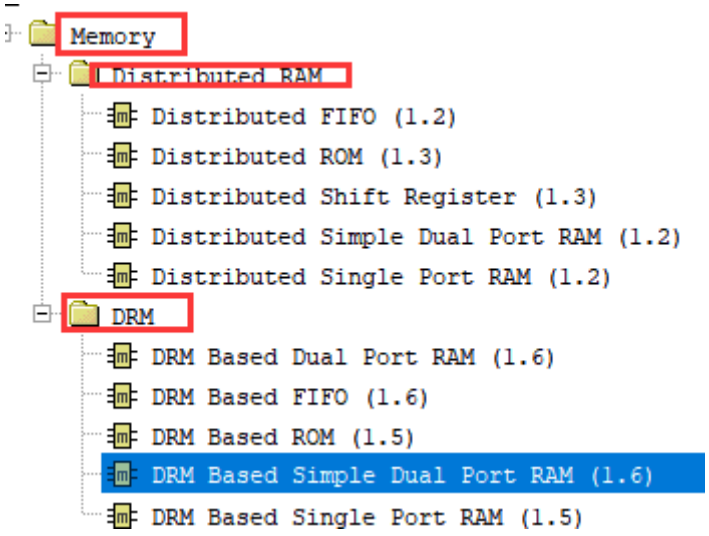

图 13-2 RAM IP 核类型

在图 13-3 页面中，首先我们点击标签 1 中的“DRM Based Simple Dual Port

RAM”按钮，选择创建 RAM 核。然后将标签 2 中“Instance Name”选项命名为“ram”。最后点击标签 3 中的“Customize”按钮进入 RAM IP 参数配置页面。

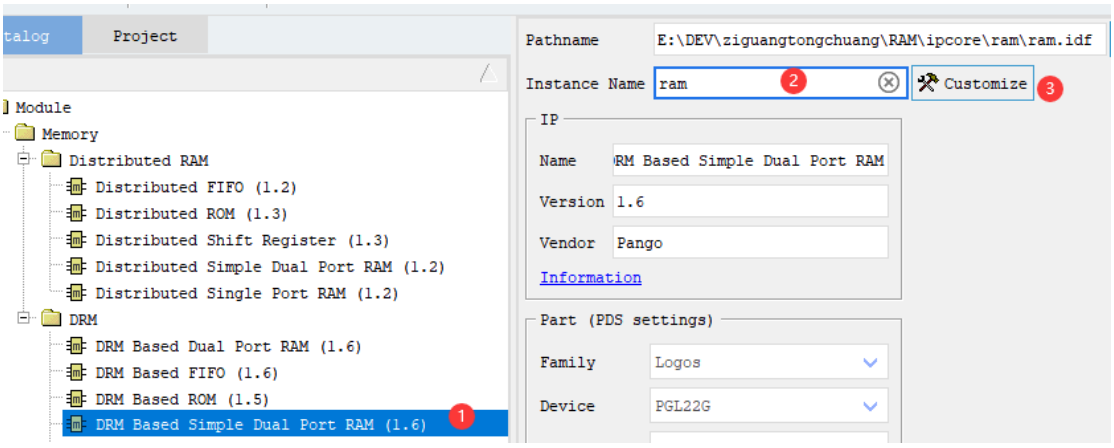

图 13-3 IP 选择页面

点击图 13-3 中的“Customize”按钮进入图 13-4 中。

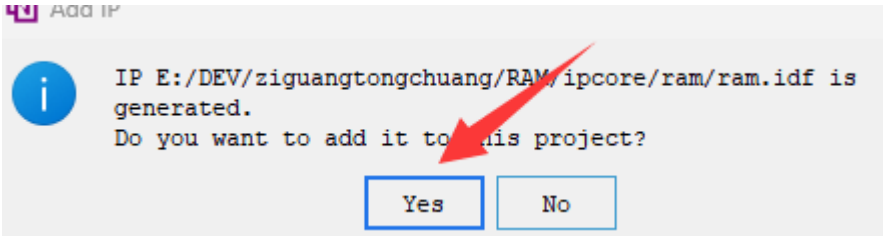

图 13-4“Add IP 弹窗

图 13-4 中表示的是 ram.idf 文件已经生成，询问我们是否将该文件加载到我们的工程中。因为这个 IP 我们后续是要在工程中用到的，因此这里直接点击图 13-4 中的“Yes”按钮即可。然后软件会自动进入 RAM 存储器的参数配置界面，如图 13-5 所示。图中展示了 RAM 存储器的若干个相关参数，包括数据位宽，地址位宽，初始化内容等，以下分别介绍：

标签 1 中“DRM Resource Type”（DRM 资源类型），一般我们设置为“AUTO”模式, 即自动模式让 PDS 软件自动选择 DR 资源类型；

标签 2 中“Write/Read Port Use Same Data Width”（读写端口位宽是否一致），这里我们对该选项进行勾选，勾选后，我们在本实验中只需要对写入端口的地址位宽和数据位宽进行设置即可；

标签 3 中 “Address Width”(地址位宽)，地址位宽的合法范围是 5-20，该实验我们设置成 8, 则 RAM 存储的存储容量为 255 个存储单元；

标签 4 中“Data Width”(数据位宽)，代表了 RAM 存储器每个地址（存储

单元）中存储的数据的位宽，可根据实际应用需求在 1 位到 1152 位之间任意设定,这里我们设置为 8；

标签 5 中“Reset Type”（复位方式,：复位方式一共有三种："ASYNC"异步复位，"SYNC"同步复位，"Sync_Internally "异步复位同步释放，本实验选择 ASYNC"即异步复位方式。

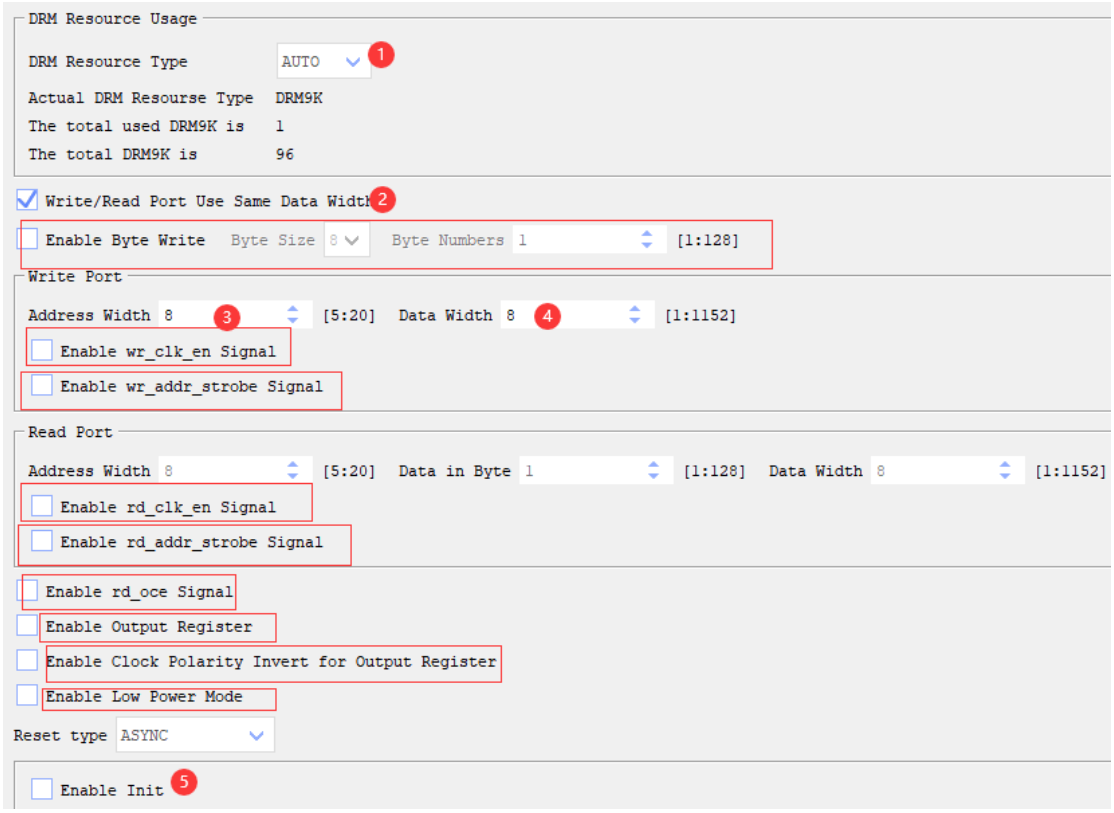

图 13-5 ram 配置界面

在本章实验中对 RAM IP 做出以上五种设置，在图 13-5 中还有一些其它的设置没有涉及，如：

“Enable Byte Write”（使能 Byte Write 功能），以字节为单位写入数据。当勾选该选项时需要对“Byte Size”（字节位宽，字节位宽可以设置为 8 或 9）和“Byte Numbers”（字节个数）进行配置。举个例子，当写入数据是 40bit,“Byte Size”设置为 8，那么就需要 4 个写使能信号。本章不使用 Byte Write 功能，保持默认不用勾选。

Enable wr_clk_en Signal：配置使能写端口的 wr_clk_en 信号，本实验保持默认不用勾选。

Enable wr_addr_strobe Signal：配置使能写端口的 wr_addr_strobe 信号，本实验保持默认不用勾选。

Enable rd_clk_en Signal：配置使能读端口的 rd_clk_en 信号，本实验保持默认不用勾选。

Enable rd_addr_strobe Signal：配置使能读端口的 rd_addr_strobe 信号，本实验保持默认不用勾选。

Enable rd_oce Signal：配置使能 rd_oce 信号，本实验保持默认不用勾选。

Enable Output Register：输出寄存器选项。如果勾选了“Enable Output Register”信号， 虽然会改善代码的时序性能，但会使输出的数据延迟一拍，这不利于我们在仿真窗口中直观清晰地观察信号，所以本实验未使用保持默认不用勾选。

Enable Clock Polarity Invert for Output Register：配置读使能端口输出时钟反向极性，端使能读口输出时钟反向极性时，必须要勾选读使能端口输出寄存（Enable Output Register），所以本实验保持默认不用勾选。

Enable Low Power Mode：配置是否使能低功耗模式，本实验不需要配置成低功耗模式，本实验保持默认不用勾选。

Enable Init：配置是否使能对当前 RAM 进行初始化，这里不需要该配置即保持默认不用勾选。

Init File：配置使能对当前 RAM 进行初始化，指定初始化文件路径，若不指定，则生成初始值为全 为“0”的初始化文件.v 文件。

File Type：配置初始化文件数据格式：有两种方式一种是"BIN"二进制，另一种是"HEX"十六进制。

至此本实验所需要的简单双单口 RAM IP 已介绍并配置完成，接下来点击图 13-6“Customize IP”窗口左上角的 “Generate”按钮即可。

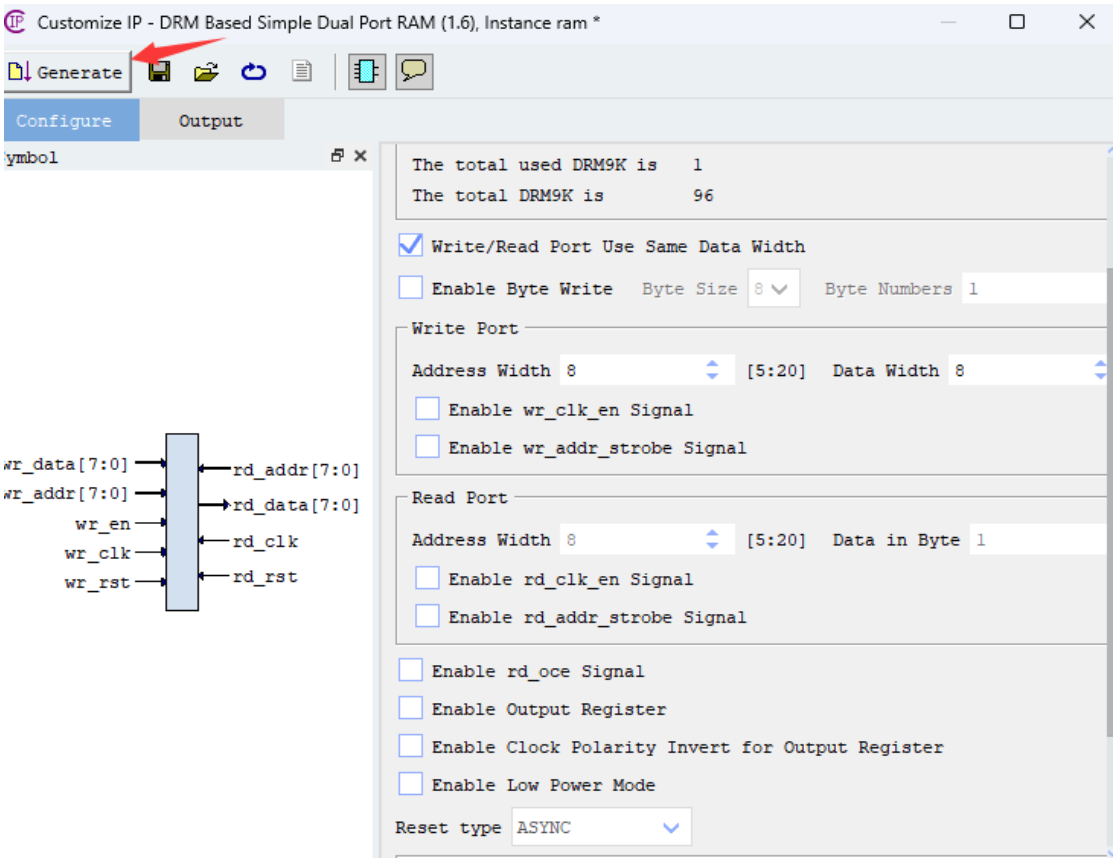

图 13-6“Customize IP”窗口

点击图 13-6 中的“Generate”按钮，进入图 13-7 界面。至此，RAM IP 核配置成功。

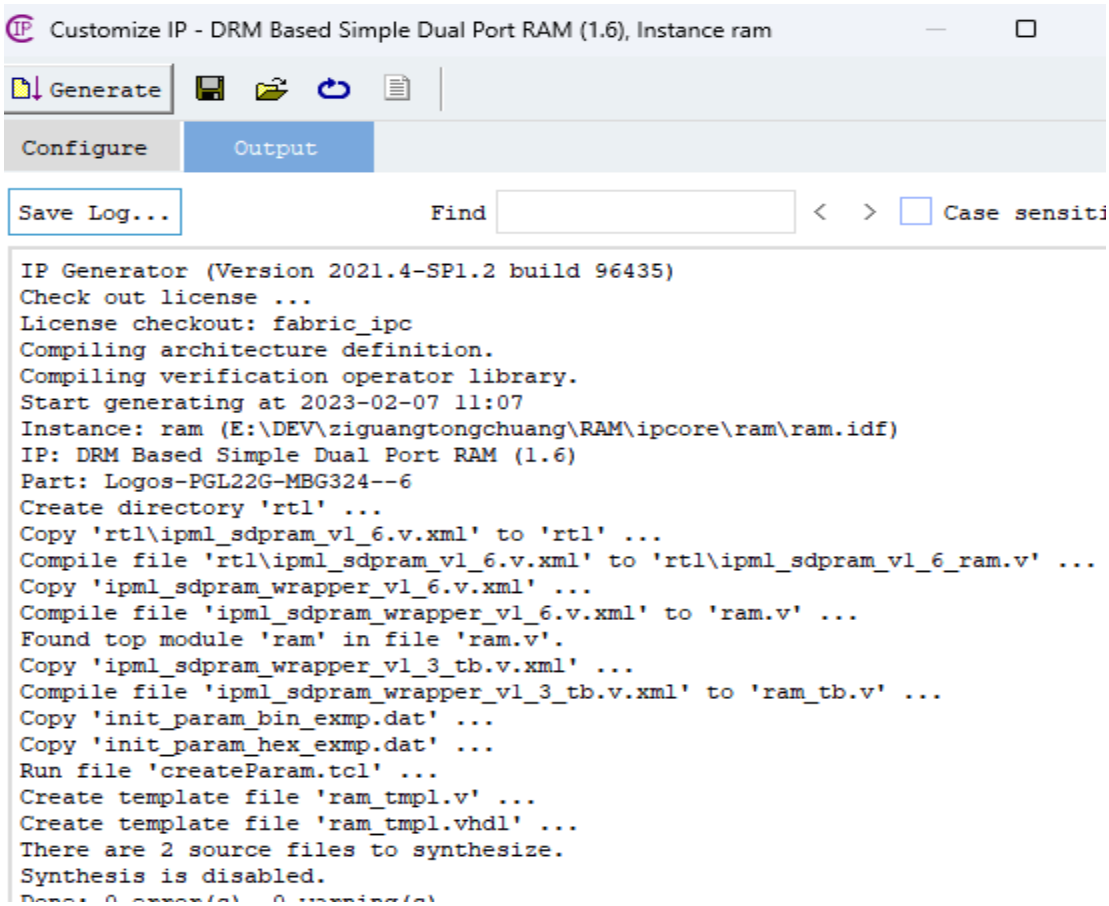

图 13-7　RAM IP 配置成功

IP 核配置成功后会自动弹出图 13-8IP 的例化模板文件，接下来我们需要例化这些自动生成的代码。

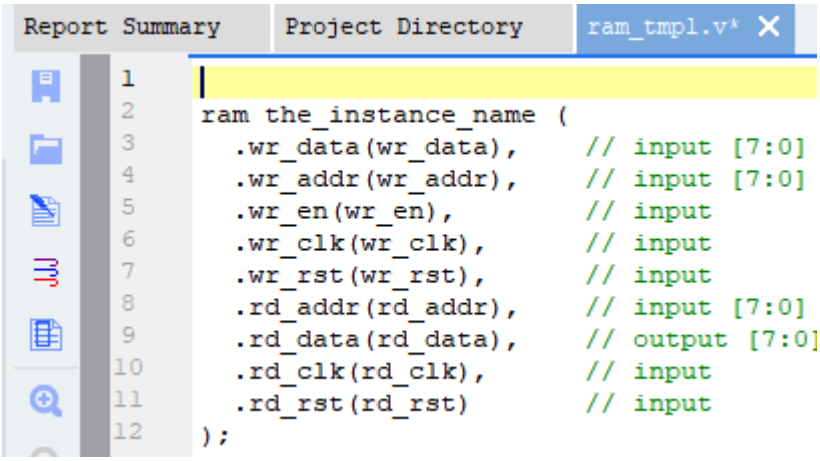

图 13-8 IP 的例化模板文件

将 IP 核配置过程中弹出的页面关闭，返回 Source 面板，如图 13-9 所示。

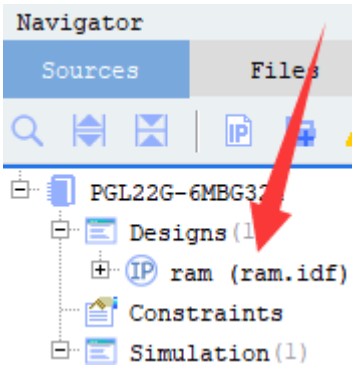

图 13-9 Source 面板

点击图 13-9 页面中箭头指向的 ram.idf 文件左侧加号位置，可以看到 ram.idf 核下有两个文件，点击图 1-10 中箭头所指的位置，打开 ram.v 文件。

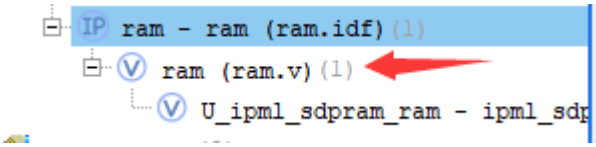

图 13-10 IP 核包含的文件

部分 ram.v 文件内容如图 13-11 所示，若要调用 RAM IP，我们可以选择将图 13-11 中的端口例化到需要调用 RAM IP 的模块中，或者是直接例化图 13-8 中内容。

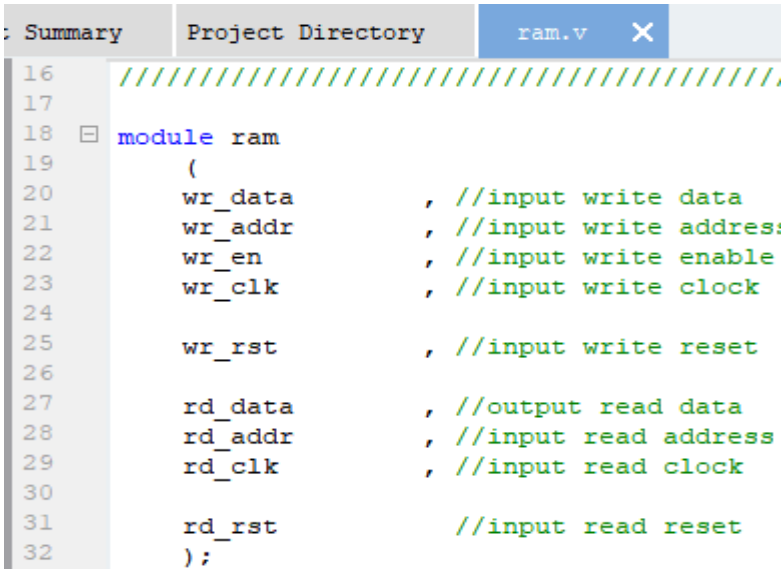

图 13-11 ram.v 文件部分内容

13.2.1代码设计

接下来我们创建一个 verilog 源文件，其名称为 ram_top，本设计的端口列表和源码比较简单，代码的主要作用就是例化 RAM IP，代码如下：

```
module ram_top(
  clk,
  reset_n,
  rd_data
   );

 input clk;
 input reset_n;
 output [7:0] rd_data;

 reg[7:0] wr_addr;
 reg [7:0] rd_addr;
 reg [8:0]counter;
 reg wr_en;
 reg rd_en;
 reg [7:0]w_data;

always@(posedge clk or negedge reset_n)
 if(!reset_n)
  wr_en<=0;
 else if(counter<=255)
 wr_en<=1;
 else
  wr_en<=0;

always@(posedge clk or negedge reset_n)
 if(!reset_n)
  counter<=0;
 else if(counter==9'd511)
  counter<=0;
 else
  counter<= counter+1'b1;

always@(posedge clk or negedge reset_n)
 if(!reset_n)
  wr_addr<=0;
 else if(wr_en==1)
  wr_addr<= wr_addr+1'b1;
 else
  wr_addr<=0;

always@(posedge clk or negedge reset_n)
 if(!reset_n)
  rd_en<=0;
else if((counter>254)&&(counter<=9'd511))
  rd_en<=1;
 else
```

```
  rd_en<=0;

always@(posedge clk or negedge reset_n)
 if(!reset_n)
  rd_addr<=0;
 else if(rd_en)
  rd_addr<= rd_addr+1'b1;
 else
  rd_addr<=0;

always@(posedge clk or negedge reset_n)
 if(!reset_n)
 w_data<=0;
else if(wr_en)
 w_data<= w_data+1'b1;
else
 w_data<=0;

  ram ram
    (
    .wr_data(w_data)       , //input write data
    .wr_addr(wr_addr)       , //input write address
    .wr_en  ( wr_en)       , //input write enable
    .wr_clk (clk)       , //input write clock
    .wr_rst (~reset_n)       , //input write reset
    .rd_data(rd_data)       , //output read data
    .rd_addr(rd_addr)       , //input read address
    .rd_clk (clk)       , //input read clock
    .rd_rst (~reset_n)         //input read reset
    );

endmodule
```

将上面的设计内容进行分析和综合直至没有错误以及警告后，按开发流程接下来进入激励创建及仿真测试环节。作为一个 FPGA 数字逻辑的完整的开发流程，仿真环节是必不可少的，这一要求请务必引起各位初学者的重视。

13.3激励创建及仿真测试

当 RAM IP 创建好之后，我们可以通过仿真方式来对该 IP 进行测试，可以通过实际写入一些数据再读取部分数据的方式来验证双端口 RAM 读写是否正常。这里针对该 IP，编写一个简单的 testbench 来进行仿真测试，testbench 代码如下所示。

```
`timescale 1ns/1ns
`define CLKA_PERIOD 20

module ram_top_tb();

 reg grs_n;

 GTP_GRS GRS_INST(
 .GRS_N (grs_n)
 );

initial begin
 grs_n = 1'b0;
 #50000;
 grs_n = 1'b1;
 end

 reg clk;
 reg reset_n;
 wire [7:0] rd_data;


initial clk = 1'b1;
always #(`CLKA_PERIOD/2) clk = ~clk;

initial begin
   reset_n=0;
   #21;
   reset_n=1;
   #2000000;
  end


  ram_top ram_top(
. clk(clk),
. reset_n(reset_n),
. rd_data(rd_data)
   );


endmodule
```

testbench 代码的是在地址从 0~255 上写入数据为从 0 递增至 255。然后再读取地址为 0~255 上的数据。

testbench 代码还例化了一个 GTP_GRS 模块，GTP_GRS 是一个全局复位模

块，在 FIFO 或者 RAM IP 中使用了这个全局复位，所以使用了 FIFO 或者 RAM IP 的文件的仿真代码里面就需要对 GTP_GRS 模块进行例化，不然联合仿真会报错。

如何对模块仿真这里我们不再赘述，仿真波形如图 13-12 所示

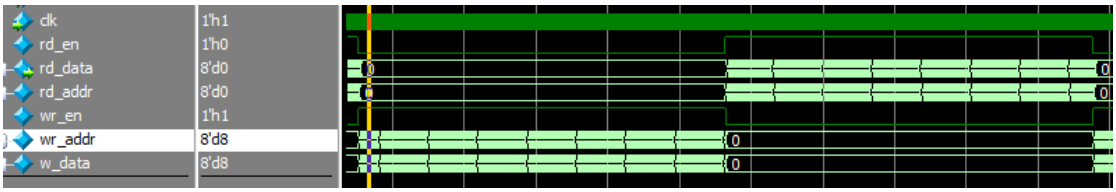

图 13-12 工程仿真波形全貌

对写数据部分放大后的波形，如图 13-13 所示。

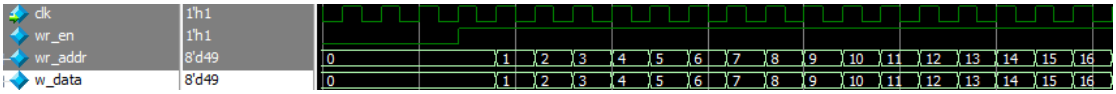

图 13-13 数据写入时放大的波形

在 wr_en 值为 1 即写入信号使能有效后，clk 上升沿来到后地址 0 写入数据 0，下一个上升沿来到后地址 1 写入数据 1，以此类推。可以看出数据写入正常。

放大读取部分数据波形，如图 13-14 所示

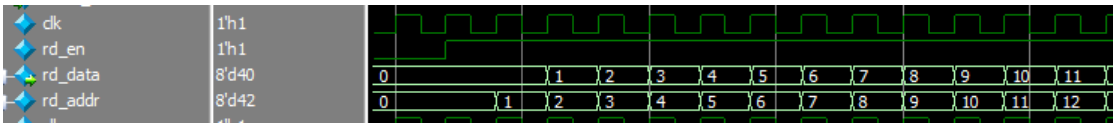

图 13-14 数据读出时放大的波形

由于 PDS 软件中的双端口 RAM 的读数据总是会延时一拍（也就是所谓的 Latency＝1），所以 clk 上升沿来到后地址 1 读出数据 0，下一个上升沿来到后地址 2 读出数据 1，以此类推。可以看出数据读出正常。

13.4板级调试与验证

本实验的板级验证环节，主要验证以下几个目标：

1. 能否正确将生成的 bit 文件下载到 PGL22G 开发板；
2. RAM IP 数据读写以及地址变化情况；

系统所需硬件：

1. PGL22G 开发板；
2. 电源电缆一根；

3. 硬件条件符合实验要求，具有完全开发功能的 PC 机一台；

13.4.1添加 I/O 约束

通常，一个设计中的 FPGA 不会是独立使用的，FPGA 一定会与其他外设、接口相连接，比如时钟，按键等。因此，FPGA 设计需要指定对应的 IO 引脚位置信息。

添加 IO 约束的方法非常简单：点击图 13-15 上方工具栏“Tools”栏中“User Constranit Editor(Timing and Logic)”后点击“Pre Synthesize USE”选项。

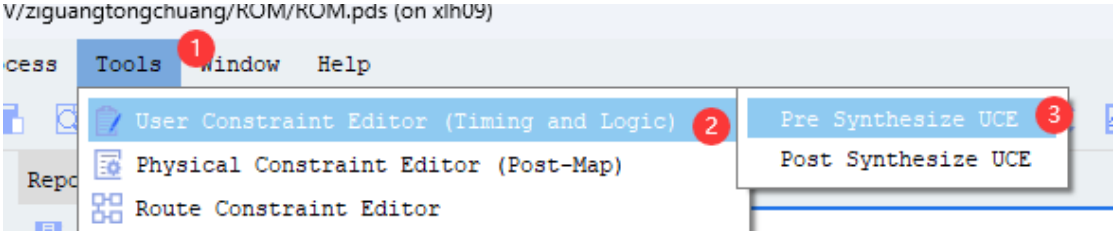

图 13-15“Tools”工具栏

点击图 13-15 中的“Pre Synthesize USE”选项后，弹出图 13-16 界面，首先点击“Pre Synthesize USE” 选项，然后点击“Device”选项，最后点击“I/O”选项，进入管脚分配页面。

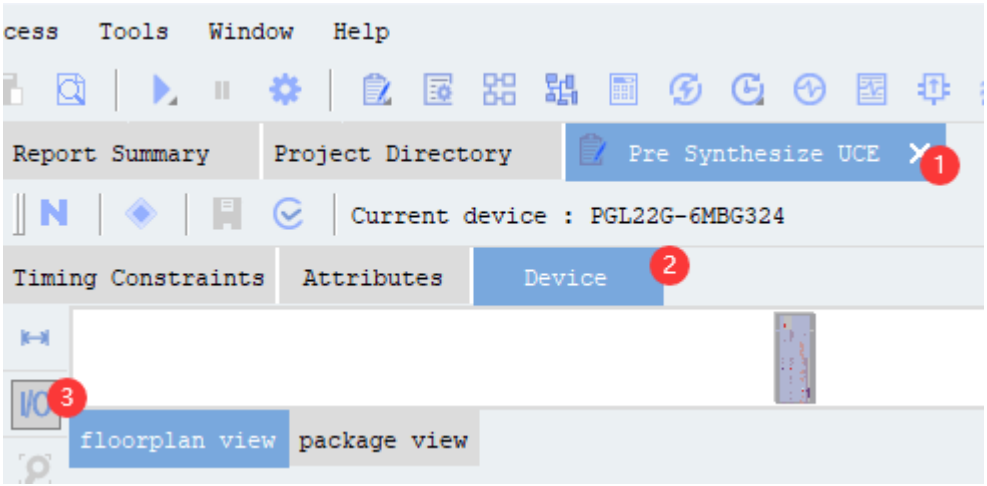

图 13-16 管脚分配入口

打开管脚约束窗口如图 13-17，其中 LOC ，VCCIO,IOSTANDARD 是我们要约束的内容。这里，参考我们提供的管脚约束表即可完成管脚绑定。

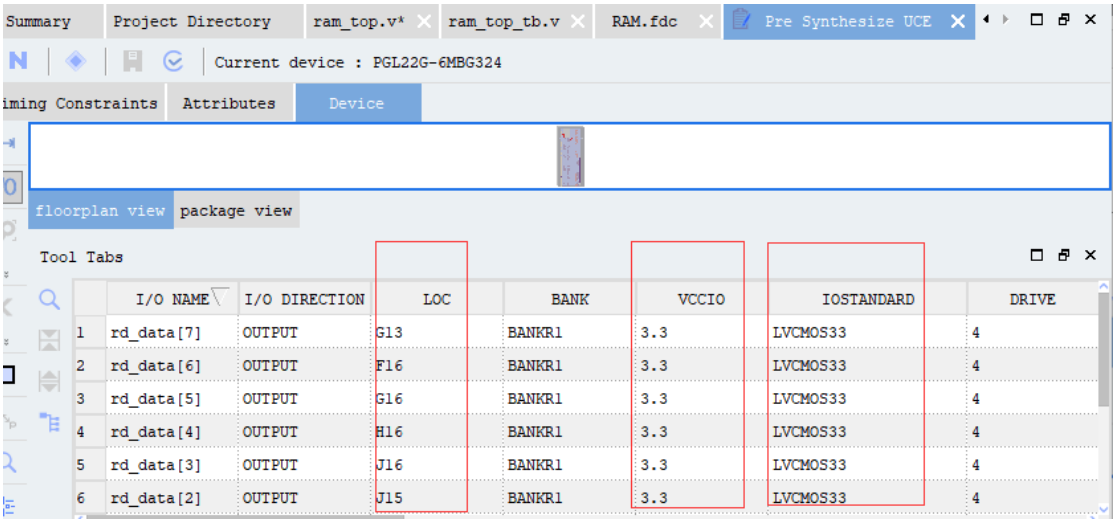

图 13-17 管脚分配页面

这样管脚约束添加完成了。但是此时约束内容保存在内存中，还没有写入文件，点击工具栏保存按钮(图 13-18 中箭头指向的位置)，或者直接 Ctrl+S。

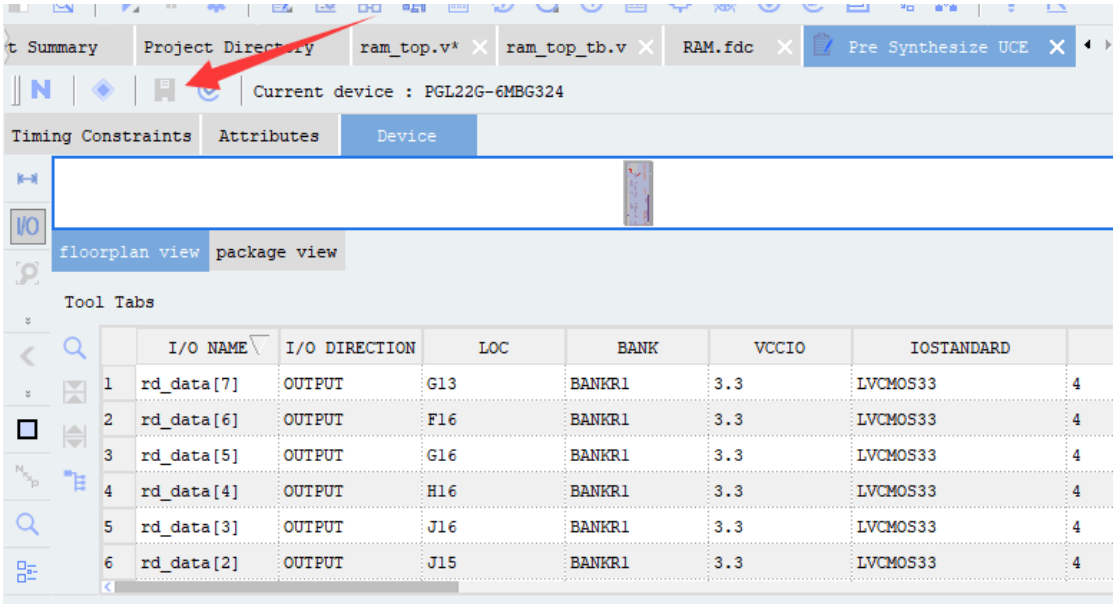

图 13-18 保存文件

将约束文件命名为 RAM 后就可以在 Source 窗口的 Constraints 下可找到刚保存的 ROM.fdc 文件。双击可以打开约束文件。

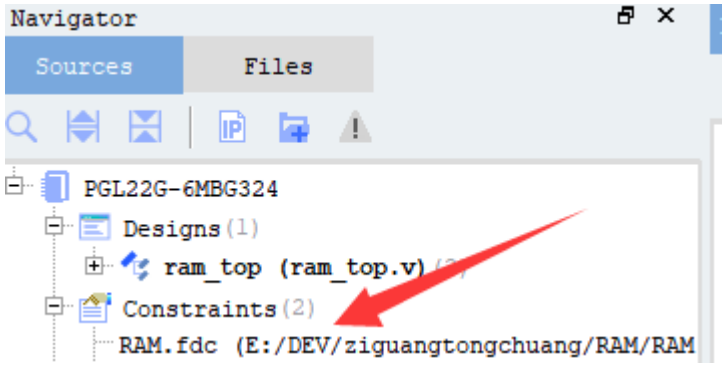

图 13-19 Source 面板

13.4.2下载验证

新建 DebugCore，将 rd_data，rd_addr，wr_addr，w_data，wr_en 以及 rd_en 这些信号添加至观察列表中，新建 DebugCore 核的方法这里不再赘述。

编译工程并生成比特流.sbit 文件后，此时将下载器一端连接电脑，另一端与开发板上的 JTAG 下载口连接，连接电源线，并打开开发板的电源开关。

点击 PDS 工具栏的下载按钮，在弹出的 Fabric Configuration 界面中双击“Scan Device”，我们将生成好的 sbit 流文件下载到开发板中去。

在线调试配置界面将 wr_en 信号设置为高电平触发，然后点击图 13-20 中的触发按钮。

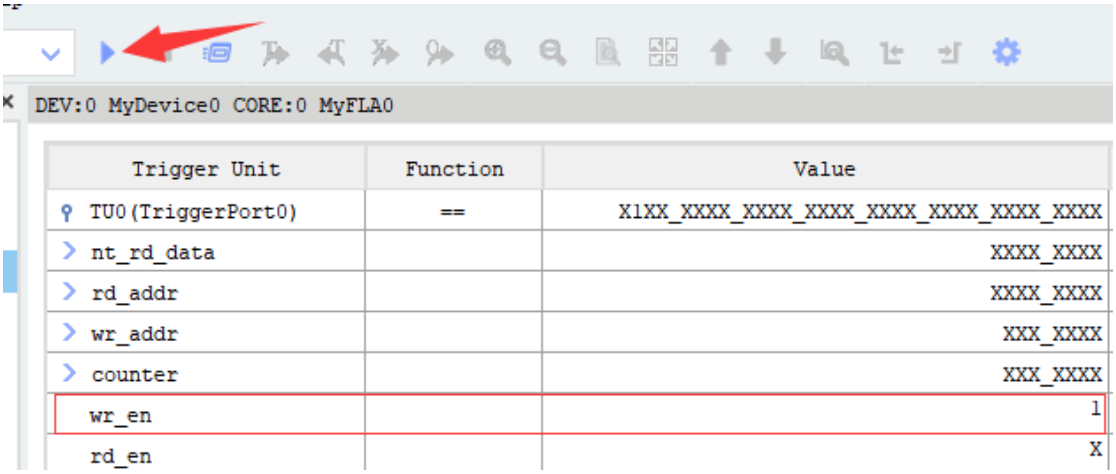

图 13-20 设置触发信号

点击图 13-20 箭头指向的触发按钮后，等待触发时按下开发板的复位按键，可以看到如图 13-21 所示一张整体的在线调试运行图。

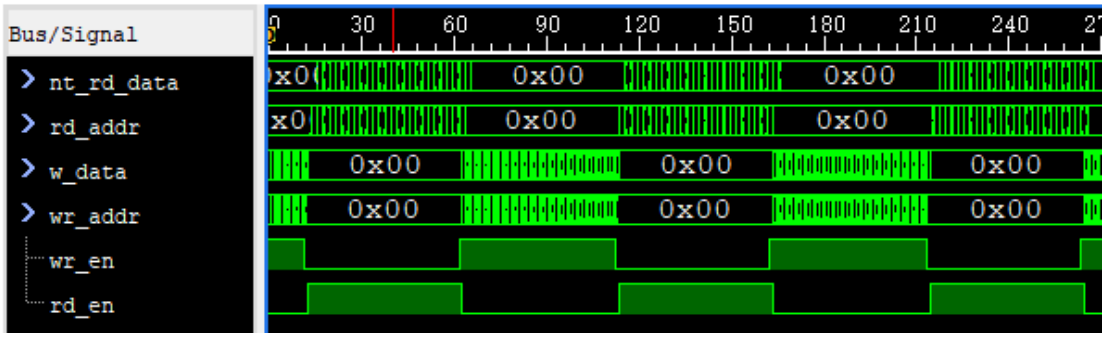

图 13-21 工程仿真波形全貌

对写数据部分放大后的波形，如图 13-22 所示。

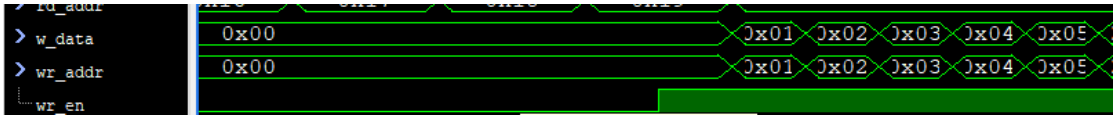

图 13-22 数据写入放大的波形

在 wr_en 值为 1 即写入信号使能有效后，地址和数据都是从 0 开始累加，也就说当 ram 地址为 0 时，写入的数据也是 0；当 ram 地址为 1 时，写入的数据也是 1。我们可以发现，上图中的数据变化和在 PDS 仿真时的波形也是一致的。放大读取部分数据波形，如图 13-23 所示。

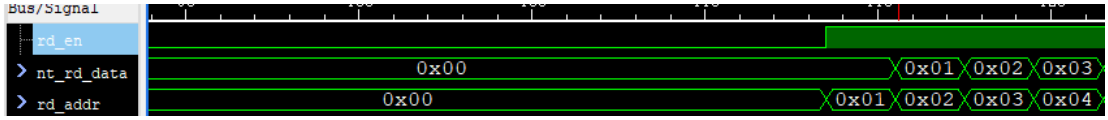

图 13-23 数据读出放大的波形

在 rd_en 值为 1 即读使能有效后，rd_addr 从 0 开始增加，也就是说从 ram 的地址 0 开始读数据；ram 中读出的数据 即 rd_data 在延时 1 个时钟周期之后，开始输出数据，输出的数据为 0，而且读出的数据写入的数据是相等的。

我们可以发现，上图中的数据变化同样和 PDS 仿真的波形是一致的。本次实验的 IP 核之简单双端口 RAM 读写实验验证成功。

13.5常见问题说明

1. 对于一个完整的 FPGA 工程开发流程，仿真是一个重要而必不可少的步骤；
2. RAM IP 中使用了 GTP_GRS 是一个全局复位模块，在对涉及到 RAM IP 代码进行仿真时，要在仿真代码里面对 GTP_GRS 模块进行例化；

13.6总结

本章简单介绍了 PDS 软件 中 RAM 的分类、区别以及相关概念。通过调用 IP 核的方式实现了一个双端口的 RAM。本章属于理论学习与上板实验相结合，建议读者能够跟随本实验内容，完整的进行整个实验。