20.RAM IP 核使用实验例程

20.1实验简介

RAM 即随机存取存储器（Random Access Memory）。它可以在运行过程中把数据写进任意地址，也可以把数据从任意地址中读出。其作用可以拿来做数据缓存，也可以跨时钟，也可以存放算法中间的运算结果等。**与 FIFO 不同的是，RAM 是有地址的。**

注意：PDS 的 IP 配置工具中提供两种不同的 RAM，一种是 Distributed RAM(分布式 RAM)另一种是 DRM Based RAM，分布式 RAM 用的是 LUT(查找表)资源去构成的 RAM，这种 RAM 会消耗大量 LUT 资源，因此通常在一些比较小的存储才会用到这种 RAM，以节省 DRM 资源。而 DRM Based RAM 是利用片内的 DRM 资源去构成的 RAM，不占用逻辑资源，而且速度快，通常设计中均使用 DRM Based RAM。

RAM 分为**单端口 RAM,伪双端口 RAM，真双端口 RAM。**他们的区别在于单端口只有一个端口进行读和写的操作；双端口具有两个端口进行读和写操作，其中伪双端口的一个端口只能用来读，另一个端口只能用来写；真双端口则是两个端口可以独立进行读和写。

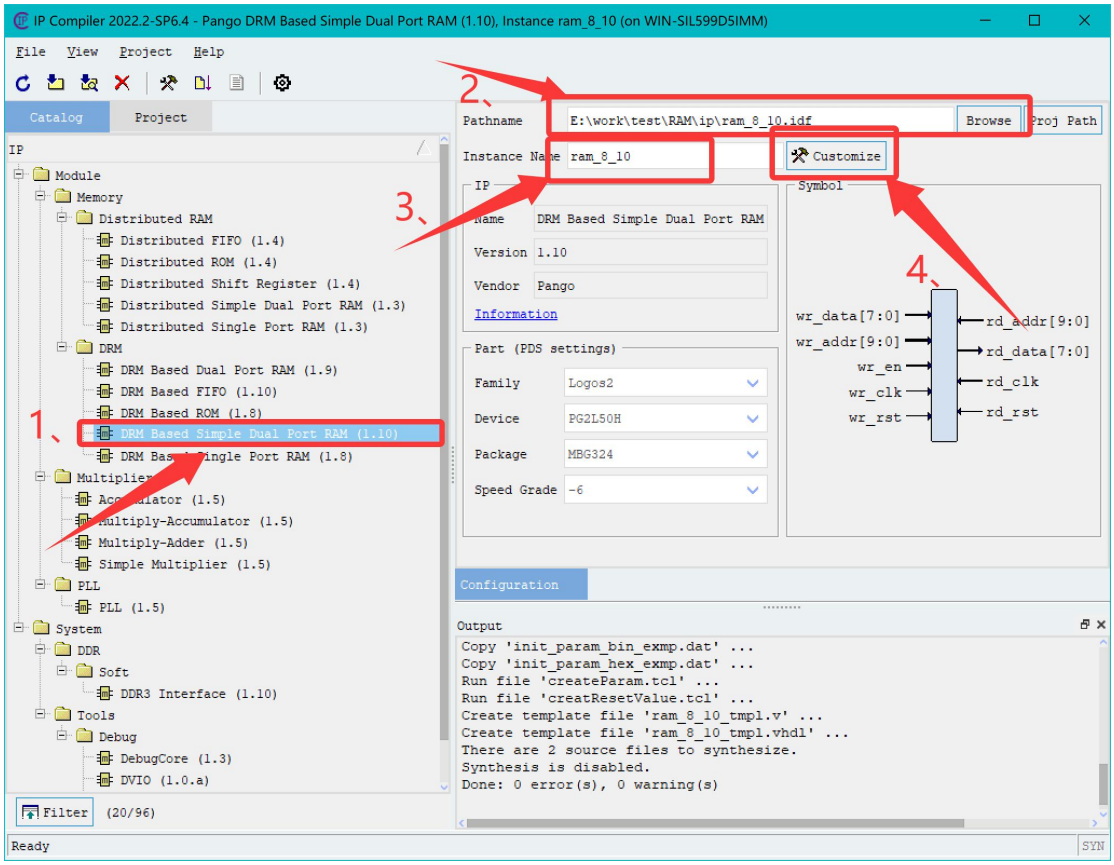
注意：使用真双端口时要避免对相同地址的同时读和写。

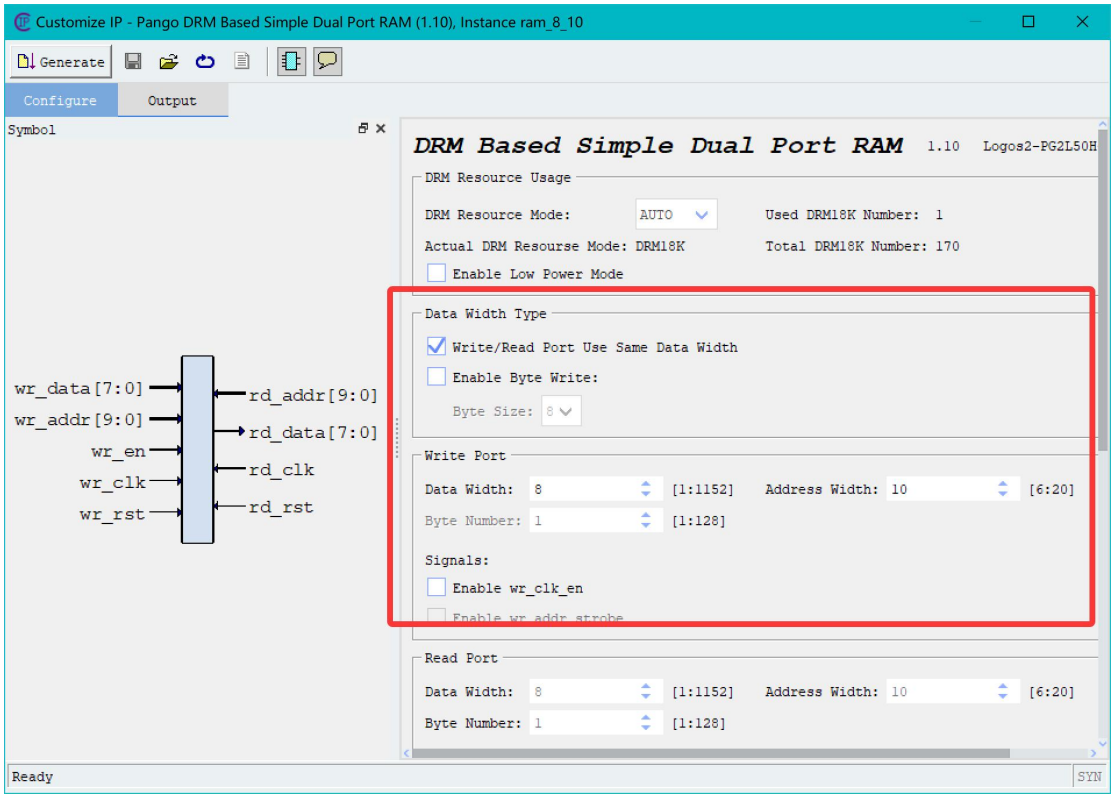
20.2实验目的

掌握 RAM 对数据的存储和读取并灵活根据需求更改

IP 配置

进入 IP 选择界面后
选择 Simple Dual Pore RAM，填写好路径以及 IP 名称后进入参数页、
照下图配置，数据位宽为 8bit，地址位宽为 10bit。
点击 Generate。





例化我们的 IP 核

激励我们在仿真文件中编写

```
module RAM_test
```

```
(
    input   wire                i_clk           ,
    input   wire                i_rst           ,
    input   wire                i_rw_en         ,
    input   wire    [9 :0]      i_wr_addr       ,
    input   wire    [9 :0]      i_rd_addr       ,
    input   wire    [7 :0]      i_wr_data       ,

    output  wire    [7 :0]      o_rd_data
);
ram_8_10 ram_8_10_u0
(
    .wr_data                    (i_wr_data      ),
    .wr_addr                    (i_wr_addr      ),
    .wr_en                      (i_rw_en        ),
    .wr_clk                     (i_clk          ),
    .wr_rst                     (i_rst          ),
    .rd_addr                    (i_rd_addr      ),
    .rd_data                    (o_rd_data      ),
    .rd_clk                     (i_clk          ),
    .rd_rst                     (i_rst          )
);
```

主要代码如下图，和之前的实验类似，这里不再过多解释。

```
    case(r_state)
        2'd0:begin
                r_rw_en <= 'd1;
                r_state <= 'd1;
            end
        2'd1:begin
                if(r_wr_addr == 'd127)
                begin
                    r_rw_en    <= 'd0;
                    r_state    <= 'd2;
                    r_wr_data  <= 'd0;
                    r_wr_addr  <= 'd0;
                    r_rd_addr  <= 'd0;
                end
                else
                begin
                    r_state    <= 'd1;
                    r_wr_data  <= r_wr_data+'b1;
                    r_wr_addr  <= r_wr_addr+'b1;
                end
            end
```

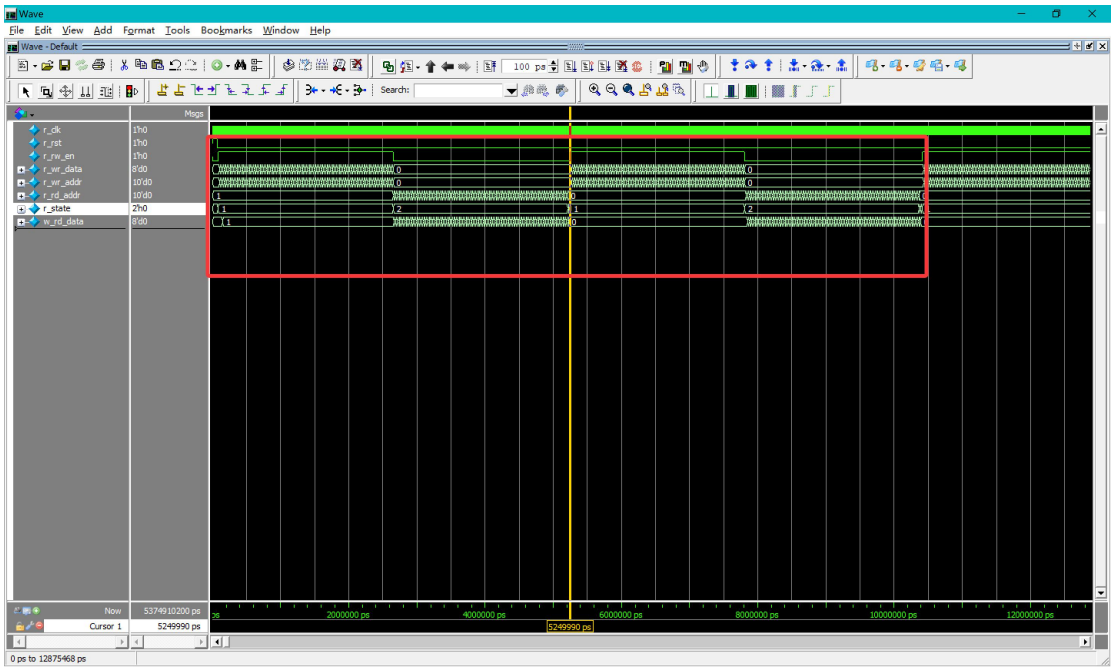
```
        2'd2:begin
                if(r_rd_addr == 'd127)
                begin
                    r_state    <= 'd3;
                    r_rd_addr  <= 'd0;
                end
                else
                begin
                    r_state    <= 'd2;
                    r_rd_addr  <= r_rd_addr+'b1;
                end
            end
        2'd3:begin
                r_state <= 'd0;
            end
        default:    r_state <= 'd0;
        endcase
    GTP_GRS GRS_INST
    (
          .GRS_N(1'b1)
    );
```

端口连接

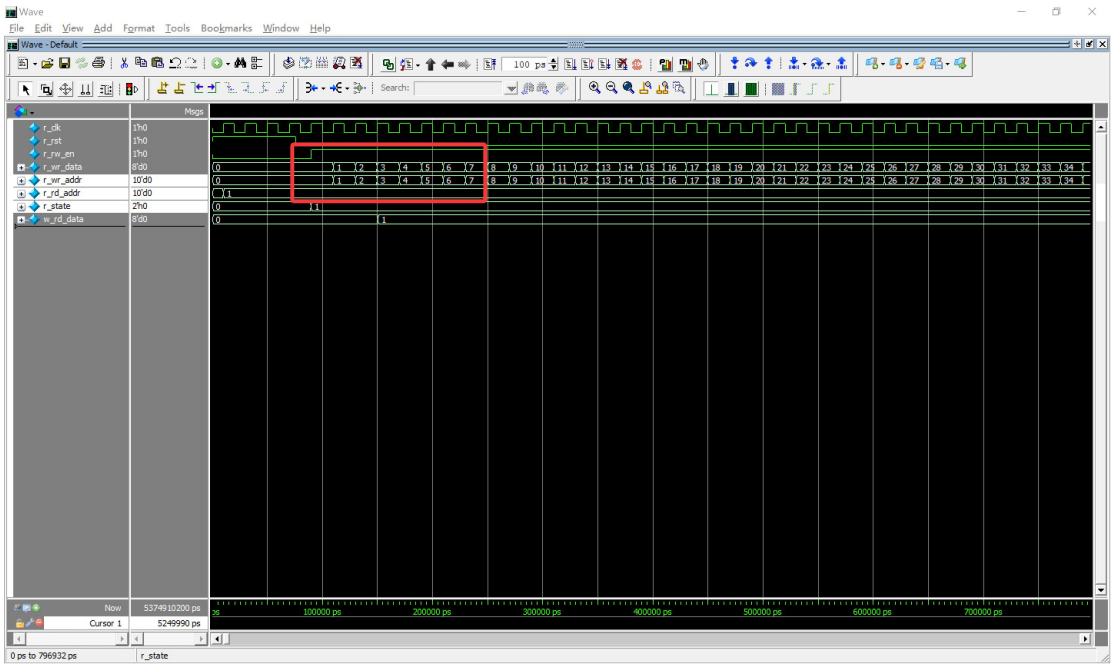
```
    ROM_test ROM_test_u0
    (
          .i_clk                (r_clk            ),
          .i_rst                (r_rst            ),
          .i_addr               (r_addr           ),
          .o_data               (w_data           )
    );
```

20.3仿真波形

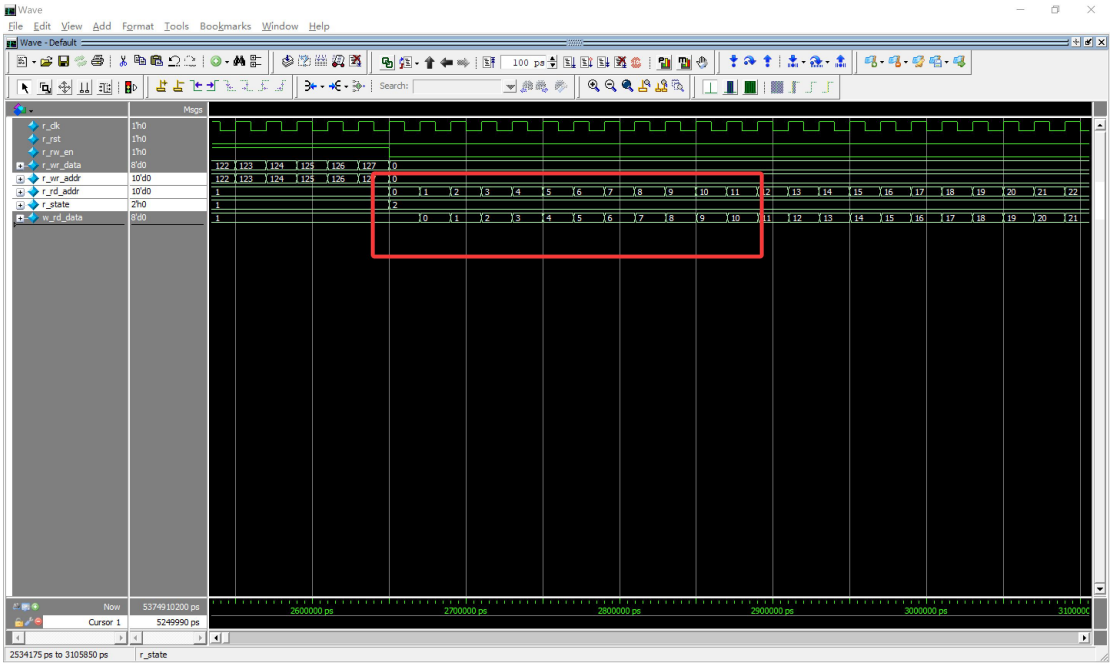
从波形中看到我们的 RAM 是分别进行读和写的操作，从 0 地址开始写递增数据；读端口从 0 地址读出我们写入的数据。详细读写时序请参考官方 IP 手册。


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
r_clk
r_rst
r_rw_en
r_wr_data
r_wr_addr
r_rd_addr
r_state
w_rd_data
Now
Cursor 1
0 ps to 12875468 ps

仿真波形


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
r_clk
r_rst
r_rw_en
r_wr_data
r_wr_addr
r_rd_addr
r_state
w_rd_data
Now
Cursor 1
0 ps to 796932 ps
r_state

写端口波形



读端口波形

至此，我们 RAM 的实验已经成功完成！！！