

# FPGA 片内 RAM 读写例程

黑金动力社区 2020-03-13

# 1 实验简介

本实验将为大家介绍如何使用 FPGA 内部的 RAM 以及程序对该 RAM 的数据读写操作。

# 2 实验原理

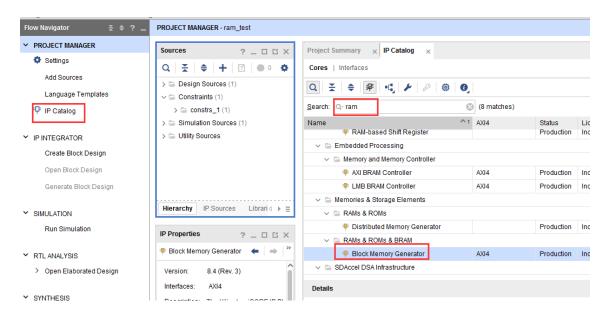
Xilinx 在 VIVADO 里为我们已经提供了 RAM 的 IP 核,我们只需通过 IP 核例化一个 RAM,根据 RAM 的读写时序来写入和读取 RAM 中存储的数据。实验中会通过 VIVADO 集成的在线逻辑分析仪 ila,我们可以观察 RAM 的读写时序和从 RAM 中读取的数据。

# 3 程序设计

#### 3.1 RAM IP 的添加和配置

在添加 RAM IP 之前先新建一个 ram test 的工程, 然后在工程中添加 RAM IP , 方法如下:

1. 点击下图中 IP Catalog,在右侧弹出的界面中搜索 ram,找到 Block Memory Generator,双击打开。

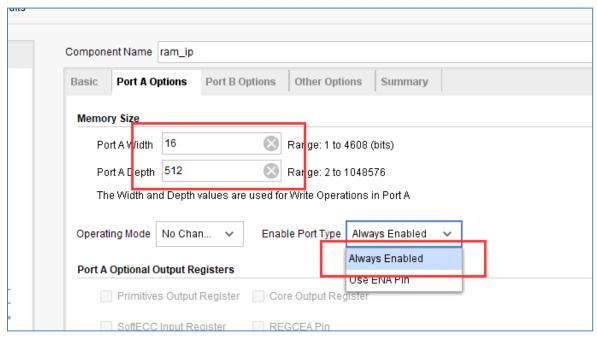




2. 将 Component Name 改为 ram\_ip,在 Basic 栏目下,将 Memory Type 改为 Simple Dual Prot RAM, 一般来讲"Simple Dual Port RAM"是最常用的,因为它是两个端口,输入和输出信号独立。

		Component Na	me ram_ip						
		Basic Port	t A Options Port B Opt	ions	Other C	ptions	Summary		
		Interface Typ	oe Native	~	_ G	enerate	address inte	rface with 3	
		Memory Type	e Simple Dual Port RAM	۱ ٧		ommon	Clock		
			Single Port RAM						
		ECC Options	Simple Dual Port RAM						
		ECC Typ	pe True Dual Port RAM				~		
·			Single Port ROM	Single Port ROM		Injection ~			
		Erro	Dual Port ROM		Injection	II			
sbiterr		Write Enable	е						
dbiterr	-	D) P) to	Data Write Enoble						
rdaddreco[3:0]	-	Byte Write Enable							
ısta_busy	-	Byte Siz	Byte Size (bits) 9						
rstb_busy s_axi_sbiterr		2),10 0.12	- ()						
s_axi_sbiteii									

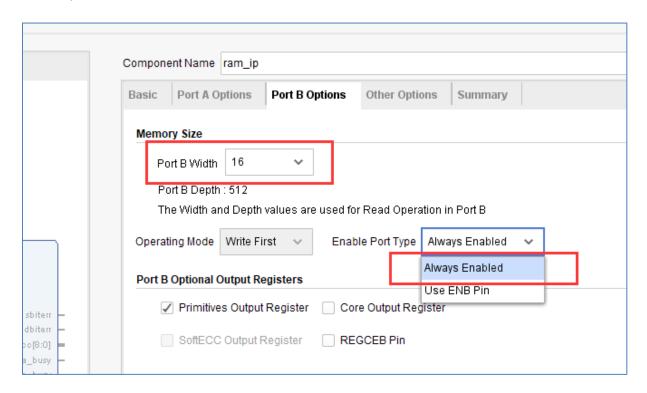
3. 切换到 Port A Options 栏目下,将 RAM 位宽 Port A Width 改为 16,将 RAM 深度 Port A Depth 改为 512,使能管脚 Enable Port Type 改为 Always Enable。



黑金动力社区 2/6



4.切换到 Port B Options 栏目下,将 RAM 位宽 Port B Width 改为 16,使能管脚 Enable Port Type 改为 Always Enable。



- 5. 在 Other Options 栏目中,这里不像 ROM 那样需要初始化 RAM 的数据,我们可以在程序中写入, 所以配置默认即可,直接点击 OK。
- 6. 点击"Generate"生成 RAM IP。

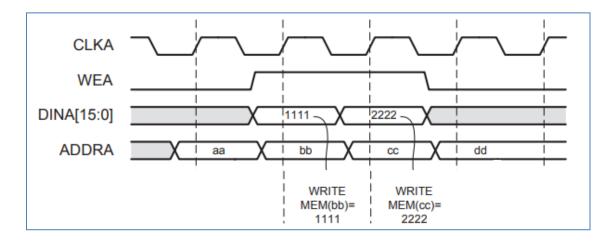
#### 3.2 RAM 的端口定义和时序

Simple Dual Port RAM 模块端口的说明如下:

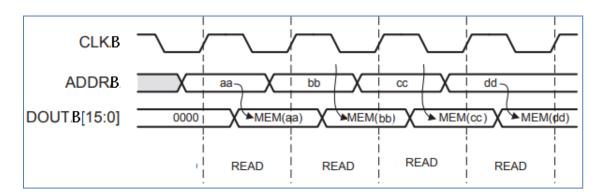
信号名称	方向	说明
clka	in	端口A时钟输入
wea	in	端口A使能
addra	in	端口 A 地址输入
dina	in	端口A数据输入
clkb	in	端口B时钟输入
addrb	in	端口B地址输入
doutb	out	端口B数据输输出

RAM 的数据写入和读出都是按时钟的上升沿操作的,端口 A 数据写入的时候需要置高 wea 信号,下图为输入写入到 RAM 的时序图。





### RAM 写时序



RAM 读时序

### 3.3 RAM 测试程序编写

添加一个首先对 RAM IP 的实例化, RAM IP 的实例化及程序设计如下:

```
calling xilinx ram IP
ram_ip ram_ip_inst
                    // input clka
       (clk
                ),
. clka
                    // input [0 : 0] wea
       (wea
.wea
       (w_addr
                ),
                    // input [8 : 0] addra
. addra
                  // input [15 : 0] dina
. dina
       (w_data
                ),
                ),
                   // input clkb
. clkb
       (clk
. addrb
                ),
                    // input [8 : 0] addrb
      (r_addr
. doutb
       (r_data
                    // output [15 : 0] doutb
```

黑金动力社区 4/6



程序上电后会对写入 512 个 0~511 的数据到 RAM 里。

```
) always@(posedge clk or negedge rst_n)
) begin
   if(rst_n=1'b0) begin
        wea <= 1' b0;
       w_addr <= 9' d0;
        w data <= 16' d0;
   end
   else begin
       if(w_addr=511) begin //write ram end
          wea <= 1' b0;
       end
       else begin
         wea<=1'b1;
                              //write ram enable
            w_addr <= w_addr + 1' b1;
            w_data <= w_data + 1' b1;
        end
    end
) end
```

再不断的读取 RAM 的数据,为了能实时看到 RAM 中读取的数据值,我们这里添加了 ila 工具来观察 RAM 的数据信号。关于如何生成 ila 大家请参考"I2C 接口 EEPROM 实验.pdf"教程。

```
56 ila_0 ila_0_inst (
57 .clk(clk),
58 .probe0(r_data),
59 .probe1(r_addr)
60 );
```

### 4 实验现象

下载 bit 文件到 FPGA。接下来我们通过 ila 来观察一下从 RAM 中读出的数据是否为我们初始 化文件 coe 的数据。





在 Waveform 的窗口我们可以看到  $r_addr$  在不断的从 0 累加到 1ff, 随着  $r_addr$  的变化,  $r_addr$  也在变化,  $r_addr$  的数据正是我们写入到 RAM 中的 512 个数据,这里需要注意, $r_addr$  出现新地址时, $r_addr$  对应的数据要延时两个时钟周期才会出现,数据比地址出现晚两个时钟周期。

