杭州電子科技大学

计算机组成原理（甲）

实 验 报 告

学　　院	网络空间安全学院
专　　业	网络工程
班　　级	19272401
学　　号	19061440
学生姓名	F001
教师姓名	袁理峰
完成日期	2020.12.11
成　　绩	

实验五 存储器设计实验

一、 实验目的

（1） 掌握灵活运用 Verilog HDL 进行各种描述与建模的技巧和方法。

（2） 学习在 ISE 中设计生成 Memory IP 核的方法。

（3） 学习存储器的结构及读写原理，掌握存储器的设计方法。

二、 实验原理

存储器用于存储数据和指令。MIPS32 处理器的地址总线为 32 位，数据总线为 32 位，但是存储器按字节编址，因此存储器容量位 4G，地址范围 0x00000000~0xFFFFFFFF。但 MIPS 处理器绝大部分指令对存储器的访问尺寸位字，即访问地址必须按照 4 字节的边界严格对齐。至于 MIPS 的一个字的字节大小端问题，选择小端格式。高字节在高地址，低字节在低地址。

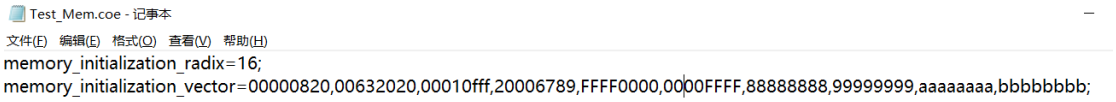
本实验要求为 MIPS 处理器设计一个 256*8 位的物理存储器，具有读/写功能，按字节编址，按字访问，即 64*32 位。输入端口为 Mem_Read、Mem_Addr、M_W_Data、Mem_Write，输出端口为 M_R_Data。分别为读控制信号，存储器的访问地址，存储器的读出数据，写控制信号和存储器的写入数据。

三、 实验环境

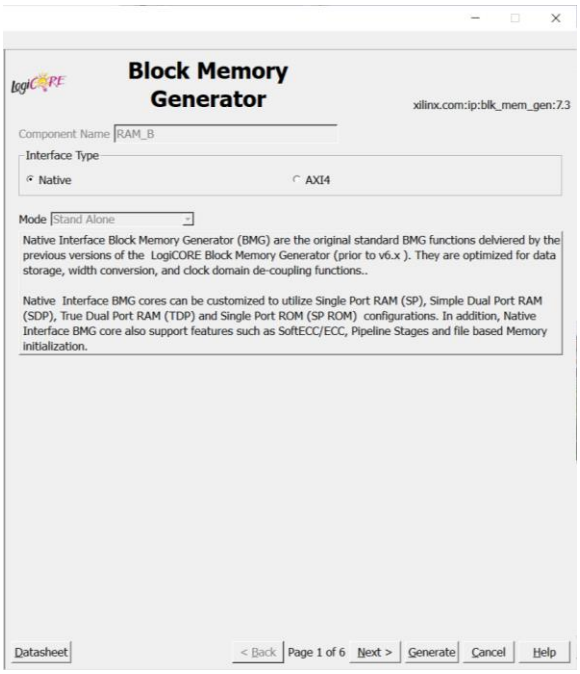
所用电脑的软硬件配置：自己的笔记本电脑、Windows10 操作系统
实验所用的软件：ISE design suite

四、 主要操作步骤及实验结果记录（不能光截图，要有相应的文字说明）

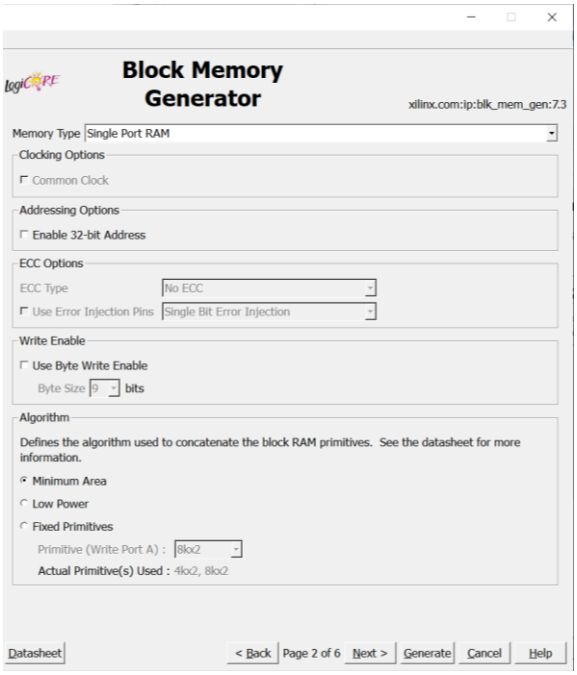
（对实验过程中的主要操作步骤进行描述，并随时记录实验过程中观察到的结果，必要时可辅助截图）
任务一：生成 FPGA 内置的存储器 IP 核
(1)建立并编辑一个关联文件。在工程目录下生成一个扩展名为*.coe 的纯文本文件。该文件是用于 Memor IP 核的初始化初始化文件，文件包括两行，具体内容如下：



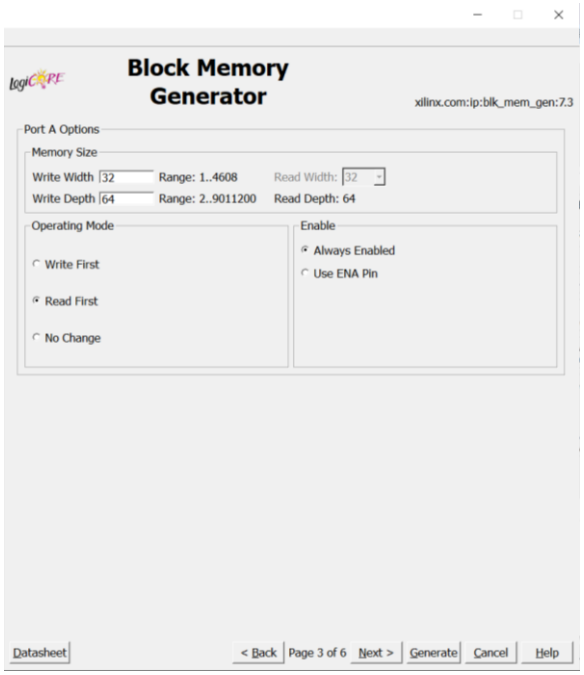
第一行用来表示数据是几进制的存在，第二行定义初始化向量。
(2)新建一个 Memory IP 核。在工程管理区的任意位置邮寄，New Source->New Source Wizard->IP(CORE Generator & Architecture Wizard)路径即为默认路径。之后需要选择 IP 核，找到 Memories & Storage Elements->RAMs & ROMs->Block Memory Generator.带有时钟同步，但是相对麻烦。之后进入 IP 核的设置
(3)Memory IP 核参数设置。共有 6 页，依次按照如下图配置进行。


Block Memory Generator
xilinx.com:ip:blk_mem_gen:7.3
Component Name RAM_B
Interface Type
Native
AXI4
Mode Stand Alone
Native Interface Block Memory Generator (BMG) are the original standard BMG functions delviered by the previous versions of the LogiCORE Block Memory Generator (prior to v6.x). They are optimized for data storage, width conversion, and clock domain de-coupling functions..
Native Interface BMG cores can be customized to utilize Single Port RAM (SP), Simple Dual Port RAM (SDP), True Dual Port RAM (TDP) and Single Port ROM (SP ROM) configurations. In addition, Native Interface BMG core also support features such as SoftECC/ECC, Pipeline Stages and file based Memory initialization.
Datasheet
< Back
Page 1 of 6
Next >
Generate
Cancel
Help

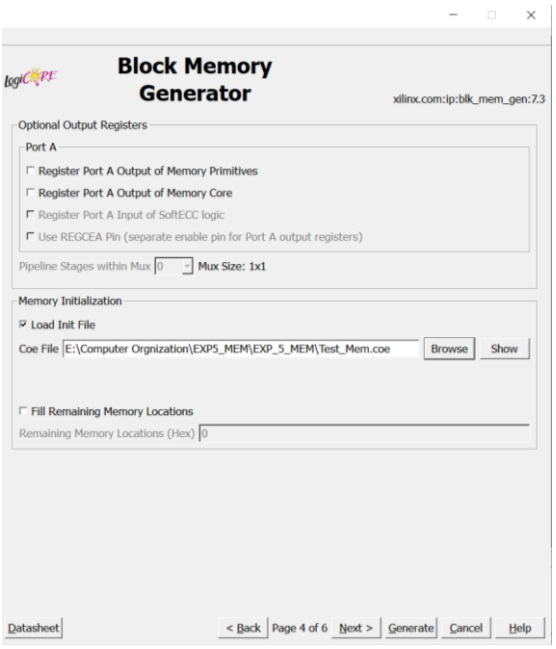
1


Block Memory Generator
xilinx.com:ip:blk_mem_gen:7.3
Memory Type Single Port RAM
Clocking Options
Common Clock
Addressing Options
Enable 32-bit Address
ECC Options
ECC Type No ECC
Use Error Injection Pins Single Bit Error Injection
Write Enable
Use Byte Write Enable
Byte Size 9 bits
Algorithm
Defines the algorithm used to concatenate the block RAM primitives. See the datasheet for more information.
Minimum Area
Low Power
Fixed Primitives
Primitive (Write Port A) : 8kx2
Actual Primitive(s) Used : 4kx2, 8kx2
Datasheet
< Back
Page 2 of 6
Next >
Generate
Cancel
Help

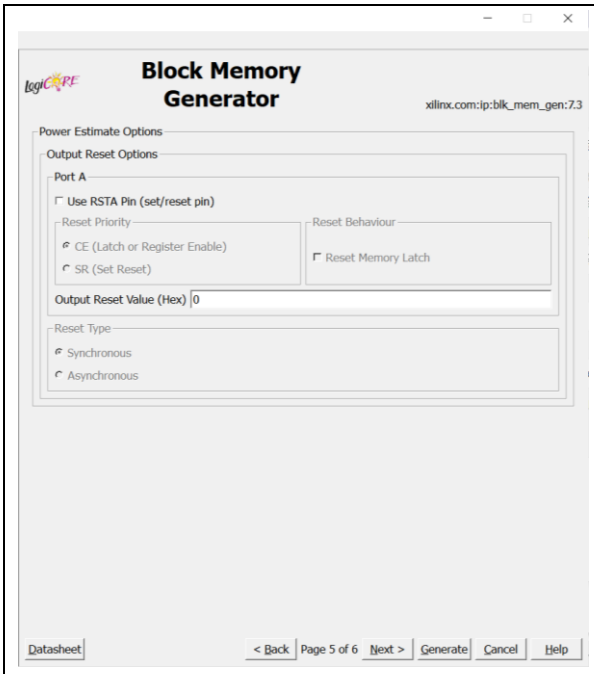
2


Block Memory Generator
xilinx.com:ip:blk_mem_gen:7.3
Port A Options
Memory Size
Write Width 32 Range: 1..4608 Read Width: 32
Write Depth 64 Range: 2..9011200 Read Depth: 64
Operating Mode
Write First
Read First
No Change
Enable
Always Enabled
Use ENA Pin
Datasheet
< Back
Page 3 of 6
Next >
Generate
Cancel
Help

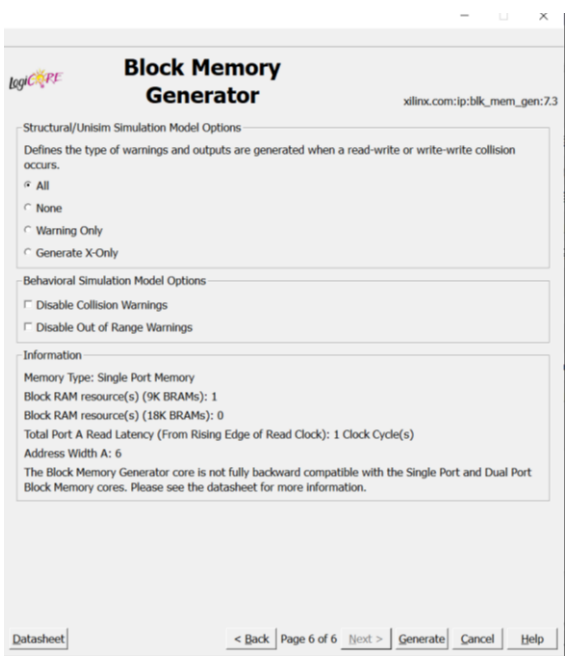
3


Block Memory Generator
xilinx.com:ip:blk_mem_gen:7.3
Optional Output Registers
Port A
Register Port A Output of Memory Primitives
Register Port A Output of Memory Core
Register Port A Input of SoftECC logic
Use REGCEA Pin (separate enable pin for Port A output registers)
Pipeline Stages within Mux 0 Mux Size: 1x1
Memory Initialization
Load Init File
Coe File E:\Computer Orgnization\EXP5_MEM\EXP_5_MEM\Test_Mem.coe Browse Show
Fill Remaining Memory Locations
Remaining Memory Locations (Hex) 0
Datasheet
< Back
Page 4 of 6
Next >
Generate
Cancel
Help

4



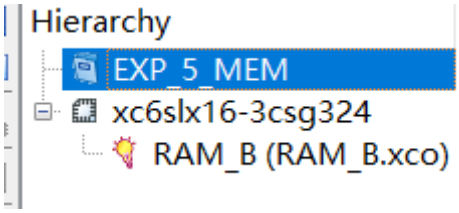
5



6

(4)调用存储器模块。双击过程管理区的 View HDL Instruction Template，在右侧的代码区会给出调用模板，将其复制到顶层模块中，修改模块实例名称和连接端口参数，就可以像一般模块引用了。

任务二：在 Xilinx ISE 中创建工程，编辑程序源代码，然后编译，综合；若编译出错，则需要修改程序代码，直至正确。



生成完毕的 RAM 模块。

```
48
49  //----------- Begin Cut here for INSTANTIATION Te
50  RAM_B your_instance_name (
51    .clka(clka), // input clka
52    .wea(wea), // input [0 : 0] wea
53    .addra(addra), // input [5 : 0] addra
54    .dina(dina), // input [31 : 0] dina
55    .douta(douta) // output [31 : 0] douta
56  );
57  // INST_TAG_END ------ End INSTANTIATION Templat
```

上图为 RAM 所显示的模板情况。此处为了简便起见，直接忽略了顶层代码，在 test 文件中直接进行相关调试，并且成功，因此认为我们的 RAM 模块是正常的。

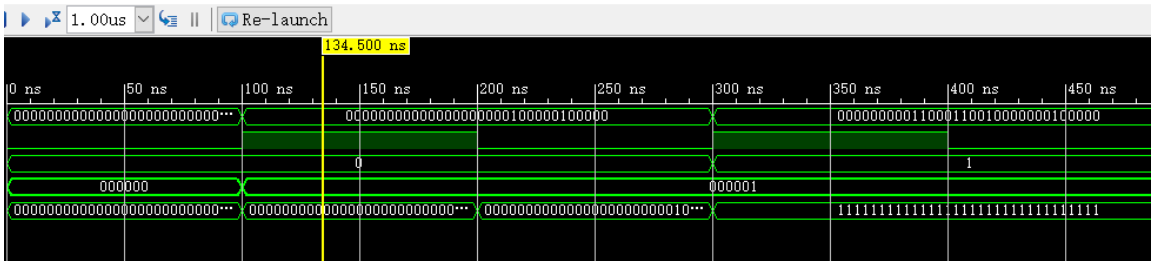
```
initial begin
   // Initialize Inputs
   clka = 0;
   wea = 0;
   addra = 0;
   dina = 0;

   // Wait 100 ns for global reset to fin
   #100;

   clka = 1;
   wea = 0;
   addra = 6'b000001;
   dina = 32'h0000_0003;

   #100;
   clka = 0;
   wea = 0;
   addra = 6'b000001;
   dina = 32'h0000_0111;

   #100;
   clka = 1;
   wea = 1;
   addra = 6'b000001;
   dina = 32'hFFFF_FFFF;

   #100;
   clka = 0;
   wea = 1;
   addra = 6'b000001;
   dina = 32'hFFFF_FFFF;
   // Add stimulus here

end
```

任务三：编写激励代码，观察仿真波形，若逻辑验证有误，则修改代码，重新编译、仿真，直至正确。

左图为所描写的激励代码。

下图为展示的波形。

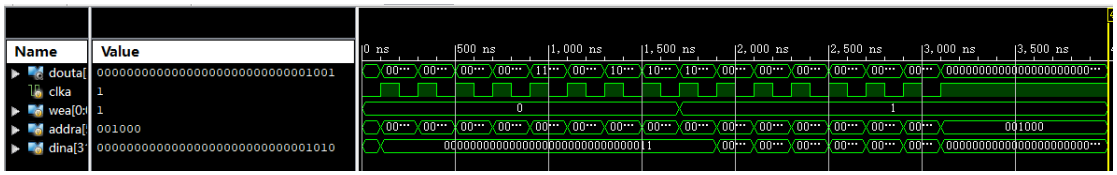
对比可得知，波形正确，相关代码正确，RAM 板块正常。



五、 实验分析总结及心得

（结合所学知识对实验过程中观察到的实验结果进行分析总结，以便加深对知识的理解，并总结通过实验学到的知识或技术）

在进行思考题的实践中，发现 ISE 又出现了一些奇奇怪怪的错误。例如修改完 test 文件之后只能进行一次的仿真波形的呈现。第二次进行的时候总是会强行退出，目前原因尚不明朗。



以上为此次针对思考题进行的仿真。表格如下：

存储器地址	初始化数据	读出数据	写入新数据	读出数据
6’b000001	8’h00000820	8’h00000820	4’b0011	4’b0011
6’b000010	8’h00632020	8’h00632020	4’b0100	4’b0100
6’b000011	8’h00010fff	8’h00010fff	4’b0101	4’b0101
6’b000100	8’h20006789	8’h20006789	4’b0110	4’b0110
6’b000101	8’Hffff0000	8’hffff0000	4’b0111	4’b0111
6’b000110	8’h0000FFFF	8’h0000FFFF	4’b1000	4’b1000
6’b000111	8’h88888888	8’h88888888	4’b1001	4’b1001
6’b001000	8’h99999999	8’h99999999	4’b1010	4’b1010

根据上述表格我们发现，读出数据与初始化关联文件中的数据完全一致，说明相关操作正常。在进行写操作，覆盖初始化数据之后，再执行读操作，这些单元的数据全部被改写。但是相关数据会根据时钟信号进行覆写，若不谨慎编程，将会产生想刷新 A 存储器中的数据却刷新到 A+1 的数据区中。

若想基于上述情况更改为 ROM，在接口上仅需要将写信号至于 0 时，且不再改变即可。并且需要去除 M_W_Data 这一接口。