**计算机组成原理 实验报告**

姓名：魏钊 学号：PB18111699 实验日期：2020-5-4

**一、实验题目：**

实验二 寄存器堆与队列

**二、实验目的：**

* 1. 掌握寄存器堆（Register File）和存储器（Memory）的功能、时序及其应用；
  2. 熟练掌握数据通路和控制器的设计和描述方法。

**三、实验平台：**

ISE / Vivado（暂不支持其他Verilog HDL开发环境的检查）

**四、实验过程：**

**寄存器堆：**

module register\_file

#(parameter WIDTH = 32)

(

input clk, //时钟（上升沿有效）

input [4:0] ra0, //读端口0地址

output [WIDTH-1:0] rd0, //读端口0数据

input [4:0] ra1, //读端口1地址

output [WIDTH-1:0] rd1, //读端口1数据

input [4:0] wa, //写端口地址

input we, //写使能，高电平有效

input [WIDTH-1:0] wd //写端口数据

);

reg [4:0] addr\_reg;

reg [WIDTH-1:0] mem[0:31];

assign rd0=mem[ra0];

assign rd1=mem[ra1];

always@(\*)

mem[0]=1'b0;

always@(posedge clk)

begin

if(we)

if(~(wa==5'b0))

mem[wa]=wd;

mem[0]=0;

end

endmodule

**该寄存器堆含有32 个寄存器（r0 ~ r31，其中r0的内容恒定为零），寄存器的位宽由参数WIDTH指定，具有2个异步读端口和1个同步写端口。**

**存储器：**

例如，设计一容量为16 x 8位（即深度DEPTH：16，宽度WIDTH：8）的单端口RAM，其逻辑符号如图-2所示。用行为方式描述的Verilog代码如下：**（请补充代码中空缺的参数）**

module ram\_16x8 //16x8位单端口RAM

(input clk, //时钟（上升沿有效）

input en, we, //使能，写使能

input [ \_\_3：0\_\_\_ ] addr, //地址

input [ \_7：0\_\_\_\_ ] din, //输入数据

output [ \_\_\_7：0\_\_ ] dout //输出数据

);

reg [ \_\_3：0\_\_\_ ] addr\_reg;

reg [ \_\_\_7：0\_\_ ] mem[ \_\_\_0：16\_\_ ];

//初始化RAM的内容

initial

$readmemh(“初始化数据文件名”, mem);

assign dout = mem[addr\_reg];

always@(posedge clk) begin

if(en) begin

addr\_reg <= addr;

if(we)

mem[addr] <= din;

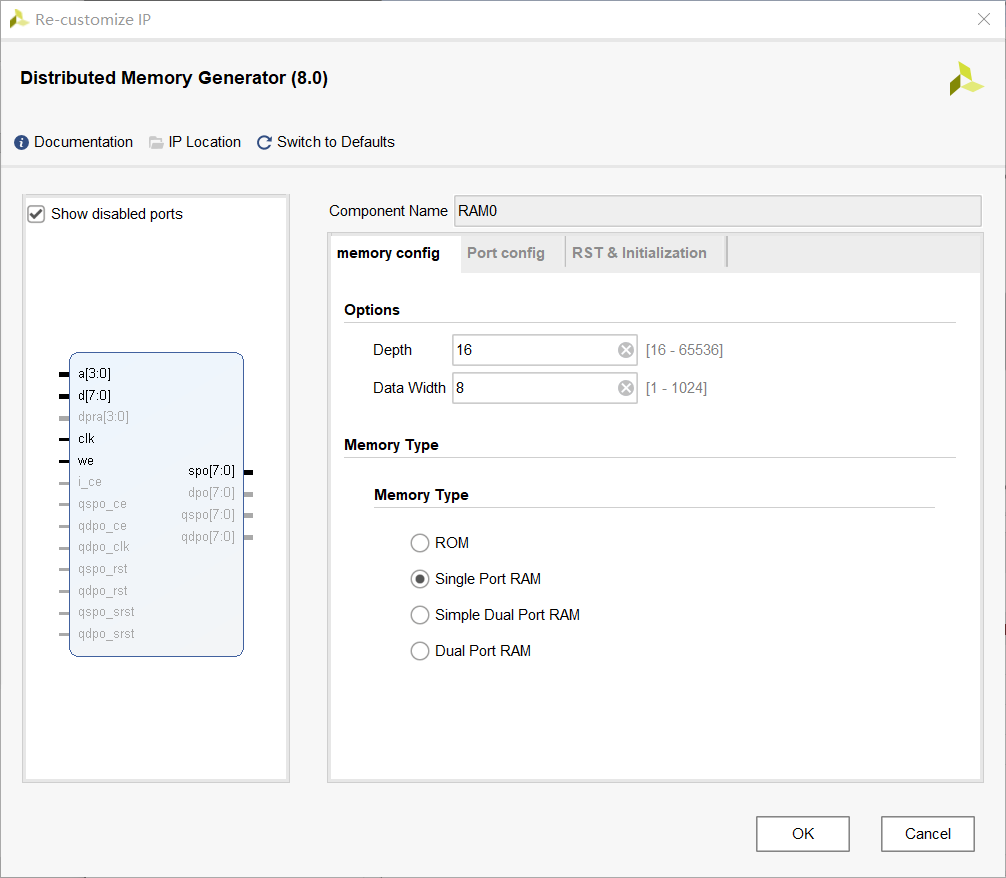
end

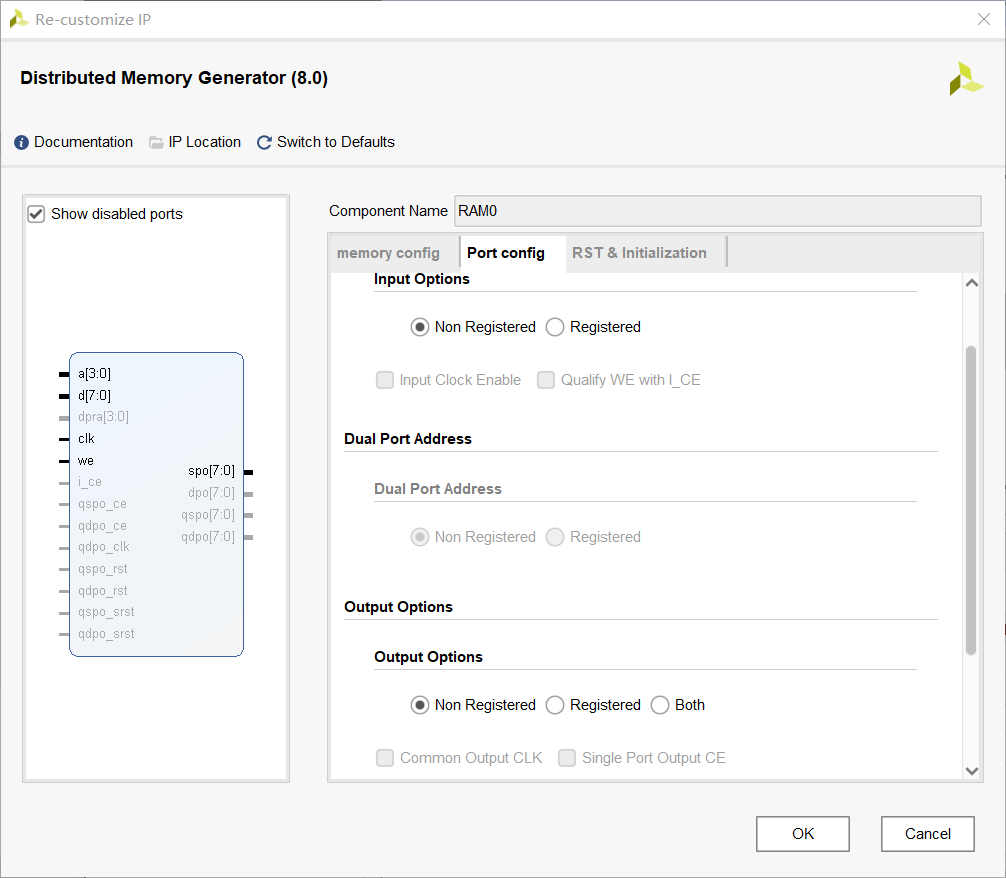
end

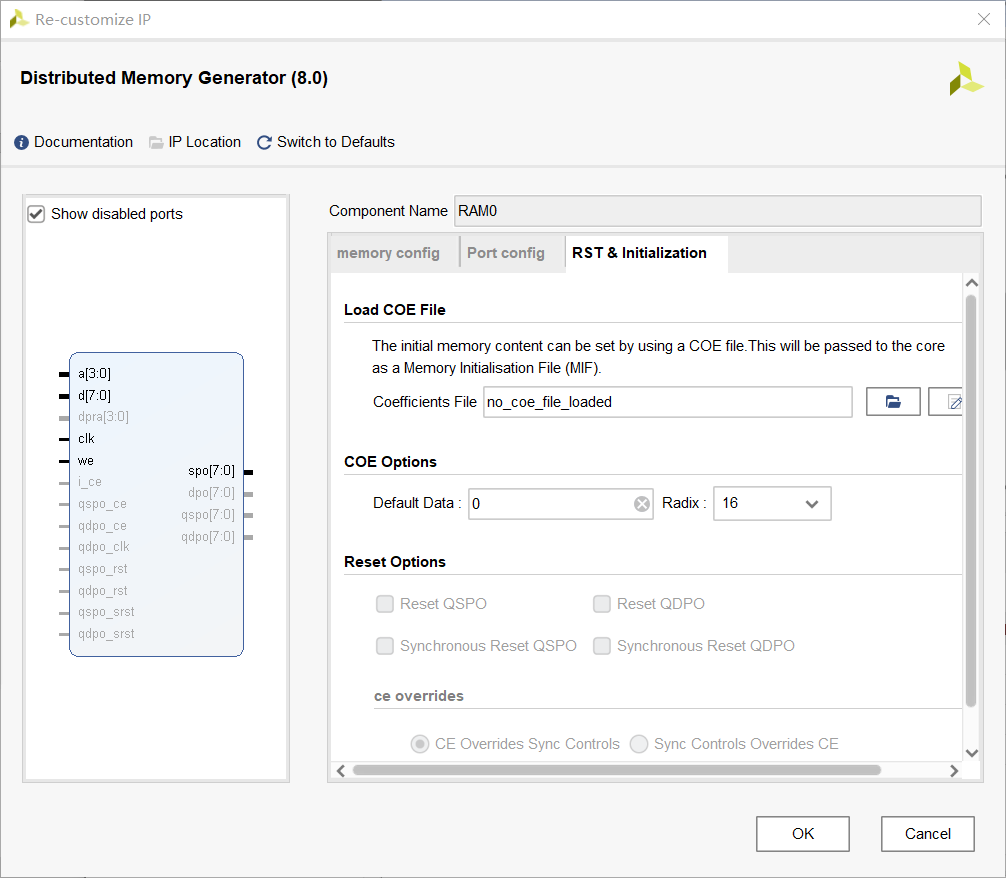
endmodule

**分布式：**

C:\Users\Lucifer.dark\Documents\Tencent Files\709990727\Image\C2C\5MMYHEB_VAL5BSQ{{U(_GOG.png

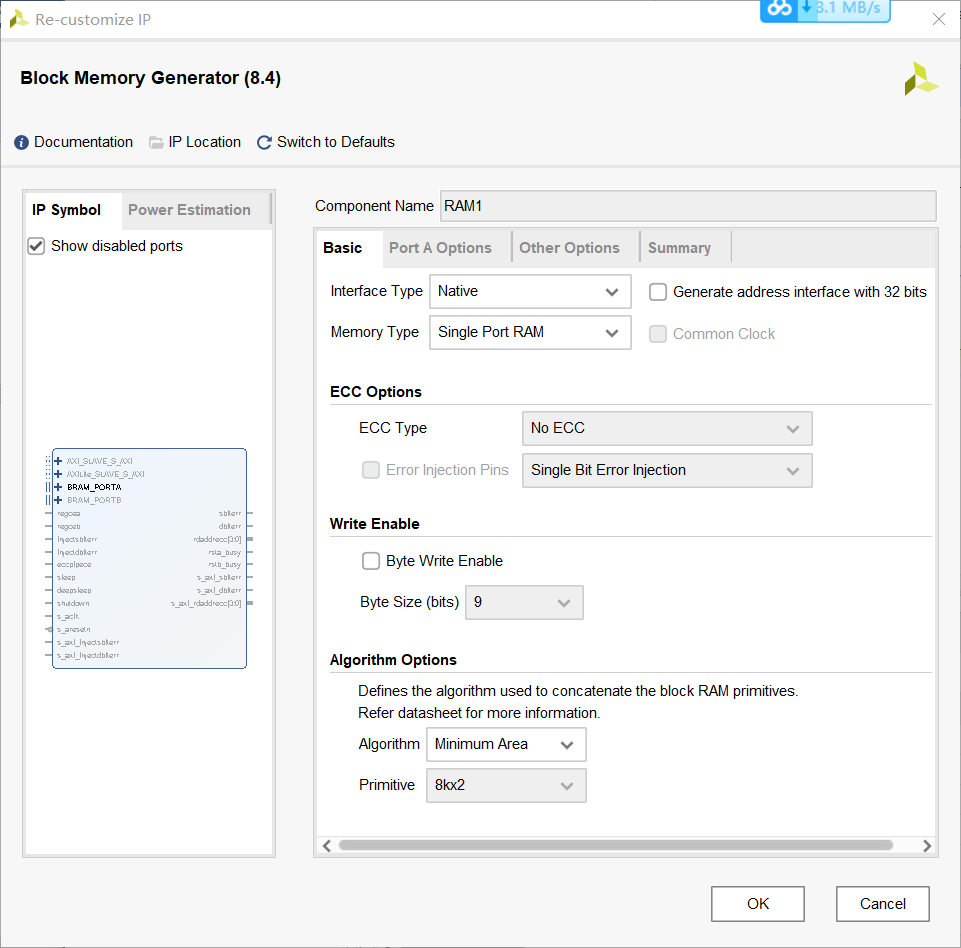


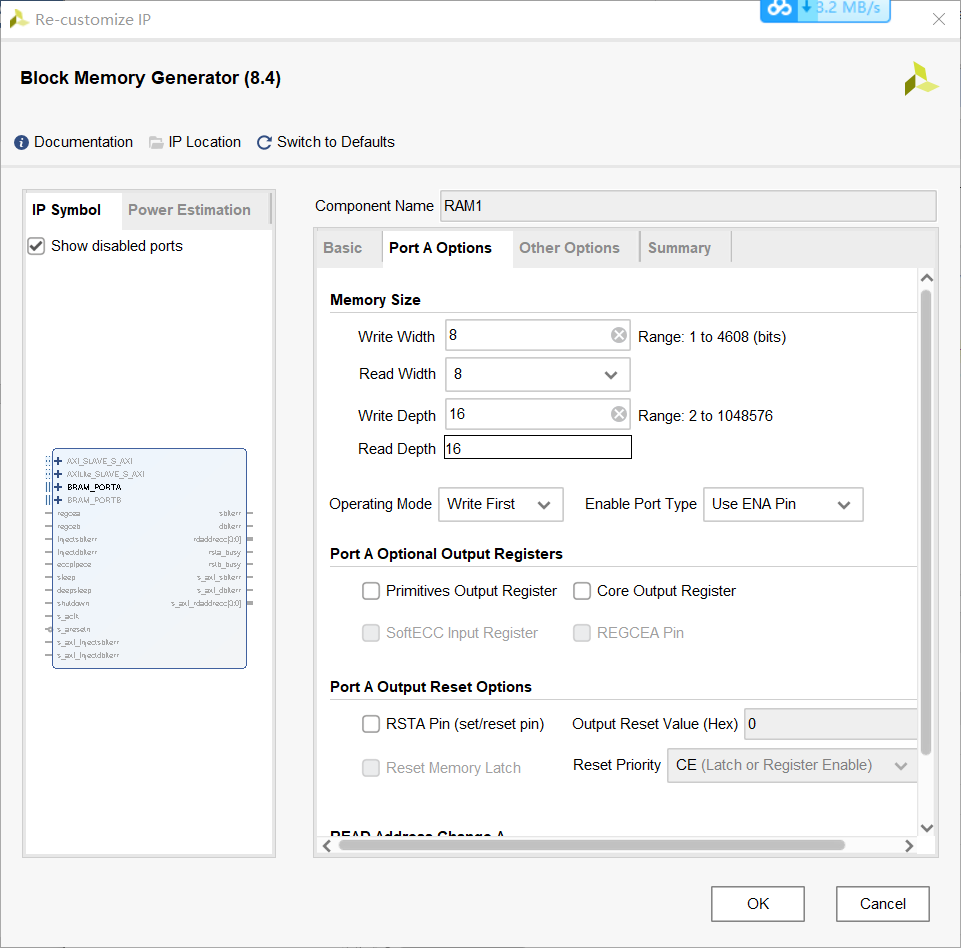


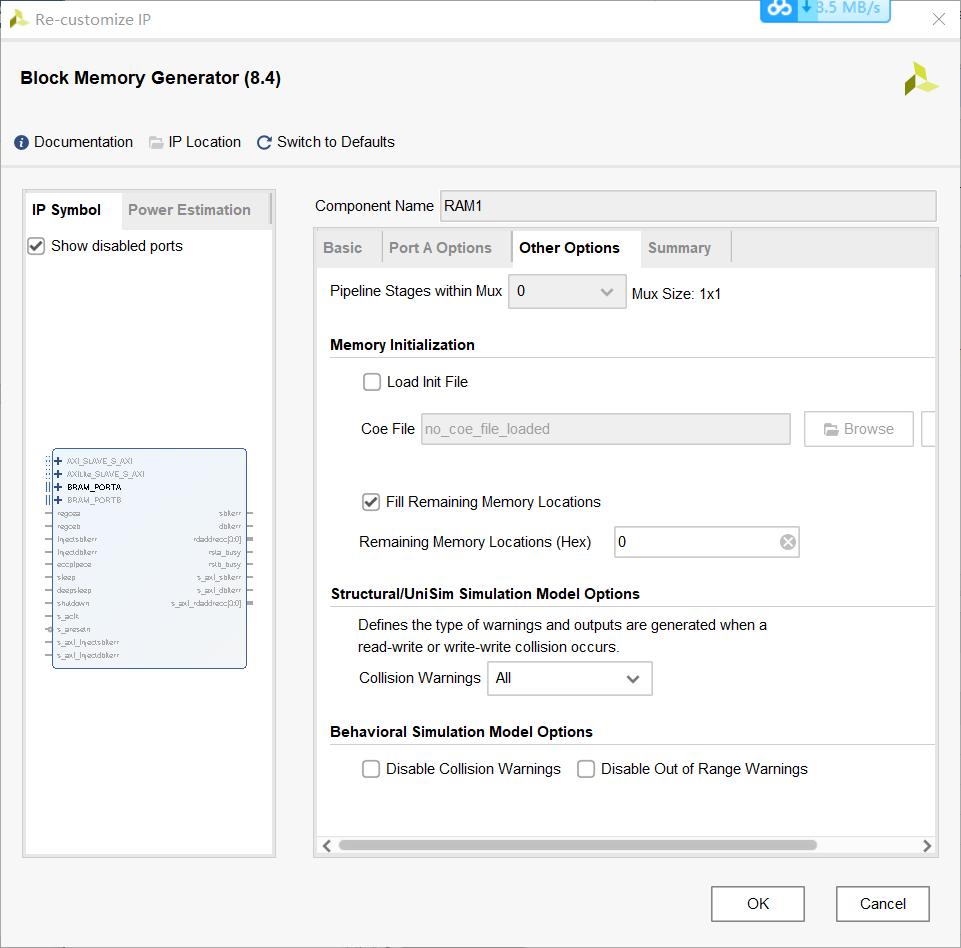


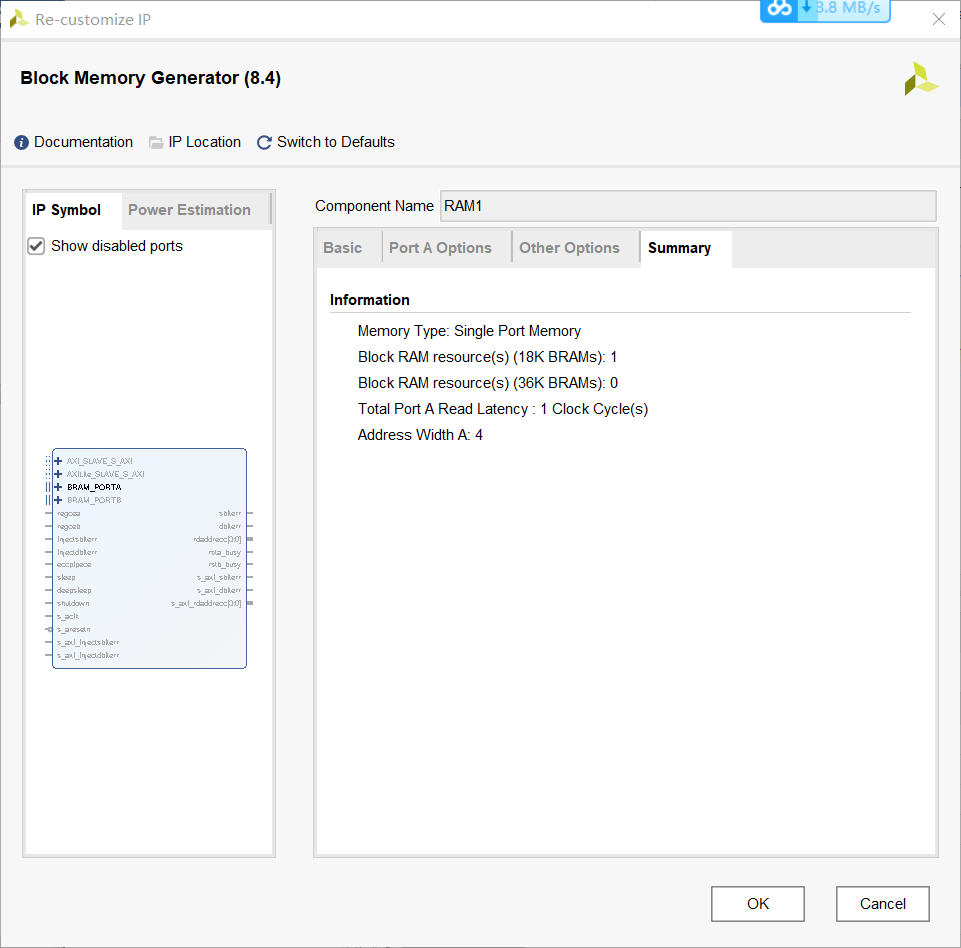
**块式：**

C:\Users\Lucifer.dark\Documents\Tencent Files\709990727\Image\C2C\5OALNU8Y{5`MKW17UWQ~~L1.png



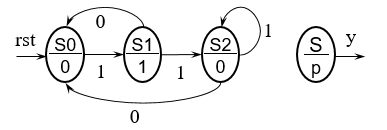






**FIFO:**

**取边沿：**



module EDG(

input clk,

input y,

input rst,

output p

);

parameter S0=2'b00;

parameter S1=2'b01;

parameter S2=2'b10;

reg [1:0] state=S0,next\_state;

assign p=(state==S1);

always@(posedge clk or posedge rst)

if(rst)

state<=S0;

else

state<=next\_state;

always@(\*)

begin

case(state)

S0:if(y)

next\_state=S1;

else

next\_state=S0;

S1:if(y)

next\_state=S2;

else

next\_state=S0;

S2:if(y)

next\_state=S2;

else

next\_state=S0;

endcase

end

endmodule

**FIFO声明及相关变量：**

**module FIFO(**

**input clk,**

**input rst,**

**input en\_in,**

**input [7:0] din,**

**input en\_out,**

**output [7:0] dout,**

**output reg [4:0] count**

**);**

**reg [3:0] head,tail;//c1为队头指针，c2为队尾指针**

**wire in,out;**

**reg en;**

**reg [3:0] addr=4'b0;**

**reg [1:0] state,next\_state;**

**输入输出使能取边沿：**

**EDG edg1(clk,en\_in,rst,in);//输入使能取边沿**

**EDG edg2(clk,en\_out,rst,out);//输出使能取边沿**

**利用例化的存储器IP（16 x 8位块式的单端口RAM）：**

**RAM1 ram(.clka(clk),.ena(en),.wea(in),.addra(addr),.dina(din),.douta(dout));**

**两段式FSM控制电路：**

**always@(posedge clk,posedge rst)**

**begin**

**if(rst)**

**begin**

**state<=2'b00;**

**count<=5'b0;**

**head<=4'b0;**

**tail<=4'b0;**

**addr<=4'b0;**

**end**

**else**

**state<=next\_state;**

**end**

**always@(\*)**

**begin**

**case(state)**

**2'b00:**

**begin**

**count=count;**

**if((in==1'b1) && ~(count==5'b10000))//输入使能且未满**

**begin**

**addr=tail;//地址指向尾指针**

**en=1'b1;**

**next\_state=2'b10;**

**end**

**else**

**if((out==1'b1) && ~(count==5'b0))//输出使能且非空**

**begin**

**addr=head;**

**en=1'b1;**

**next\_state=2'b01;**

**end**

**else**

**begin**

**addr=4'bX;**

**next\_state=2'b00;**

**en=1'b0;**

**end**

**end**

**2'b01://输出**

**begin**

**count=count-5'b1;**

**head=head+4'b1;**

**next\_state=2'b0;**

**end**

**2'b10://输入**

**begin**

**count=count+5'b1;**

**tail=tail+4'b1;**

**next\_state=2'b0;**

**end**

**default:**

**next\_state=2'b00;**

**endcase**

**end**

**endmodule**

**五、实验结果：**

**寄存器堆仿真：**

module register\_file\_test;

reg [31:0] WriteData;

reg [4:0] ReadRegNum1, ReadRegNum2, WriteRegNum;

reg Clock, RegWrite;

wire [31:0] ReadData1, ReadData2;

register\_file regf0 (.clk(Clock), .rd0(ReadData1), .rd1(ReadData2), .ra0(ReadRegNum1), .ra1(ReadRegNum2), .wd(WriteData), .wa(WriteRegNum), .we(RegWrite));

always

#5 Clock = ~ Clock;

initial

begin

#0 Clock = 0;

#0 RegWrite = 1'b1;

#10 WriteRegNum = 1;

#0 WriteData = 1;

#10 WriteRegNum = 2;

#0 WriteData = 2;

#10 WriteRegNum = 3;

#0 WriteData = 3;

#10 WriteRegNum = 4;

#0 WriteData = 4;

#10 WriteRegNum = 0;

#0 WriteData = 1;

#10 WriteRegNum = 5;

#0 WriteData = 5;

#10 RegWrite = 1'b0;

#5 ReadRegNum1 = 1;

#0 ReadRegNum2 = 2;

#5 ReadRegNum1 = 3;

#0 ReadRegNum2 = 0;

#5 ReadRegNum1 = 4;

#0 ReadRegNum2 = 5;

#5 WriteRegNum = 1;

#0 RegWrite = 1'b1;

#0 WriteData = 7;

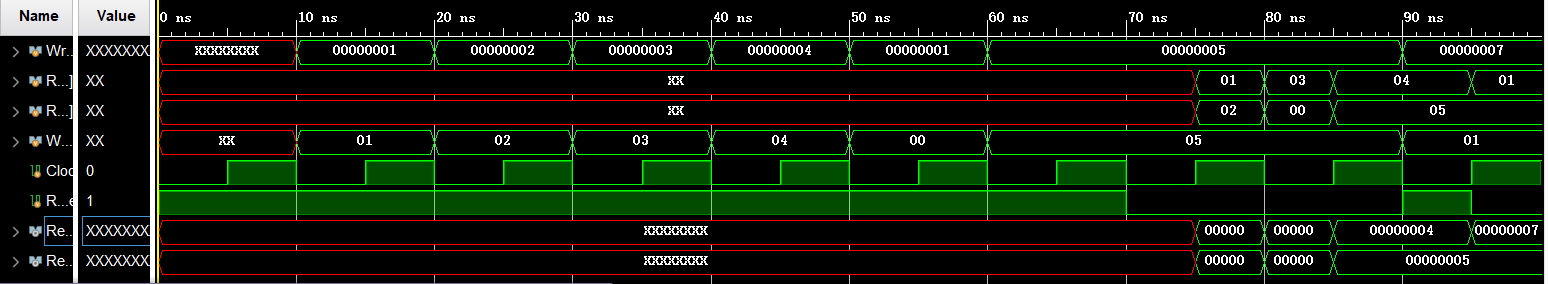
#5 ReadRegNum1 = 1;

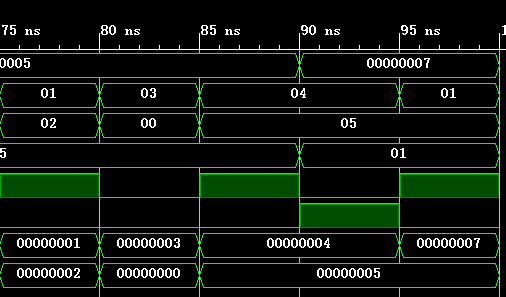
#0 RegWrite = 1'b0;

#5 $finish;

end

endmodule





**分布式RAM：**

module RAM0\_test;

reg [3 : 0] a;

reg [7 : 0] d;

reg clk;

reg we;

wire [7 : 0] spo;

RAM0 ram0(a,d,clk,we,spo);

always

#5 clk = ~ clk;

initial

begin

#0 clk = 1'b0;

#0 we = 1'b1;

#5 a=1;

#0 d=1;

#10 a=4;

#0 d=4;

#10 a=10;

#0 d=10;

#10 a=15;

#0 d=15;

#10 a=7;

#0 d=7;

#10 a=1;

#0 we=0;

#10 a=4;

#10 a=10;

#10 a=15;

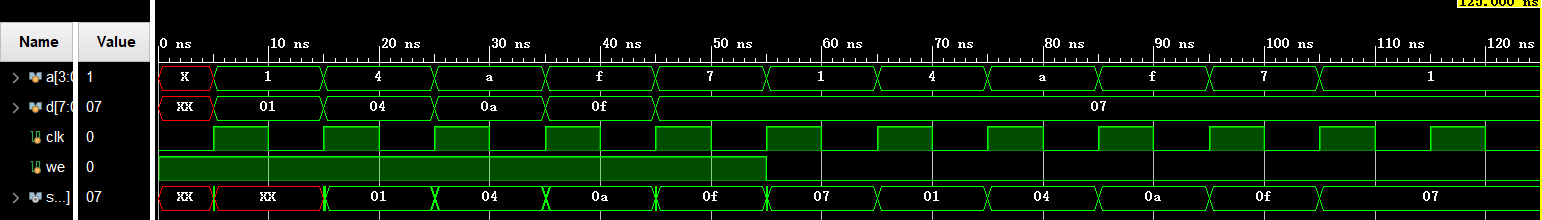
#10 a=7;

#10 a=1;

#20 $finish;

end

endmodule



**块式RAM：**

module RAM1\_test;

reg clk;

reg en;

reg we;

reg [3:0] a;

reg [7:0] d;

wire [7:0] douta;

RAM1 ram1(clk,en,we,a,d,douta);

always

#5 clk = ~ clk;

initial

begin

#0 clk = 1'b0;

#0 we = 1'b1;

#0 en=1'b1;

#5 a=1;

#0 d=1;

#10 a=4;

#0 d=4;

#10 a=10;

#0 d=10;

#10 a=15;

#0 d=15;

#10 a=4'b0;

#0 d=8'b110;

#10 a=7;

#0 d=7;

#10 a=1;

#0 we=0;

#10 a=4;

#10 a=10;

#10 a=15;

#10 a=0;

#10 a=7;

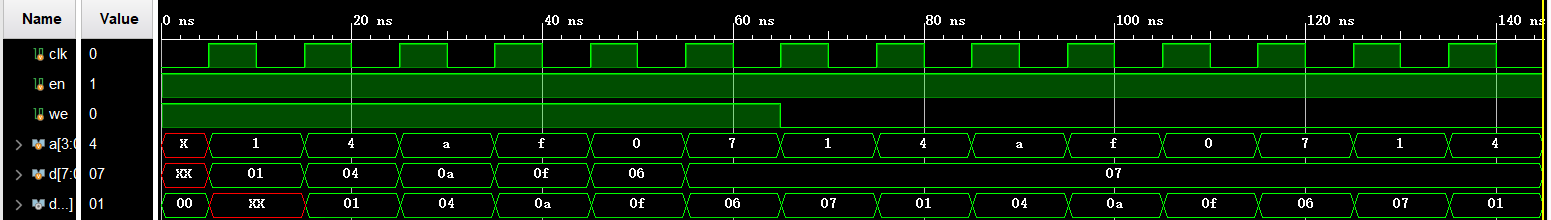
#10 a=1;

#10 a=4;

#10 $finish;

end

endmodule



 块RAM 和 分布式RAM

① Xilinx 的FPGA结构主要由CLB、IOB、IR、Block RAM组成，其中CLB是最最重要的资源。

② 以V5为例，1个CLB包括的2个Slice，每个Slice包括4个6输入查找表，4个FlipFlop和相关逻辑。在这里需要注意的是Slice分两种，SliceM和SliceL，它们都包括前面的东西，但是很特别的是SliceM还增加了基于查找表的分布式RAM和移位寄存器。

③ 每个CLB中都包含SliceL，但并不是每个CLB中都包含SliceM，整个一块V5芯片中SliceM和SliceL的比例为1：3。SliceM的放置有一定的规则，这里不做阐述。

④ Xilinx的FPGA中有 分布式RAM 和 Block RAM 两种存储器。用分布式RAM 时其实要用到其所在的SliceM，所以要占用其中的逻辑资源；而Block RAM 是单纯的存储资源，但是要一块一块的用，不像分布式RAM 想要多少bit都可以。

⑤ 用户申请存储资源，FPGA先提供Block RAM ，当Block RAM 数量不够时再用分布式RAM补充。

**FIFO：**

**module FIFO\_test;**

**reg clk;**

**reg rst;**

**reg en\_in;**

**reg [7:0] din;**

**reg en\_out;**

**wire [7:0] dout;**

**wire [4:0] count;**

**FIFO fifo(clk,rst,en\_in,din,en\_out,dout,count);**

**always**

**#5 clk = ~ clk;**

**initial**

**begin**

**#0 clk = 1'b0;**

**#0 en\_in=1'b0;**

**#0 en\_out=1'b0;**

**#0 rst=1'b1;**

**#10 rst=1'b0;**

**#5 en\_in=1'b1;**

**#0 din=1;**

**#20 en\_in=1'b0;**

**#20 en\_in=1'b1;**

**#0 din=2;**

**#20 en\_in=1'b0;**

**#20 en\_in=1'b1;**

**#0 din=3;**

**#20 en\_in=1'b0;**

**#20 en\_in=1'b1;**

**#0 din=4;**

**#20 en\_in=1'b0;**

**#20 en\_in=1'b1;**

**#0 din=5;**

**#20 en\_in=1'b0;**

**#20 en\_in=1'b1;**

**#0 din=6;**

**#20 en\_in=1'b0;**

**#20 en\_in=1'b1;**

**#0 din=7;**

**#20 en\_in=1'b0;**

**#20 en\_in=1'b1;**

**#0 din=8;**

**#20 en\_in=1'b0;**

**#20 en\_in=1'b1;**

**#0 din=9;**

**#20 en\_in=1'b0;**

**#20 en\_in=1'b1;**

**#0 din=10;**

**#20 en\_in=1'b0;**

**#20 en\_in=1'b1;**

**#0 din=11;**

**#20 en\_in=1'b0;**

**#20 en\_in=1'b1;**

**#0 din=12;**

**#20 en\_in=1'b0;**

**#20 en\_in=1'b1;**

**#0 din=13;**

**#20 en\_in=1'b0;**

**#20 en\_in=1'b1;**

**#0 din=14;**

**#20 en\_in=1'b0;**

**#20 en\_in=1'b1;**

**#0 din=15;**

**#20 en\_in=1'b0;**

**#20 en\_in=1'b1;**

**#0 din=7;**

**#20 en\_in=1'b0;**

**#20 en\_in=1'b1;**

**#0 din=8;**

**#20 en\_in=1'b0;**

**#20 en\_out=1'b1;**

**#20 en\_out=1'b0;**

**#20 en\_out=1'b1;**

**#20 en\_out=1'b0;**

**#20 en\_out=1'b1;**

**#20 en\_out=1'b0;**

**#20 en\_out=1'b1;**

**#20 en\_out=1'b0;**

**#20 en\_in=1'b1;**

**#0 din=9;**

**#20 en\_in=1'b0;**

**#20 en\_in=1'b1;**

**#0 din=11;**

**#20 en\_in=1'b0;**

**#20 en\_out=1'b1;**

**#20 en\_out=1'b0;**

**#20 en\_out=1'b1;**

**#20 en\_out=1'b0;**

**#20 en\_out=1'b1;**

**#20 en\_out=1'b0;**

**#20 en\_out=1'b1;**

**#20 en\_out=1'b0;**

**#20 en\_out=1'b1;**

**#20 en\_out=1'b0;**

**#20 en\_out=1'b1;**

**#20 en\_out=1'b0;**

**#20 en\_out=1'b1;**

**#20 en\_out=1'b0;**

**#20 en\_out=1'b1;**

**#20 en\_out=1'b0;**

**#20 en\_out=1'b1;**

**#20 en\_out=1'b0;**

**#20 en\_out=1'b1;**

**#20 en\_out=1'b0;**

**#20 en\_out=1'b1;**

**#20 en\_out=1'b0;**

**#20 en\_out=1'b1;**

**#20 en\_out=1'b0;**

**#20 en\_out=1'b1;**

**#20 en\_out=1'b0;**

**#20 en\_out=1'b1;**

**#20 en\_out=1'b0;**

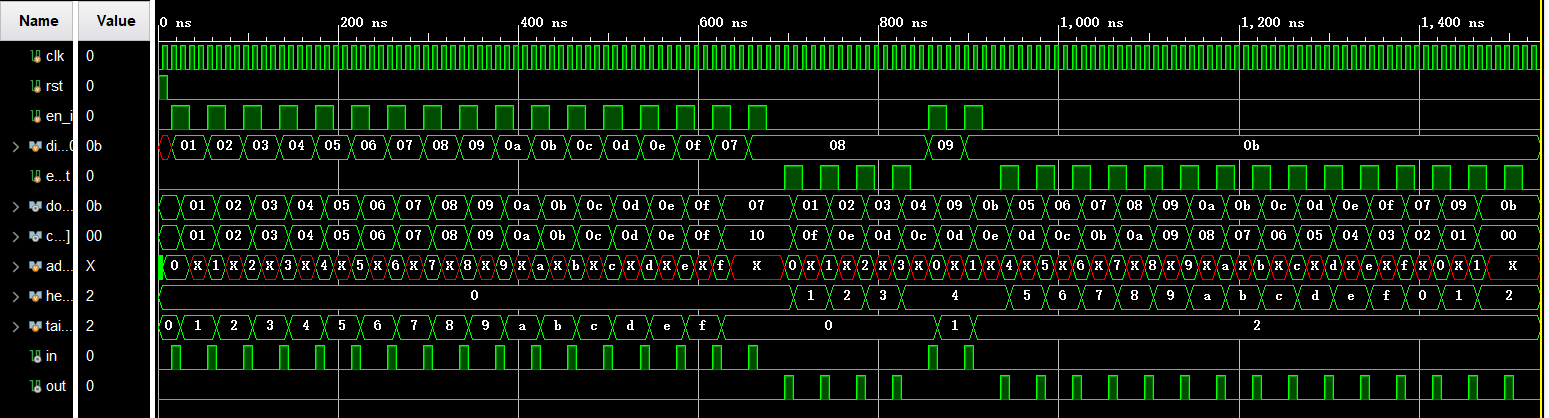
**#20 en\_out=1'b1;**

**#20 en\_out=1'b0;**

**#20 $finish;**

**end**

**endmodule**



1. **心得体会：**

通过本次实验掌握了寄存器堆（Register File）和存储器（Memory）的功能、时序及其应用；熟练掌握了数据通路和控制器的设计和描述方法。