**《数字逻辑》实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | |  | |  | |  |
|  | |  | |  | |  |
| **实验名称** | 实验十四 存储器实验 | | | | | |
| **实验时间** | 2023.11.25 | | **实验地点** | | DS1410 | |
| **实验成绩** |  | | **实验性质** | | **☑验证性 □设计性 □综合性** | |
| 教师评价：  □算法/实验过程正确； □源程序/实验内容提交 □程序结构/实验步骤合理；  □实验结果正确； □语法、语义正确； □报告规范；  评语：  评价教师签名（电子签名）： | | | | | | |
| 一、实验目的  1.掌握随机存储器原理，学会FPGA内部存储器控制器的设计方法。  2.掌握单端口与双端口RAM(随机存储器)设计与实现。  3.掌握FIFO (先入先出存储队列)设计与实现。 | | | | | | |
| 二、实验项目内容  1、利用BASYS3片内存储器单元实现单端口RAM设计(带异步读和同步读两种模式),在时钟(c1k).上升沿,采集地址(addr)、输入数据(data\_ in)、执行相关控制信息。当写使能(we) 有效，则执行写操作，否则执行读取操作。同步与异步设计仅针对读操作:对于异步RAM 而言，读操作为异步，即地址信号有效时，控制器直接读取RAM阵列;对于同步RAM而言，地址信号在时钟上升沿被采集。并保存在寄存器中，然后使用该地址信号读取RAM阵列。  2、 实现双端口(同步与异步)RAM设计,相对于单端口RAM而言，双端口RAM存在两个存取端口,并且可独立进行读写操作，具有自己的地址(addr\_ a、addr\_ b)、数据输入(din\_ a、din\_ b) /输出端口(dout\_ \_a、dout\_ b)以及  3、实现FIFO设计,FIFO由存储单元队列或阵列构成,和RAM不同的是FIFO没有地址，第一个被写入队列的数据也是第-一个从队列中读出的数据。 | | | | | | |
| 1. 实验设计   实验原理：  RAM(random access memory)又称“随机存储器”，存储单元的内容按需要随意取出或者存入，速度很快，但断电时将丢失数据，所以一般被作为临时数据的存储媒介。Basys3开发板上拥有1,800Kbits快速RAM块，可以根据需求定制ROM、RAM或者FIFO。  1.单端口RAM设计(带异步读和同步读两种模式)  在时钟(clk)上升沿,采集地址(addr)、输入数据(data\_in)、执行相关控制信息。当写使能(we)有效，则执行写操作，否则执行读取操作。同步与异步设计仅针对读操作:对于异步RAM而言，读操作为异步，即地址信号有效时，控制器直接读取RAM阵列;对于同步RAM而言，地址信号在时钟上升沿被采集，并保存在寄存器中，然后使用该地址信号读取RAM阵列，单端口RAM框图如下:  Screenshot_20231130_181457  2.双端口(同步与异步) RAM  相对于单端口RAM而言，双端口RAM存在两个存取端口，并且可独立进行读写操作，具有自己的地址(addr\_a、addr\_b)、数据输入(din\_a、din\_b)、输出端口(dout\_a、dout\_b)以及控制信号。双端口RAM常用于视频、图像处理设计中。双端口RAM框图如下:  Screenshot_20231130_181716  3.FIFO设计  FIFO是一个先入先出的存储队列，和RAM不同的是FIFO没有地址，第一个被写入队列的数据也是第一个从队列中读出的数据。FIFO可以在输入输出速率不匹配时，作为临时存储单元;可用于不同时钟域中间的同步;输入数据路径和输出数据路径之间数据宽度不匹配时，可用于数据宽度调整电路。FIFO的框图和信号功能如下:  Screenshot_20231130_181736  Screenshot_20231130_181748 | | | | | | |
| 四、实验过程或算法  1.单端口RAM设计代码实现：  （1）编写 Design source 文件  //同步  module ram\_single\_same(clk,we,addr,data\_in,data\_out  );  input clk;  input we;  input [1:0]addr;  input[3:0]data\_in;  output reg [3:0]data\_out;  reg [3:0] ram[3:0];  always@(posedge clk)begin  if(we)begin  ram[addr]<=data\_in;  end  else  data\_out<=ram[addr];  end  endmodule  //异步  module ram\_single\_dif(clk,we,addr,data\_in,data\_out  );  input clk;  input we;  input [1:0]addr;  input[3:0]data\_in;  output reg [3:0]data\_out;  reg [3:0] ram[3:0];  always@(posedge clk)begin  if(we)begin  ram[addr]<=data\_in;  end  end  always@(we or addr)begin  if(!we)  data\_out<=ram[addr];  end  Endmodule   1. 通过行为仿真验证正确性   仿真代码：  `timescale 1ns / 1ns  module ram\_single\_same\_tb;  reg clk,we;  reg[1:0]addr;  reg[3:0]data\_in;  wire [3:0]data\_out;  ram\_single\_same u(clk,we,addr,data\_in,data\_out);  initial clk=0;  always #7 clk=~clk;  initial  begin  #50 addr=2'b00;we=1;data\_in=4'b0000;  #50 addr=2'b01;we=1;data\_in=4'b0011;  #50 addr=2'b10;we=1;data\_in=4'b0001;  #50 addr=2'b11;we=1;data\_in=4'b1110;  #50 addr=2'b00;we=0;data\_in=4'b0111;  #50 addr=2'b01;we=0;data\_in=4'b0101;  #50 addr=2'b10;we=0;data\_in=4'b1101;  #50 addr=2'b11;we=0;data\_in=4'b1111;  end  endmodule  （3）波形图    单端口同步 RAM 波形图    单端口异步 RAM 波形图  （4）RTL分析，查看电路图    单端口 RAM RTL 电路图  2.双端口代码(同步和异步）实现：  （1）编写 Design source 文件  module  ram\_dual(clk,we\_a,we\_b,addr\_a,addr\_b,din\_a,din\_b,dout\_a,dout\_b);  input clk;  input we\_a;  input we\_b;  input[1:0]addr\_a;  input[1:0]addr\_b;  input[3:0]din\_a;  input[3:0]din\_b;  output reg[3:0]dout\_a;  output reg[3:0]dout\_b;  reg [3:0]ram\_a[3:0];  reg[3:0]ram\_b[3:0];  always@(posedge clk)begin //a 和 b 同步写操作  if(we\_a)  ram\_a[addr\_a]<=din\_a;  if(we\_b)  ram\_b[addr\_b]<=din\_b;  end  always@(posedge clk)begin //a 实现同步读操作  if(!we\_a)  dout\_a<=ram\_a[addr\_a];  end  always@(we\_b or addr\_b)begin //b 实现异步读操作  if(!we\_b)  dout\_b<=ram\_b[addr\_b];  end  endmodule  （2）通过行为仿真验证正确性  仿真代码:  `timescale 1ns / 1ns  module ram\_dual\_tb;  reg clk;  reg we\_a;  reg we\_b;  reg[1:0]addr\_a;  reg[1:0]addr\_b;  reg[3:0]din\_a;  reg[3:0]din\_b;  wire[3:0]dout\_a;  wire [3:0]dout\_b;  ram\_dual  u(clk,we\_a,we\_b,addr\_a,addr\_b,din\_a,din\_b,dout\_a,dout\_b);  initial clk=0;  always #7 clk=~clk;  initial begin  din\_a=0;din\_b=1;we\_a=1;we\_b=1;addr\_a=0;addr\_b=0;  #50 din\_a=1;din\_b=3;we\_a=1;we\_b=1;addr\_a=1;addr\_b=1;  #50 din\_a=3;din\_b=2;we\_a=1;we\_b=1;addr\_a=2;addr\_b=2;  #50 din\_a=2;din\_b=0;we\_a=1;we\_b=1;addr\_a=3;addr\_b=3;  #50 din\_a=0;din\_b=1;we\_a=0;we\_b=0;addr\_a=0;addr\_b=0;  #50 din\_a=1;din\_b=2;we\_a=0;we\_b=0;addr\_a=2;addr\_b=1;  #50 din\_a=2;din\_b=3;we\_a=0;we\_b=0;addr\_a=1;addr\_b=2;  #50 din\_a=3;din\_b=0;we\_a=0;we\_b=0;addr\_a=3;addr\_b=3;  end  Endmodule  (3)波形图    双端口 RAM 波形图  （4）RTL 分析，查看电路图    双端口 RAM RTL 电路图  3.FIFO设计  （1）编写 Design Source 文件  ①顶层模块  module top\_fifo(  input clk,button,rst,rd\_en,wr\_en,  input [3:0]data\_in,  output[3:0]data\_out,  output full,empty);  wire de\_button;  debkey u0(.clk(clk),.rst(rst),.key\_in(button),.key\_out(de\_button));  //debkey中的de\_button输出信号作为输入信号给fifo模块  fifo u1(.clk(de\_button),.rst(rst),.wr\_en(wr\_en),.rd\_en(rd\_en),.data\_in(data\_in),  .empty(empty),.full(full),.data\_out(data\_out));  endmodule  ②按键消抖模块  module debkey(//消抖  input clk,  input rst,  input key\_in,  output key\_out);  parameter T100Hz = 249999;  integer cnt\_100Hz;  reg clk\_100Hz;    always @(posedge clk) begin  if(rst)  cnt\_100Hz<=32'b0;  else begin  cnt\_100Hz<=cnt\_100Hz+1'b1;  if(cnt\_100Hz==T100Hz) begin  cnt\_100Hz<=32'b0;  clk\_100Hz<=~clk\_100Hz;  end  end  end  reg[2:0]key\_rrr,key\_rr,key\_r;  always @(posedge clk\_100Hz) begin  if(rst) begin  key\_rrr<=1'b1;  key\_rr<=1'b1;  key\_r<=1'b1;  end  else begin  key\_rrr<=key\_rr;  key\_rr<=key\_r;  key\_r<=key\_in;  end  end  assign key\_out = key\_rrr&key\_rr&key\_r;  endmodule  ③FIFO核心模块  module fifo(  input clk,rst,rd\_en,wr\_en,  input [3:0]data\_in,  output reg [3:0]data\_out,  output full,empty);    reg [3:0]arm[3:0];  reg [2:0]counter;  reg [1:0]wr\_p=2'b00;  reg [1:0]rd\_p=2'b00;  assign full=(counter==4)?1:0;  assign empty=(counter==0)?1:0;  always@(posedge clk or posedge rst) begin  if(rst)  counter<=0;  else if(wr\_en&&!rd\_en&&!full)  counter<=counter+1;  else if(!wr\_en&&rd\_en&&!empty)  counter<=counter-1;  else  counter<=counter;  end  always@(posedge clk or posedge rst) begin  if(rst)begin  wr\_p<=0;  end  else if(wr\_en&&!full)begin  arm[wr\_p]<=data\_in;  wr\_p<=wr\_p+1;  end  end  always@(posedge clk or posedge rst) begin  if(rst) begin  rd\_p<=0;  data\_out<=0;  end  else if(rd\_en&&!empty)begin  data\_out<=arm[rd\_p];  rd\_p<=rd\_p+1;  end  end  endmodule  （2）通过行为仿真验证正确性  代码如下：  `timescale 1ns / 1ns  module top\_fifo\_tb();  reg clk,rst,rd\_en,wr\_en;  reg [3:0]data\_in;  wire [3:0]data\_out;  wire full,empty;  fifo u( .clk(clk),.rst(rst),.rd\_en(rd\_en),.wr\_en(wr\_en),.data\_in(data\_in),.data\_out(data\_out),.full(full),.empty(empty) );  initial clk=1;  always #50 clk=~clk;    initial begin  data\_in=4'b0000;wr\_en=0;rd\_en=0;rst=1;  #100;  data\_in=4'b0001;wr\_en=1;rd\_en=0;rst=0;  #100;  data\_in=4'b0010;wr\_en=1;rd\_en=0;rst=0;  #100;  data\_in=4'b0100;wr\_en=1;rd\_en=0;rst=0;  #100;  data\_in=4'b1000;wr\_en=1;rd\_en=0;rst=0;  #100;  data\_in=4'b1001;wr\_en=1;rd\_en=0;rst=0;  #100;  data\_in=4'b0110;wr\_en=0;rd\_en=1;rst=0;  #100;  end  endmodule   1. 波形图     FIFO 仿真波形图  (4) RTL 分析，查看电路图    FIFO RTL 电路图 | | | | | | |
| （5）实现综合 Run Synthesis  （6）Run implementation 后实现管脚分配    （7）创建比特流  （8）连接板子，烧录  （9）上板实现，再次验证  7DEC9D369CF46A575618EAFCEC684BCB  验证成功，功能实现正常。 | | | | | | |
| 五、实验过程中遇到的问题及解决情况  1.FIFO 上板验证时，发现出现按键抖动现象影响实验实现。  解决：学习老师提供的按键消抖模块学习文档，设计按键消抖模块，来消除抖动对实验带来的干扰。  2.判断 FIFO 的满空情况出现困难。  解决：利用count，计数队列中的个数。当队列中数目达到最大值，则亮“满状态”灯，全部出队或者队为空时，亮“空状态”灯。  3.对FIFO存储器定义不清晰，在设计代码时因为不理解出现很多错误  解决：利用网络平台去查阅FIFO相关知识和定义，明白先进先出队列的概念和实现思路。  4.在分配输入输出管脚时出现错误  解决：利用button键和led灯，将入队和出队操作分别实现。  5.对时序电路不是很熟悉，在设计同步异步电路时出现代码错误  解决：阅读相关可见PPT和教材，并查阅网上资料，学习同步和异步操作。 | | | | | | |
| 六、实验结果及分析和（或）源程序调试过程  1.仿真结果：  单端口仿真实现：  仿真波形图：    双端口仿真实现：  仿真波形图：    FIFO仿真实现：  仿真波形图：    2.实验结果  利用随机存储器原理，学会 FPGA 内部存储器控制器的设计方法，成功  设计并实现了单端口与双端口 RAM（随机存储器），完成了 FIFO（先入  先出存储队列）的设计与实现  7DEC9D369CF46A575618EAFCEC684BCB  七、小组分工情况说明  主要负责实验代码编写、实际实现、仿真波形、RTL电路图观测和实验报告收尾工作  主要负责编写实验报告工作 | | | | | | |