**《数字逻辑》实验报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **姓名** | |  | | **年级** | | 2020级 |
| **学号** | |  | | **专业、班级** | | 计算机科学与技术计卓2班 |
| **实验名称** | **储存器实验** | | | | | |
| **实验时间** | **2021.11.25** | | **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（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。  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（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。  3、 实现 FIFO 设计，FIFO 由存储单元队列或阵列构成，和RAM不同的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 | | | | | | |
| 三、实验设计  **单端口RAM**  根据RAM设计需求，单端口RAM包含一个地址端口、一个数据输入端口、一个数据输出端口，同时通过写使能、读使能信号控制读写行为，完成对数据的读取和写入。  针对实验要求中的同时具有同步读和异步读的要求，本实验中加入了mode信号来选择最终读取数据的来源，需注意，实际上本实验的设计中同时产生了异步读和同步读RAM的输出，只是根据需求通过mode信号对于输出结果进行选择。  同时针对器件使用时可能面临的拓展问题，本实验中加入了en使能信号来控制RAM工作状态。实验对于读写模式的要求是非写即读，产生了读取数据有效性的问题，因此在设计中加入了valid信号，表示读出数据的有效性。    **双端口RAM**  根据双端口RAM的设计需求，调用vivado提供的Block Memory Generator生成双端口RAM。  需要注意的是生成RAM过程中参数的设置，除了设置基本的位长和深度等参数，为保证读取数据的准确性，选择写优先模式，同时为了去除不必要的时延，取消RAM后寄存器的生成。    **FIFO**  本实验在完成RAM设计的基础上进行FIFO的设计。  在利用RAM储存数据的同时，额外加入两组寄存器分别记录FIFO两端的地址，即数据写入端的头地址和数据读出端的尾地址。写入数据时，将头地址递增，给与RAM写使能信号，则完成了FIFO的写入，读出数据时，将尾地址递增，同时给与RAM读使能信号，则完成了FIFO的读出。  针对FIFO空和满的状态需要特殊说明。上述对于头地址和尾地址的说明中忽略了一点：两个地址的位长应该比能表示最大地址的位长多出最高位的一位。通过多出这一位的两个地址可以判断FIFO当前是否是满状态或空状态：当两个地址完全相同时，FIFO为空，当两个地址最高位不同，而其他位完全相同时，FIFO为满，这是因为最高位不同说明两个地址并不处于同一趟数（即头指针已经递增并从尾部“追赶”上了尾指针），于是利用这两个地址便可设计出表示FIFO已满的fu信号和FIFO为空的em信号。    需要补充的是，针对FIFO为空的情况，本实验的设计中不允许读出数据，当FIFO为满时，本设计中不允许写入数据。  **逻辑器件**  1、寄存器  2、数据选择器  3、储存器 | | | | | | |
| 四、实验过程或算法  **单端口RAM**  module RAM(clk,rst,addr,data\_in,we,en,mode,data\_out,out1,out2,valid);  parameter WIDTH\_DATA=16;  parameter WIDTH\_ADDRESS=10;  input clk;  input rst;  input mode;  input we;  input en;  input [WIDTH\_DATA-1:0]data\_in;  input [WIDTH\_ADDRESS-1:0]addr;  output [WIDTH\_DATA-1:0]data\_out;  output reg valid;  reg [WIDTH\_DATA-1:0]data[((1<<WIDTH\_ADDRESS)-1):0];  output reg [WIDTH\_DATA-1:0]out1,out2;  integer k;  always@(posedge clk,negedge rst)  begin  if(!rst)  begin  for(k=0;k<=(1<<WIDTH\_ADDRESS)-1;k=k+1)data[k]<=0;  out1<=0;  out2<=0;  end  else  begin  if(en)  begin  valid<=1;  if(we)data[addr]<=data\_in;  else out2<=data[addr];  end  else valid<=0;  end  end  always@(addr)  begin  out1=data[addr];  end  assign data\_out=mode?out1:out2;  endmodule  **双端口RAM（引入IP）**  module RAM(clka,clkb,addra,addrb,dina,dinb,ena,enb,wea,web,douta,doutb);  parameter WIDTH\_DATA=16;  parameter WIDTH\_ADDRESS=10;  input clka,clkb;  input [WIDTH\_ADDRESS-1:0]addra,addrb;  input [WIDTH\_DATA-1:0]dina,dinb;  input ena,enb;  input wea,web;  output [WIDTH\_DATA-1:0]douta,doutb;  mem\_0 i(  .clka(clka),  .clkb(clkb),  .addra(addra),  .addrb(addrb),  .dina(dina),  .dinb(dinb),  .ena(ena),  .enb(enb),  .wea(wea),  .web(web),  .douta(douta),  .doutb(doutb)  );  Endmodule  **FIFO（引入IP）**  module FIFO(clk,rst,data\_in,wen,ren,data\_out,em,fu,front,rear);  parameter DATA\_WIDTH=16;  parameter ADDRESS\_WIDTH=10;  input clk,rst;  input [DATA\_WIDTH-1:0]data\_in;  input wen,ren;  output [DATA\_WIDTH-1:0]data\_out;  output em,fu;  output reg [ADDRESS\_WIDTH:0]front,rear;  reg wea,enb;  reg [DATA\_WIDTH-1:0]din;  reg [ADDRESS\_WIDTH:0]addra,addrb;  wire [DATA\_WIDTH-1:0]temp\_out;  blk\_mem\_gen\_0 i(  .clka(clk),  .clkb(clk),  .addra(addra),  .dina(din),  .ena(1),  .wea(wea),  .addrb(addrb),  .enb(enb),  .doutb(data\_out)  );  always@(posedge clk,negedge rst)  begin  if(!rst)  begin  front<=0;  rear<=0;  end  else  begin  if(wen&(~fu))  begin  wea<=1;  din<=data\_in;  addra<=front;  front<=front+1;  end  else wea<=0;    if(ren&(~em))  begin  enb<=1;  addrb<=rear;  rear<=rear+1;  end  else enb<=0;  end  end  assign em=(front==rear);  assign fu=((front[ADDRESS\_WIDTH]^rear[ADDRESS\_WIDTH])&(front[ADDRESS\_WIDTH-1:0]==rear[ADDRESS\_WIDTH-1:0]));  endmodule | | | | | | |
| 五、实验过程中遇到的问题及解决情况  1、RAM IP读取时延。在设置IP参数时若不取消生成寄存器的选项则会在输出数据时有一个时钟周期的时延，通过查阅资料解决。  2、FIFO生成后资源消耗极大。若通过行为语言方式设计FIFO，结果生成的电路中大量使用逻辑资源，造成极大浪费，对此人为引入RAM IP用作数据的储存器件。 | | | | | | |
| 六、实验结果及分析和（或）源程序调试过程  **单端口RAM**  1、仿真结果    2、资源使用和能耗情况    资源使用情况    能耗情况  **双端口RAM**  1、仿真结果    2、资源使用和能耗情况    资源使用情况    能耗情况  **FIFO**  1、仿真结果    2、资源使用和能耗情况    资源使用情况    能耗情况  七、小组分工情况说明 | | | | | | |