课程实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **课程名称** | 计算机组成原理 | **指导教师** | 钟 旭  李运兰 | **实验日期** | 2022年04月6日 |
| **班 级** | 20软件2班 | **姓 名** | 李佳骏 | **学 号** | B20190103224 |
| **实验成绩** |  | | | | |

|  |
| --- |
| **一、实验项目名称**  数据通路实验 |
| **二、实验目的**  1. 运算器的组成结构及工作原理。  2. 简单运算器的数据通路与控制信号的关系。  3．数据算术运算及逻辑运算方法。  4. 数据通路中运算器与存储器协调工作原理。。  5. 数据及地址在数据通路上传输方法 |
| **三、实验要求**  必开，4学时。一人一组独立完成。 |
| **四、实验环境**  硬件：PC机一台 Intel Pentium 4 CPU 1.2GHz，512MB RAM及以上；  FPGA实验板一块（DDA-III主芯片*CycloneII（EP2C5T144C8））；*  软件：Windows 10，QUARTUS II配套软件 |
| **五、实验内容**   1. 设计组合逻辑运算单元：基本部分为，算术逻辑运算单元，通用寄存器组，输入数据选择电路，输出数据控制电路；实现两个操作数常用的算术运算和逻辑运算功能。两个参加运算的数A，B分别来自DR1和DR2，运算结果通过收发器74244经数据总线至通用寄存器R4或R5。 2. 依据实验原理图，利用运算器exp\_r\_alu与存储器ram\_io\_vhd源设计文件连接成数据通路 3. 完成芯片设置与管脚设置 4. 波形仿真验证或实验板硬件验证 |
| **六、实验原理**  1、运算器原理图。    2.数据通路原理图 |
| **七、实验步骤 （数据通路）**  (1) 新建源文件  新建文件夹sjtl，新建空白block文件，并保存在sjtl文件夹中。  新建VHDL文件exp\_r\_alu，VHDL代码如下：  *library ieee;*  *use ieee.std\_logic\_1164.all;*  *use ieee.std\_logic\_unsigned.all;*  *entity exp\_r\_alu is //运算器实体描述*  *port( clk :in std\_logic; //端口定义*  *sw\_bus,r4\_bus,r5\_bus,alu\_bus :in std\_logic;*  *lddr1,lddr2,ldr4,ldr5 :in std\_logic;*  *m,cn :in std\_logic;*  *s :in std\_logic\_vector(3 downto 0);*  *k :in std\_logic\_vector(7 downto 0);*  *d :inout std\_logic\_vector(7 downto 0) );*  *end exp\_r\_alu;*  *architecture rtl of exp\_r\_alu is //运算器结构体描述*  *signal dr1,dr2,r4,r5,aluout,bus\_reg:std\_logic\_vector(7 downto 0);*  *signal sel:std\_logic\_vector(5 downto 0);*  *begin*  *ldreg:process(clk,lddr1,lddr2,ldr4,ldr5,bus\_reg)*  *begin*  *if clk'event and clk='1' then*  *if lddr1='1' then dr1<=bus\_reg;*  *elsif lddr2='1' then dr2<=bus\_reg;*  *elsif ldr4='1' then r4<=bus\_reg;*  *elsif ldr5='1' then r5<=bus\_reg;*  *end if;*  *end if;*  *end process;*  *alu:process(m,cn,s,dr1,dr2,sel,aluout) //运算器各运算类型所对应的逻辑功能*  *begin*  *sel<=m & cn & s;*  *case sel is*  *when "000000" => aluout<=dr1+1;*  *when "010000" => aluout<=dr1;*  *when "100000" => aluout<=not dr1;*  *when "000001" => aluout<=(dr1 or dr2)+1;*  *when "010001" => aluout<=dr1 or dr2;*  *when "100001" => aluout<=not(dr1 or dr2);*  *when "000010" => aluout<=(dr1 or (not dr2))+1;*  *when "010010" => aluout<=dr1 or (not dr2);*  *when "100010" => aluout<=(not dr1)and dr2;*  *when "000011" => aluout<=x"00";*  *when "010011" => aluout<=aluout-1;*  *when "100011" => aluout<=x"00";*  *when "000100" => aluout<=dr1+(dr1 and (not dr2))+1;*  *when "010100" => aluout<=dr1+(dr1 and (not dr2));*  *when "100100" => aluout<=not (dr1 and dr2);*  *when "000101" => aluout<=((dr1 or dr2 or dr1) and dr2) or x"01";*  *when "010101" => aluout<=(dr1 or dr2)+(dr1 and(not dr2));*  *when "100101" => aluout<=not dr2;*  *when "000110" => aluout<=dr1-dr2;*  *when "010110" => aluout<=dr1-dr2-1;*  *when "100110" => aluout<=dr1 xor dr2;*  *when "000111" => aluout<=dr1 and(not dr2);*  *when "010111" => aluout<=(dr1 and (not dr2))-1;*  *when "100111" => aluout<=dr1 and(not dr2);*  *when "001000" => aluout<=dr1+(dr1 and dr2)+1;*  *when "011000" => aluout<=dr1+(dr1 and dr2);*  *when "101000" => aluout<=(not dr1)or dr2;*  *when "001001" => aluout<=dr1+dr2+1;*  *when "011001" => aluout<=dr1+dr2;*    *when "101001" => aluout<=dr1 xnor dr2;*  *when "001010" => aluout<=(dr1 or(not dr2))+(dr1 and dr2)+1;*  *when "011010" => aluout<=(dr1 or(not dr2))+(dr1 and dr2);*  *when "101010" => aluout<=dr2;*  *when "001011" => aluout<=dr1 and dr2;*  *when "011011" => aluout<=(dr1 and dr2)-1;*  *when "101011" => aluout<=dr1 and dr2;*  *when "001100" => aluout<=dr1+dr1+1;*  *when "011100" => aluout<=dr1 or dr1;*  *when "101100" => aluout<=x"01";*  *when "001101" => aluout<=(dr1 or dr2)+dr1+1;*  *when "011101" => aluout<=(dr1 or dr2)+dr1;*  *when "101101" => aluout<=dr1 or(not dr2);*  *when "001110" => aluout<=(dr1 or (not dr2))+dr1+1;*  *when "011110" => aluout<=(dr1 or (not dr2))+dr1;*  *when "101110" => aluout<=dr1 or dr2;*  *when "001111" => aluout<=dr1;*  *when "011111" => aluout<=dr1-1;*  *when "101111" => aluout<=dr1;*  *when others => aluout<=x"ff";*  *end case;*  *end process; //结束进程*  *bus\_reg<=k when sw\_bus='0' else*  *r4 when r4\_bus='0' else*  *r5 when r5\_bus='0' else*  *aluout when alu\_bus='0' else*  *d;*  *d<=bus\_reg when (sw\_bus='0' or r4\_bus='0' or r5\_bus='0' or alu\_bus='0') else*  *(others=>'Z’);*  *end rtl;*  保存至shtl文件夹。  将上个实验完成的ccq\_8中如下文件加入本文件夹中，具体文件如下图：    打开ccq\_8和exp\_r\_alu的block文件，并将ccq\_8和exp\_r\_alu进行符号化（File->Create\Update->Create Symbol Files For Current File)  打开空白block，加入已符号化的ccq\_8和exp\_r\_alu，如下图进行连线:     1. 新建工程   将空白block命名为sjtl，并保存在sjtl文件夹中，为它新建一个工程，在创建工程时，要加入如下文件：    进行编译   1. 仿真运行   新建仿真波形文件，加入引脚时选择如下：      对各引脚赋值  进行仿真，结果如下图所示： |
|  |
| **八、实验总结**  数据通路的设计直接影响到控制器的设计，同时也影响到数字系统的速度指标和成本，一般来说，处理速度快的数字系统，其独立传送信息的通路也较多，当然独立数据传送通路的数量增加势必提高控制器设计复杂度，因此，在满足速度指标的前提下，为使数字系统尽量简单，一般小型系统多采用单一总线结构，较大系统中可采用双总线结构或三总线结构。  数据通路运算器模块和存储器模块连接在一起，由于RAM的输入信号是三态的，因而可以将RAM连接到运算器的数据总线上。写入RAM的数据由运算器提供。RAM读出的数据可以到达运算工作器的暂存工作寄存器保存。各位控制信号仍由二进制开关输入，其信号含义与运算器组成实验和存储器实验相同。 |
| **九、教师评阅意见** |