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** | 2021年10月29 日 |
| **班 级** | 20计科4班 | **姓 名** | 孙紫铭 | **学 号** | B20200601133 |
| **实验成绩** |  | | | | |

|  |
| --- |
| **一、实验项目名称**  控制器实验 |
| **二、实验目的**   * 1. 状态机的工作原理与设计方法   2. .时序电路的组成原理和控制原理   3. 计算机中周期、节拍、脉冲之间的关系。 |
| **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．依据实验原理图与状态转换图，设计状态机依据实验原理图   * 按照下面实验内容所述进行波形仿真验证，并记录实验数据。 * 下载到实验箱，并进行实验。 * 根据不同的指令设计多周期控制器状态转换表 * (在主存中取一个数与R5中的数据进行加法运算，结果放到R5中)  |  |  |  |  | | --- | --- | --- | --- | | 当前状态 | 条件 | 动作 | 下一状态 | | A | Clk | K=>pc（由k给出主存的首地址） | B | | B | Clk | PC=>AR，PC+1=>PC | C | | C | Clk | M[AR] =>R4 | D | | D | Clk | PC=>AR，PC+1=>PC | E | | E | Clk | M[AR] =>dR2 | F | | F | Clk | R5 => DR1 | G | | g | Clk | DR1+DR2=>R5 | A |   4.用VHDL语言设计状态机实现多周期控制器  5实验板硬件验证 |
| **六、实验原理**  1.状态机的工作原理  状态机可归纳为4个要素，即现态、条件、动作、次态。这样的归纳，主要是出于对状态机的内在因果关系的考虑。"现态"和"条件"是因，"动作"和"次态"是果。  2.FSM的分类  限状态机有两种基本类型：米利（Mealy）机和摩尔（Moore）机，米利机的下一状态和输出取决于当前状态和当前输入；摩尔机的下一状态取决于当前状态和当前输入，但其输出仅取决于当前状态。这两类有限状态机的下一状态和输出都是由组合逻辑电路形成的。  **3．**状态机实现控制器的原理  状态机控制器的原理图如图3.1所示。状态机控制器是由状态机产生的控制器，控制器的各个状态是指一条指令从取指令、指令译码、执行指令的各个状态的集合 ，例如，指令ADD,0E,它的功能是ADD 0E 将0E 地址中的内容与R5内容相加，结果送R5，整个指令的执行过程就是：   1. 取指：PC=>AR，PC+1=>PC \_\_状态A 2. 译码：M[AR] =>IR \_\_状态B 3. **执行：**PC=>AR，PC+1=>PC \_\_状态C   M[AR] =>dR2 \_\_状态D  R5 =>dR1 \_\_状态E  DR1+DR2=>R5 \_\_状态F  A:pc=>ar  Pc+1=>pc  B:  M[AR] =>IR  C:pc=>ar  Pc+1=>pc  F:  dr1+DR2=>r5  E:  R5 =>dR1  D:  M[AR]=>dr2  图3.1 状态机控制器原理图 |
| **七、实验步骤**  1．依据实验原理图与状态转换图，设计状态机 。   1. 新建State Machine File(\*.smf)状态机文件，利用状态图输入方法   flie→new→state machine flie 打开编辑窗口如图  ]{YX50ZM()B3HQIH(FJ~_N9  State Machine Edtor 窗口  选择Tools→State Machine Wizard命令，弹出 如图所示的状态机创建对话框，选择create a new state machine design ,单击OK，进入下一个页面，选择reset为异步高电平有效，输出无寄存器。如图所示。单击Next，进入下一个页面如图所示。    创建状态机向导步骤1    创建状态机向导步骤2  在State 栏中设置状态名称（idle，st1,st2,s\_st1…….），在INPUT PORTS中clock，reset，qd,dp,tj 信号，在状态转换栏中设置状态转换，设置完成后，点击NEXT，进入page3，如图5.10，在Output Ports设置Output PortName为t1,t2,t3,t4; Output State中设置为Current clock cycle.Action conditions中按状态输出表b 设置。    创建状态机向导步骤3    创建状态机向导步骤4   1. General常规设置Reset模式为异步且高电平有效。 2. 利用States工具画状态圈。 3. Inputs栏添加信号：qd、dp、tj。 4. Outputs栏添加信号：t1、t2、t3、t4。 5. 利用Transitions转换工具画状态间线及条件。 6. 设置每个状态Action输出如下表。   表b 状态输出表   |  |  | | --- | --- | | 状态名 | 输出t[4..1] | | idle | 0000 | | st1 | 0001 | | st2、s\_st2 | 0010 | | st3、s\_st3 | 0100 | | st4、s\_st4 | 1000 |   2． 通过工具TOOLS→Generate HDL File生成VHDL文件并设置成顶层实体。   * 完成芯片设置与管脚设置。 * 按照下面实验内容所述进行波形仿真验证，结果如图所示，并记录实验数据。 * 下载到实验箱，并进行实验。   exp_timing  时序电路实验的时序仿真  3,根据不同的指令设计多周期控制器状态转换表  (在主存中取一个数与R5中的数据进行加法运算，结果放到R5中)   |  |  |  |  | | --- | --- | --- | --- | | 当前状态 | 条件 | 动作 | 下一状态 | | A | Clk | K=>pc（由k给出主存的首地址） | B | | B | Clk | PC=>AR，PC+1=>PC | C | | C | Clk | M[AR] =>R4 | D | | D | Clk | PC=>AR，PC+1=>PC | E | | E | Clk | M[AR] =>dR2 | F | | F | Clk | R5 => DR1 | g | | g | Clk | DR2-DR1=>R5 | a |   4.用VHDL语言设计状态机实现多周期控制器  LIBRARY ieee;  USE ieee.std\_logic\_1164.ALL;  USE ieee.std\_logic\_unsigned.ALL;  ENTITY fsm is  generic(N:integer:=50000; X:integer:=10 ; Y:integer:=100); -- pin21 100Khz N for clk input freq  port(  reset:in std\_logic;-------------------------------复位reset  clk:in std\_logic;--------------------------------100khz时钟信号  clkcnt:out std\_logic;--------------------------------2hz时钟信号  clkfresh:out std\_logic;--------------------------------10khz时钟信号  test:out std\_logic\_vector(7 downto 0);--------------数码管显示的状态值  mcode:out std\_logic\_vector(21 downto 0)--------------控制信号  );  END fsm;  Architecture RTL of fsm is  TYPE State\_type IS (A, B, C, D,E,F,G); -- 定义状态  SIGNAL State : State\_Type; -- 创建信号  SIGNAL stest: std\_logic\_vector(7 downto 0);--------------数码管显示的状态值  signal ctick,clk\_state,clk\_fresh:std\_logic;  BEGIN  -- 产生2hz clk1hz  PROCESS(clk) --产生2hz信号ctick  variable cnt1 : INTEGER RANGE 0 TO N -1;  BEGIN  IF clk='1' AND clk'event THEN  IF cnt1=N -1 THEN  cnt1:=0;  ELSE  IF cnt1<N/2 THEN  ctick<='1';  ELSE  ctick<='0';  END IF;  cnt1:=cnt1+1;  END IF;  END IF;  end process;  PROCESS(ctick) --产生1hz信号clk\_state  BEGIN  IF (reset ='0') THEN  clk\_state<='0';  ELSIF ctick='1' AND ctick'event THEN  clk\_state<=not clk\_state;  END IF;  end process;  -- 产生10khz clk clk\_fresh  PROCESS(clk) --产生10Khz信号  variable cnt1 : INTEGER RANGE 0 TO X -1;  BEGIN  IF rising\_edge(clk) THEN  IF cnt1=X -1 THEN  cnt1:=0;  ELSE  IF cnt1<X/2 THEN  clk\_fresh<='1';  ELSE  clk\_fresh<='0';  END IF;  cnt1:=cnt1+1;  END IF;  END IF;  end process;  PROCESS (clk\_state, reset)  BEGIN  --21sw-20r4-19r5-18alu-17pc|16r1-15r2-14r4-13r5-12ar|11m-10cn-9s38s27s16s0|5pcclr-4pcld-3pcen|2memen-1mw-0mr  If (reset = '0') THEN -- 复位reset，复位状态为A  State <= A;  mcode <="1111100000000000100100"; -- 什么都不做  stest <="00000000";  ELSIF rising\_edge(clk\_state) THEN  CASE State IS  WHEN A =>  mcode <="0111100000000000101100";-- k=>PC  stest <="00000001";  State <= B;  WHEN B =>  mcode <="1111000001000000111100"; -- pc=>AR pc+1=>pc  stest <="00000010";  State <= C;    WHEN C =>  mcode <="1111110000000000100101";-- M[AR]=> R4  stest <="00000011";  State <= D;  WHEN D =>  mcode <="1111000001000000111100";-- pc=>AR pc+1=>pc  stest <="00000100";  State <= E;    WHEN E =>  mcode <="1111101000000000100101";-- -- M[AR]=>dR2  stest <="00000101";  State <= F;  WHEN F=>  mcode <="1111010000000000100101";-- R5=>dR1  stest <="00000110";  State <= G;  WHEN G=>  mcode <="1110100000000110100100";-- R2-R1  stest <="00000110";  State <= A;    WHEN others =>  stest <="00000001";  State <= A;  END CASE;  END IF;    END PROCESS;  test <=stest;  clkcnt <= ctick;  clkfresh<=clk\_fresh;  END rtl;  5，顶层实体的RTL图    参考代码如下：  LIBRARY IEEE;  USE IEEE.STD\_LOGIC\_1164.ALL;  USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;  --LIBRARY work;  LIBRARY lpm;  USE lpm.lpm\_components.all;  ENTITY datapath IS  port(  reset:in std\_logic; --时钟  clk:in std\_logic; --时钟  k:in std\_logic\_vector(7 downto 0);--段显示控制（abcdefg）  seg:out std\_logic\_vector(7 downto 0);--段显示控制（abcdefg）  y:out std\_logic\_vector(7 downto 0); --数码管地址选择控制  dout:inout std\_logic\_vector(7 downto 0) --数码管地址选择控制  );  END datapath;  ARCHITECTURE Structural OF datapath IS  --ATTRIBUTE black\_box : BOOLEAN;  --nATTRIBUTE noopt : BOOLEAN;  --COMPONENT lpm\_ram\_io\_0  -- PORT(memenab : IN STD\_LOGIC;  -- inclock : IN STD\_LOGIC;  -- we : IN STD\_LOGIC;  -- outenab : IN STD\_LOGIC;  -- address : IN STD\_LOGIC\_VECTOR(7 DOWNTO 0)  --);  --END COMPONENT;  --ATTRIBUTE black\_box OF lpm\_ram\_io\_0: COMPONENT IS true;  --ATTRIBUTE noopt OF lpm\_ram\_io\_0: COMPONENT IS true;  COMPONENT lpm\_ram\_io0 IS  PORT  (  address : IN STD\_LOGIC\_VECTOR (7 DOWNTO 0);  we : IN STD\_LOGIC := '1';  inclock : IN STD\_LOGIC ;  outclock : IN STD\_LOGIC ;  outenab : IN STD\_LOGIC := '1';  dio : INOUT STD\_LOGIC\_VECTOR (7 DOWNTO 0)  );  END COMPONENT;  COMPONENT exp\_r\_alu is  port( clk,reset :IN STD\_LOGIC;  sw\_bus,r4\_bus,r5\_bus,alu\_bus,memen :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 COMPONENT;    COMPONENT sw\_pc\_ar IS  PORT( clk,pcclr,pcld,pcen,reset:IN STD\_LOGIC;  sw\_bus,pc\_bus,ldar :IN STD\_LOGIC;  inputd :IN STD\_LOGIC\_VECTOR(7 DOWNTO 0);  arout :OUT STD\_LOGIC\_VECTOR(7 DOWNTO 0);  d :INOUT STD\_LOGIC\_VECTOR(7 DOWNTO 0)  );  END COMPONENT;  COMPONENT display is  port(  clk:in std\_logic;--------------------------------时钟信号  in84:in std\_logic\_vector(7 downto 0);--------------数码管  in83:in std\_logic\_vector(7 downto 0);--------------数码管  in82:in std\_logic\_vector(7 downto 0);--------------数码管  in81:in std\_logic\_vector(7 downto 0);--------------数码管  seg:out std\_logic\_vector(7 downto 0);--------------数码管  sel:out std\_logic\_vector(7 downto 0) -------------段位  );  END COMPONENT;  COMPONENT fsm is  port(  reset:in std\_logic;--------------------------------时钟信号  clk:in std\_logic;--------------------------------时钟信号  clkcnt:out std\_logic;--------------------------------时钟信号  clkfresh:out std\_logic;--------------------------------时钟信号  test:out std\_logic\_vector(7 downto 0);  mcode:out std\_logic\_vector(21 downto 0)  );  END COMPONENT;  --21bus16ld11op5pc2m  --21sw-20r4-19r5-18alu-17pc|16r1-15r2-14r4-13r5-12ar|11m-10cn-9s38s27s16s0|5pcclr-4pcld-3pcen|2memen-1mw-0mr  SIGNAL smcode: std\_logic\_vector(21 downto 0);  SIGNAL clkcnt: std\_logic;  SIGNAL clkfresh: std\_logic;  SIGNAL sar: std\_logic\_vector(7 downto 0);  SIGNAL sdbus: std\_logic\_vector(7 downto 0);  SIGNAL smibus: std\_logic\_vector(7 downto 0);  SIGNAL test: std\_logic\_vector(7 downto 0);  SIGNAL clkon:STD\_LOGIC;  begin  PROCESS(clkcnt) --  BEGIN  IF (reset ='0') THEN  clkon<='1';  ELSIF clkcnt='1' AND clkcnt'event THEN  clkon<=not clkon;  END IF;  end process;  U0: fsm PORT MAP (  reset => reset,  clk =>clk,  clkcnt => clkcnt,  clkfresh => clkfresh,  test =>test,  mcode => smcode  );  U1: sw\_pc\_ar PORT MAP (  clk =>clkcnt,  reset =>reset,  pcclr =>smcode( 5) ,  pcld=>smcode( 4),  pcen=>smcode( 3),  sw\_bus=>smcode( 21),  pc\_bus=>smcode( 17),  ldar=>smcode( 12),  inputd =>k,  arout =>sar,  d=>sdbus  );  --U2 : lpm\_ram\_io\_0  --PORT MAP(memenab => '1',  -- inclock => clkcnt,  -- we => smcode( 1),  -- outenab => smcode( 0),  -- address => sar,  -- dio => smbus);  U2: lpm\_ram\_io0  PORT MAP  (  address =>sar,  we => smcode( 1),  inclock => clkcnt, --and (not clkon)  outclock => clkcnt, -- and (not clkon),  outenab => smcode( 0),  dio =>sdbus  );  U3: exp\_r\_alu PORT MAP (  clk =>clkcnt,  reset =>reset,  memen =>smcode( 0),  sw\_bus=>smcode( 21),  r4\_bus=>smcode( 20),  r5\_bus=>smcode( 19),  alu\_bus =>smcode( 18),  lddr1 =>smcode( 16),  lddr2=>smcode( 15),  ldr4=>smcode( 14),  ldr5 =>smcode( 13),  m =>smcode( 11) ,  cn =>smcode(10),  s =>smcode(9 downto 6),  k =>k,  d =>sdbus  );  U4: display PORT MAP (  clk => clkfresh,  in84 => test,  in83 => sar,  in82 => "00000000",  in81 => sdbus,  seg => seg,  sel => y  );  end Structural;  6.显示模块的参考代码  library ieee;  use ieee.std\_logic\_1164.all;  use ieee.std\_logic\_unsigned.all;  entity display is  port(  clk:in std\_logic;--------------------------------时钟信号  in84:in std\_logic\_vector(7 downto 0);--------------数码管  in83:in std\_logic\_vector(7 downto 0);--------------数码管  in82:in std\_logic\_vector(7 downto 0);--------------数码管  in81:in std\_logic\_vector(7 downto 0);--------------数码管  seg:out std\_logic\_vector(7 downto 0);--------------数码管  sel:out std\_logic\_vector(7 downto 0) -------------段位  );  end display;  architecture rtl of display is  signal cnt8:integer range 0 to 7;---------------计数  signal pos:integer range 0 to 7;---------------计数  signal tubein4:std\_logic\_vector(3 downto 0);--------------数码管  begin  -- 8进制计数器cnt8  process(clk) --  VARIABLE tmpcnt :integer range 0 to 7;  begin  if rising\_edge(clk) then  if (tmpcnt=7) then  tmpcnt:=0;  else  tmpcnt:=tmpcnt+1;  end if;  end if;  cnt8<=tmpcnt;  end process;  --  --  process(cnt8)------------------只是显示在某种状态下的段位显示，不参与计算  begin  case cnt8 is  when 7 => pos <=7;sel <="01111111";------------cnt是1时，q显示0,依次选通s  when 6 => pos <=6;sel <="10111111";  when 5 => pos <=5;sel <="11011111";  when 4 => pos <=4;sel <="11101111";  when 3 => pos <=3;sel <="11110111";  when 2 => pos <=2;sel <="11111011";  when 1 => pos <=1;sel <="11111101";  when 0 => pos <=0;sel <="11111110";  when others => null;  end case;  end process;  process(tubein4)------------------只是显示在某种状态下的段位显示，不参与计算  begin  case tubein4 is  when "0000" => seg<="11111100";  when "0001" => seg<="01100000";  when "0010" => seg<="11011010";  when "0011" => seg<="11110010";  when "0100" => seg<="01100110";  when "0101" => seg<="10110110";  when "0110" => seg<="10111110";  when "0111" => seg<="11100000";  when "1000" => seg<="11111110";  when "1001" => seg<="11110110";  when "1010" => seg<="11101110";  when "1011" => seg<="00111110";  when "1100" => seg<="10011100";  when "1101" => seg<="01111010";  when "1110" => seg<="10011110";  when "1111" => seg<="10001110";  when others => null;  end case;  end process;  process(pos)------------------只是显示在某种状态下的段位显示，不参与计算  begin  case pos is  when 0 => tubein4(3 downto 0)<=in81(3 downto 0);  when 1 => tubein4(3 downto 0)<=in81(7 downto 4);  when 2 => tubein4(3 downto 0)<=in82(3 downto 0);  when 3 => tubein4(3 downto 0)<=in82(7 downto 4);  when 4 => tubein4(3 downto 0)<=in83(3 downto 0);  when 5 => tubein4(3 downto 0)<=in83(7 downto 4);  when 6 => tubein4(3 downto 0)<=in84(3 downto 0);  when 7 => tubein4(3 downto 0)<=in84(7 downto 4);  when others => null;  end case;  end process;  end rtl; |
| IMG_20211124_220412_edit_103781698259163  IMG_20211124_220403_edit_103773886901872  IMG_20211124_220416_edit_103788344964891 |
| **七、实验总结**  通过该实验能更好了解控制器工作原理。我懂得了用进位控制运算器进行简单的算术运算，掌握了进位控制运算器的工作特性及使用方法。在做这个实验中，较顺利地完成了这个实验，但是对逻辑运算器实验的时间的把握不是很好和对实验的仪器仍不熟悉，从而导致进位控制运算器的实验要在很短的时间内完成，虽然时间仓促，但是总的来说，经过努力基本完成了实验的要求。  思考题   1. 时序电路实行了哪几种启停控制逻辑？   答：时序电路用启动，停机，单拍三种控制信号来控制T1-T4的发送，使原始节拍脉冲变成CPU真正需要的节拍信号T1-T4。   1. 举例说明机器周期、节拍、脉冲？   答：如本次实验中，机器周期就是CLK，节拍就是四个脉冲循环一次所用的时间，脉冲就是t1, t2, t3，t4单次触发用的时间。   1. 单步运行状态如何进入？用途是什么？   答：单步运行状态进入的方式是在状态机启动后，且不为停机状态时，使单拍信号有效，即dp=1。其用途是实现状态机产生一个节拍，即t1, t2, t3, t4单个脉冲循环且仅循环一次。   1. 控制器的功能？如何取指令、分析指令、执行指令？   答：功能：控制指令的正确执行；控制程序和数据的输入及结果的输出；异常情况和特殊请求的处理。先取指令，后分析指令，再执行指令。   1. 不同指令取指周期、译指周期、执行周期执行的操作都一样吗？有哪些是相同的，哪些是不同的？   答：不一定都一样。 |
| **八、教师评阅意见** |