**内存储器和串行接口实验**

**2012011344 胡钧 2013011697 李泽清 2014010475 蒋圣翊**

# 目的

1. 熟悉THINPAD教学计算机内存储器、串口的配置及与总线的连接方式；
2. 掌握教学机内存和串口UART的访问时序和方法；
3. 理解总线数据传输的基本原理。

# 实验环境

1. 硬件环境：安装有Windows7操作系统的PC，THINPAD教学计算机；
2. 软件环境：Xilinx ISE 14.7,串口调试精灵。

# 实验内容

1. RAM实验

使用教学计算机上的FPGA芯片，设计一个状态机和内存读写逻辑，完成对存储器RAM的访问，实现以下功能：

* 1. 写RAM1
  2. 读RAM1
  3. 写RAM2
  4. 读RAM2

1. 串口实验

使用教学计算机上的FPGA芯片，编写代码完成对教学机CPLD芯片中UART的访问，实现和PC上运行的串口调试精灵的通信，实现以下功能：

* 1. 接收串口调试精灵发送的数据并用LED显示；
  2. 通过UART向串口调试精灵发送数据并在PC上显示；
  3. 接收串口精灵发过来的数据，加1后发送回串口调试精灵并显示；

# 实验代码

RAM实验

----------------------------------------------------------------------------------

-- Company:

-- Engineer:

--

-- Create Date: 23:43:20 10/24/2016

-- Design Name:

-- Module Name: ram - Behavioral

-- Project Name:

-- Target Devices:

-- Tool versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

----------------------------------------------------------------------------------

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

USE ieee.std\_logic\_signed.all;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity ram is

Port ( clk : in STD\_LOGIC;

rst : in STD\_LOGIC;

InputSW : in STD\_LOGIC\_VECTOR (15 downto 0);

OutputL : out STD\_LOGIC\_VECTOR (15 downto 0);

Ram1Addr, Ram2Addr : out STD\_LOGIC\_VECTOR (17 downto 0);

Ram1Data, Ram2Data : inout STD\_LOGIC\_VECTOR (15 downto 0);

Ram1OE, Ram2OE : out STD\_LOGIC;

Ram1WE, Ram2WE : out STD\_LOGIC;

Ram1EN, Ram2EN : out STD\_LOGIC;

DYP1 : out STD\_LOGIC\_VECTOR (6 downto 0));

end ram;

architecture Behavioral of ram is

begin

process(clk, rst)

--variable DriAddr : std\_logic\_vector(17 downto 0) := "000000000000000000";

variable data : std\_logic\_vector(15 downto 0);

variable addr, addr0 : std\_logic\_vector(17 downto 0);

variable RW : std\_logic := '1'; --'1'表示读数据

variable i: integer :=0;

variable state: integer :=0;

begin

if(rst = '0') then

OutputL <= (others => '1');

Ram1Addr <= (others => '0');

Ram1Data <= (others => '0');

data := (others => '0');

addr := (others => '0');

Ram1OE <= '1';

Ram1WE <= '1';

Ram1EN <= '0';

Ram2Addr <= (others => '0');

Ram2Data <= (others => '0');

Ram2OE <= '1';

Ram2WE <= '1';

Ram2EN <= '0';

DYP1 <= (others => '0');

state:=0;

i:=0;

else

if(clk'event and clk = '1') then

case state is

when 0 => --读入地址

addr(15 downto 0):=InputSW;

-- addr0:=InputSW;

state:=1;

when 1 =>--读入数据

data:= InputSW;

state:=2;

when 2=> --RAM1进入写状态

Ram1WE <= '1';

state:=3;

when 3=> --写入

Ram1Addr <= addr;

Ram1Data <= data;

state:=4;

when 4 => --data，addr递增，进入下一循环

Ram1WE<= '0';

data:=data+1;

addr:=addr+1;

i:=i+1;

if(i < 10) then

state:=2;

else

i:=0;

state:=5;

end if;

when 5 => --Ram1进入读取状态

Ram1Data <= "ZZZZZZZZZZZZZZZZ";

Ram1WE <= '1';

Ram1OE <= '0';

state:=6;

Ram1Addr<=addr;

when 6 => --读取十个数据

OutputL<=Ram1Data;

i:=i+1;

addr:=addr-1;

Ram1Addr<=addr;

if(i = 10) then

state:=7;

i:=0;

end if;

when 7 => --开始写Ram2

Ram2WE<='1';

state:=8;

Ram1Addr<=addr;

when 8=> --读取Ram1数据，减去1后写入Ram2

Ram2Addr <= addr;

Ram2Data <= Ram1Data-'1';

state:=9;

when 9=> --读取地址+1

Ram2WE<='0';

addr:=addr+1;

i:=i+1;

if(i<10)then

state:=7;

i:=i+1;

else

state:=10;

i:=0;

end if;

when 10=> --开始读Ram2

Ram2Data <= "ZZZZZZZZZZZZZZZZ";

Ram2WE <= '1';

Ram2OE <= '0';

state:=11;

Ram2Addr<=addr;

when 11 => --依次读取RAM2中各数据

OutputL<=Ram2Data;

i:=i+1;

addr:=addr-1;

Ram1Addr<=addr;

if(i = 10) then

state:=12;

i:=0;

end if;

when others => null;

end case;

end if;

end if;

case state is –按照state不同亮灯提示

when 0=> DYP1 <= "0111111";

when 1=> DYP1 <= "0000110";

when 2=> DYP1 <= "1011011";

when 3=> DYP1 <= "1001111";

when 4=> DYP1 <= "1100110";

when 5=> DYP1 <= "1101101";

when 6=> DYP1 <= "1111101";

when 7=> DYP1 <= "0000111";

when 8=> DYP1 <= "1111111";

when 9=> DYP1 <= "1101111";

when 10 => DYP1 <= "1111111";

when 11 => DYP1 <= "0001111";

when others=> DYP1 <= "0000000";

end case;

end process;

end Behavioral;

串口实验代码

----------------------------------------------------------------------------------

-- Company:

-- Engineer:

--

-- Create Date: 16:45:36 10/25/2016

-- Design Name:

-- Module Name: CK - Behavioral

-- Project Name:

-- Target Devices:

-- Tool versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

----------------------------------------------------------------------------------

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

USE ieee.std\_logic\_signed.all;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity CK is

Port ( clk : in STD\_LOGIC;

rst : in STD\_LOGIC;

InputSW : in STD\_LOGIC\_VECTOR (7 downto 0);

OutputL : out STD\_LOGIC\_VECTOR (7 downto 0);

Ram1Data : inout STD\_LOGIC\_VECTOR (7 downto 0);

Ram1OE : out STD\_LOGIC;

Ram1WE : out STD\_LOGIC;

Ram1EN : out STD\_LOGIC;

data\_ready : in STD\_LOGIC;

rdn : inout STD\_LOGIC;

tbre : in STD\_LOGIC;

tsre : in STD\_LOGIC;

wrn : inout STD\_LOGIC);

end CK;

architecture Behavioral of CK is

begin

process(clk, rst)

variable xx: integer := 0;

variable state: integer :=0;

variable tmp: STD\_LOGIC\_VECTOR (7 downto 0):="00000000";

begin

if(rst = '0') then --初始化

wrn <= '1';

Ram1OE <= '1';

Ram1EN <= '1';

Ram1WE <= '1';

state := 0;

xx :=0;

else

if(clk'event and clk = '1') then

--分频

if(xx=1023)then

xx:=0;

case state is

when 0 => --初始化

Ram1OE <='1';

Ram1EN <='1';

Ram1WE <='1';

wrn <='1';

state:=state+1;

when 1 => --设置Ram1数据为高

rdn <='1';

Ram1Data <="ZZZZZZZZ";

state:=state+1;

when 2 => --如果dataready就读取，否则等待

if(data\_ready='1')then

rdn<='0';

state:=3;

else

state := 1;

end if;

when 3 => --读取一个数据，显示在led

OutputL <=Ram1Data;

tmp:=Ram1Data;

rdn <= '1';

wrn<='1';

state:=state+1;

when 4 => --将读取的数+1

tmp := tmp + 1;

state:=state+1;

when 5 => --写的初始化

wrn<='0';

Ram1data<=tmp;

OutputL<=tmp;

state:=state+1;

when 6 => --wrn置1

wrn<='1';

state:=state+1;

when 7 => --等待tbre为1即可发送

if (tbre = '1') then

state:=state+1;

end if;

when 8 => --等待tsre为1发送完成

if (tsre = '1') then

state:=0;

end if;

when others=>

end case;

else

xx:=xx+1;

end if;

end if;

end if;

end process;

end Behavioral;

实验中遇到的一些困难和解决方案：

尝试了较多的版本才找到了可用的串口驱动和调试工具，最终我使用的是serial port utility这一工具

在最初写串口实验的时候忘了进行分频，导致奇怪的结果出现，加上这一功能后得到解决

错误理解了实验要求，将读串口-写串口-先读再写理解为一项任务而非三项，在大实验中我们会进行调整

1. **思考题**
2. 静态存储器的读和写各有什么特点？

只有写信号，没有单独的读信号；

读SRAM时要提前准备好地址，并将数据线设置成高阻；SRAM时要提前准备好地址数据，然后将写信号拉低。RAM1通信时需要让串口不工作。

1. 什么是RAM芯片输出的高阻态？它的作用是什么？

输出端口足有高阻抗，使得芯片可以与其他芯片并联输出。

1. 本实验完成的是将RAM1和RAM2作为独立存储器单独进行访问的功能。如果希望将RAM1和RAM2作为一个统一的32位数据的存储器进行访问该如何进行？

状态机读入16位地址，同时接到RAM1和RAM2的地址总线，将两条数据线设置为高阻态，等到数据读出后拼接成32位输出。

1. 请总结教学计算机上的UART和普通串口芯片8251的异同点。

相同点：功能都是将将串行信号与并行信号进行转换，可连接RS232串口，；

不同点：教学机UART没有设计缓冲也不具有可编程性质；