计算机组成原理自主实验之

----------------------多重中断服务

小组成员：杨志飞

黄渝光

郭宏子龙

--------------------------目录--------------------------

一：实验总体完成情况--------------------------------------3

二：小组分工任务---------------------------------------------4

三：开发板的开关设定---------------------------------------5

四：实验代码的模块介绍------------------------------------6

五：实验总结-----------------------------------------------------7

附录：vhdl源代码-----------------------------------------------8-13

一：实验总体完成情况

实验模块介绍

设置屏蔽字---------------------------

-- Company:

-- Engineer:

--

-- Create Date: 23:10:34 06/01/2015

-- Design Name:

-- Module Name: Interrupt - 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\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity Interrupt is

Port(Ready : in STD\_LOGIC\_VECTOR(1 to 4);

maskset: in STD\_LOGIC\_VECTOR(1 to 4);

btn : in STD\_LOGIC\_VECTOR(0 to 4);

clk : in STD\_LOGIC;

Led : out STD\_LOGIC\_VECTOR(1 to 4);

ProgressLed : out STD\_LOGIC\_VECTOR(1 to 4);

seg : out STD\_LOGIC\_VECTOR(7 downto 1);

an : out STD\_LOGIC\_VECTOR(3 downto 0)

);

end Interrupt;

architecture Behavioral of Interrupt is

type DISPLAY is array(1 to 4) of STD\_LOGIC\_VECTOR(7 downto 1);

signal digit: DISPLAY := ("1111111", "1111111", "1111111", "1111111"); --Initially display nothing

signal MASK: STD\_LOGIC\_VECTOR(1 to 4) := "0000";

type MASKWORDS is array(0 to 4) of STD\_LOGIC\_VECTOR(1 to 4);

signal maski: MASKWORDS := ("0000", "1111", "0111", "0011", "0001");

signal js: std\_logic\_vector(26 downto 0) :="000000000000000000000000000";

signal clkcount, clkseg: STD\_LOGIC;

signal shift: STD\_LOGIC\_VECTOR(3 downto 0) :="0111";

signal INTR: std\_logic\_vector(1 to 4) := "0000";--涓柇璇锋眰

--signal INTP: std\_logic\_vector(1 to 4) := "0000";--涓柇鎺掗槦杈撳嚭

signal EINT: std\_logic;--寮€鍏充腑鏂 shared variable top: integer range 0 to 4:= 0; --Stack Top Pointer

signal sigtop: integer range 0 to 4 := 0;

type STACK4 is array(0 to 4) of integer range 0 to 4;

shared variable stackInt, stackTime : STACK4 := (0,0,0,0,0);

signal timeProgress: INTEGER range 0 to 4:=0;

constant int2digit: DISPLAY := ("1111001","0100100","0110000","0011001");

begin

--clock

process(clk)

begin

if(rising\_edge(clk)) then

js <= js + '1';

end if;

end process;

clkseg <= js(15);

clkcount <= js(26);

--set mask

process(maskset, btn, maski)

begin

if maskset = "0000" then

if btn(0) = '1' then

Led <= MASK;

elsif btn(1) = '1' then

Led <= maski(1);

elsif btn(2) = '1' then

Led <= maski(2);

elsif btn(3) = '1' then

Led <= maski(3);

elsif btn(4) = '1' then

Led <= maski(4);

else Led <= INTR;

end if;

else

Led <= maskset;

if btn(1) = '1' then

maski(1) <= maskset;

end if;

if btn(2) = '1' then

maski(2) <= maskset;

end if;

if btn(3) = '1' then

maski(3) <= maskset;

end if;

if btn(4) = '1' then

maski(4) <= maskset;

end if;

end if;

end process;

MASK <= maski(stackInt(top));

--Query, Queue and Scene Preserving

process(intr,eint,mask,maski,clkcount)

begin

if (rising\_edge(clkcount)) then

INTR <= Ready;

if (top > 0) then

stackTime(top) := stackTime(top) + 1;

end if;

if (stackTime(top) = 5) then

top := top - 1;

end if;

timeProgress <= stackTime(top);

if (EINT='1') then

if (INTR(1)='1' and MASK(1)='0') then

EINT <= '0';

--INTP(1) <= '1';

top := top + 1;

stackInt(top) := 1;

stackTime(top) := 0;

elsif (INTR(2)='1' and MASK(2)='0') then

EINT <= '0';

--INTP(2) <= '1';

top := top + 1;

stackInt(top) := 2;

stackTime(top) := 0;

elsif (INTR(3)='1' and MASK(3)='0') then

EINT <= '0';

--INTP(3) <= '1';

top := top + 1;

stackInt(top) := 3;

stackTime(top) := 0;

elsif (INTR(4)='1' and MASK(4)='0') then

EINT <= '0';

--INTP(4) <= '1';

top := top + 1;

stackInt(top) := 4;

stackTime(top) := 0;

end if;

else

EINT <= '1';

end if;

end if;

sigtop <= top;

end process;

--display

process(clkseg)

begin

if(rising\_edge(clkseg)) then

shift(1)<=shift(0);

shift(2)<=shift(1);

shift(3)<=shift(2);

shift(0)<=shift(3);

an <= shift;

case shift is

when "0111" => seg <= digit(1);

when "1011" => seg <= digit(2);

when "1101" => seg <= digit(3);

when others => seg <= digit(4);

end case;

end if;

end process;

process(timeProgress)

begin

if (timeProgress=0) then

ProgressLed <= "0000";

elsif (timeProgress=1) then

ProgressLed <= "1000";

elsif (timeProgress=2) then

ProgressLed <= "1100";

elsif (timeProgress=3) then

ProgressLed <= "1110";

else

ProgressLed <= "1111";

end if;

end process;

process(sigtop, clkcount)

begin

digit(1) <= "1111111";

digit(2) <= "1111111";

digit(3) <= "1111111";

digit(4) <= "1111111";

if sigtop = 1 then

if clkcount = '1' then

digit(1) <= int2digit(stackInt(1));

else digit(1) <= "1111111";

end if;

elsif sigtop = 2 then

digit(1) <= int2digit(stackInt(1));

if clkcount = '1' then

digit(2) <= int2digit(stackInt(2));

else digit(2) <= "1111111";

end if;

elsif sigtop = 3 then

digit(1) <= int2digit(stackInt(1));

digit(2) <= int2digit(stackInt(2));

if clkcount = '1' then

digit(3) <= int2digit(stackInt(3));

else digit(3) <= "1111111";

end if;

elsif sigtop = 4 then

digit(1) <= int2digit(stackInt(1));

digit(2) <= int2digit(stackInt(2));

digit(3) <= int2digit(stackInt(3));

if clkcount = '1' then

digit(4) <= int2digit(stackInt(4));

else digit(4) <= "1111111";

end if;

end if;

end process;

end Behavioral;

中断请求

排队器与堆栈模块

时钟降频

服务进度条

中断响应

七段数码管的显示

结束

实现的功能：

1. 总共设置了四个中断源，并且用户能够为这四个中断源主动设置屏蔽字
2. 每个中断源的响应执行时间为4秒钟
3. 中断源请求中断能够被执行，如果优先级更高的中断源请求中断，那么原先的中断被打断进入栈中，从而执行优先级高的中断程序，如果优先级低的中断源在此刻请求中断，那么先将原来的中断程序执行完毕后再执行该中断。
4. 中断源的执行过程默认为在七段数码管的显示时间过程。
5. 栈的内容可以在七段数码管中显示出来
6. 初始化的优先级为1<2<3<4
7. 中断源的屏蔽字能够显示出来（开关的亮灭）

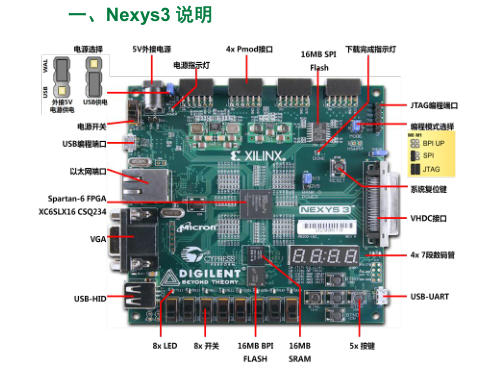
二：小组分工任务

杨志飞：主要扮演领导的角色，决策了本实验的功能的设计，并且编写了VHDL的部分代码，并参与了调试工作。

黄渝光：主要扮演执行的角色。设计了本次实验的部分功能，并且编写了VHDL的部分代码，参与调试工作。

郭宏子龙：主要扮演完善和总结的角色。参与了本次试验的功能设计，并且指出了一些不足之处，完成实验报告和总结。

三：开发板的开关设定



用到的内容：

8XLED，8X开关，5X按键，4\*7段数码管

1-4x开关：代表了四个中断源

5-8x开关：用来设置四个中断源的屏蔽字

8xLED:根据8x开关的上下而亮灭，用来显示

中x按键：用来确定屏蔽字设置完成

上x按键：显示中断源1的屏蔽字

下x按键：显示中断源2的屏蔽字

左x按键：显示中断源3的屏蔽字

右x按键：显示中断源4的屏蔽字

四：实验代码的模块介绍

1. 设置屏蔽字模块：

如果开始时用户想自己设置屏蔽字而不是想使用初始化的屏蔽字可以在执行中断之前自己设置屏蔽字，此设置会记录在开发板中知道用户进一步修改。

1. 中断请求模块：

接收8X开关中的左侧四个中断源的请求，并根据优先级来判断是响应还是放入栈中。

1. 排队堆栈模块：

在设置完屏蔽字后，开始模拟各个中断源请求中断响应的部分，如果出现更高一级的中断，则将现有的中断压栈并保护现场，保存时间，执行新的中断，执行过后再返回原来的中断并且继续上次的时间执行。如果出现低级的中断，则在此中断结束或执行。

1. 中断响应

执行排队堆栈模块的的栈顶元素的中断请求，执行四秒钟。

可以被高级的中断打断。

1. 七段数码管显示模块：

将排队堆栈中的内容在七段数码管中显示出来，不光显示出正在执行的中断，而且能够显示出所有栈中的元素，执行的元素每一秒闪烁一次

1. 服务进度条与时钟降频模块

修改Clk的电平变化来使中断变为四秒钟

通过控制8XLED灯的右面四个灯的亮灭来显示中断执行的时间刚开始是右面四个灯全亮，之后每过一秒左侧的灯熄灭，执行完毕后所有的灯全部熄灭

五：设计、调试、波形、下载过程中遇到的问题及解决方法

杨志飞：由于几乎是第一次接触VHDL，我们在程序编写的过程中遇到了很多困难。比如多个process中不能对同一个信号赋值的问题，为解决这个问题我们重新设计了相关模块的工作流程，实现了一个信号只在一个process中被赋值。还有signal信号不能被立即更新的问题，我们在中断的管理中使用了栈，出入栈时需要对栈顶指针立即赋值，通过查阅资料，我们采用了shared variable变量来表示栈顶，解决了这个问题。

我们遇到的最后一个bug是INTR的维持只在第一次发中断时有效，如果后面再发就会无法维持。经过排查，我们发现这是因为回到主程序后INTA没有恢复成0000造成的。让我们十分欣慰的是，添加代码恢复0000后，这个bug就消失了。

六：实验总结

通过本次实验，我们小组成员初步了解往开发板中植入程序的一系列过程，首先需要对开发板有一定的了解，并且需要熟悉ISE开发环境，更要掌握VHDL的编程语言。

本次实验我们首先学习了中断的相关概念，对实验的功能进行了设计，然后学习了此开发板的接口和VHDL语言，并在ISE中进行编程来实现之前设计的功能，最后在开发板上进行演示，做最后的调试工作。

我们小组也通过这次实验感受到了小组合作完成实验的魅力，只有分配好任务，并且充分发挥每个人的智慧才能做到事半功倍，而且我们也对硬件方面产生了浓厚的兴趣，所以我们会在接下来的课程中继续钻研，学到更多的本领。

附录：VHDL源代码

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

-- Company:

-- Engineer:

--

-- Create Date: 23:10:34 06/01/2015

-- Design Name:

-- Module Name: Interrupt - 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\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity Interrupt is

Port(Ready : in STD\_LOGIC\_VECTOR(1 to 4);

maskset: in STD\_LOGIC\_VECTOR(1 to 4);

btn : in STD\_LOGIC\_VECTOR(0 to 4);

clk : in STD\_LOGIC;

Led : out STD\_LOGIC\_VECTOR(1 to 4);

ProgressLed : out STD\_LOGIC\_VECTOR(1 to 4);

seg : out STD\_LOGIC\_VECTOR(8 downto 1);

an : out STD\_LOGIC\_VECTOR(3 downto 0)

);

end Interrupt;

architecture Behavioral of Interrupt is

type DISPLAY is array(1 to 4) of STD\_LOGIC\_VECTOR(8 downto 1);

signal digit: DISPLAY := ("11111111", "11111111", "11111111", "11111111"); --Initially display nothing

signal MASK: STD\_LOGIC\_VECTOR(1 to 4) := "0000";

type MASKWORDS is array(0 to 4) of STD\_LOGIC\_VECTOR(1 to 4);

signal maski: MASKWORDS := ("0000", "1111", "0111", "0011", "0001");

signal js: std\_logic\_vector(26 downto 0) :="000000000000000000000000000";

signal clkcount, clkseg: STD\_LOGIC;

signal shift: STD\_LOGIC\_VECTOR(3 downto 0) :="0111";

signal INTR, INTA: std\_logic\_vector(1 to 4) := "0000";--中断请求

--signal INTP: std\_logic\_vector(1 to 4) := "0000";--中断排队输出

signal EINT: std\_logic;--开关中断

shared variable top: integer range 0 to 4:= 0; --Stack Top Pointer

signal sigtop: integer range 0 to 4 := 0;

type STACK4 is array(0 to 4) of integer range 0 to 4;

shared variable stackInt, stackTime : STACK4 := (0,0,0,0,0);

signal timeProgress: INTEGER range 0 to 4:=0;

constant int2digit: DISPLAY := ("11111001","10100100","10110000","10011001");

begin

--clock

process(clk)

begin

if(rising\_edge(clk)) then

js <= js + '1';

end if;

end process;

clkseg <= js(15);

clkcount <= js(26);

--set mask

process(maskset, btn, maski)

begin

if maskset = "0000" then

if btn(0) = '1' then

Led <= MASK;

elsif btn(1) = '1' then

Led <= maski(1);

elsif btn(2) = '1' then

Led <= maski(2);

elsif btn(3) = '1' then

Led <= maski(3);

elsif btn(4) = '1' then

Led <= maski(4);

else Led <= INTR;

end if;

else

Led <= maskset;

if btn(1) = '1' then

maski(1) <= maskset;

end if;

if btn(2) = '1' then

maski(2) <= maskset;

end if;

if btn(3) = '1' then

maski(3) <= maskset;

end if;

if btn(4) = '1' then

maski(4) <= maskset;

end if;

end if;

end process;

MASK <= maski(stackInt(top));

--Query, Queue and Scene Preserving

process(intr,eint,mask,maski,clkcount)

begin

if (rising\_edge(clkcount)) then

INTR <= (INTR and not INTA) or Ready;

if (top = 0) then

INTA <= "0000";

end if;

if (top > 0) then

stackTime(top) := stackTime(top) + 1;

end if;

if (stackTime(top) = 5) then

INTA(sigtop) <= '0';

top := top - 1;

end if;

timeProgress <= stackTime(top);

if (EINT='1' and top < 4) then

if (((INTR(1) and not INTA(1)) = '1' or Ready(1) = '1') and MASK(1)='0') then

EINT <= '0';

INTA(1) <= '1';

top := top + 1;

stackInt(top) := 1;

stackTime(top) := 0;

elsif (((INTR(2) and not INTA(2)) = '1' or Ready(2) = '1') and MASK(2)='0') then

EINT <= '0';

INTA(2) <= '1';

top := top + 1;

stackInt(top) := 2;

stackTime(top) := 0;

elsif (((INTR(3) and not INTA(3)) = '1' or Ready(3) = '1') and MASK(3)='0') then

EINT <= '0';

INTA(3) <= '1';

top := top + 1;

stackInt(top) := 3;

stackTime(top) := 0;

elsif (((INTR(4) and not INTA(4)) = '1' or Ready(4) = '1') and MASK(4)='0') then

EINT <= '0';

INTA(4) <= '1';

top := top + 1;

stackInt(top) := 4;

stackTime(top) := 0;

end if;

else

EINT <= '1';

end if;

end if;

sigtop <= top;

end process;

--display

process(clkseg)

begin

if(rising\_edge(clkseg)) then

shift(1)<=shift(0);

shift(2)<=shift(1);

shift(3)<=shift(2);

shift(0)<=shift(3);

an <= shift;

case shift is

when "0111" => seg <= digit(1);

when "1011" => seg <= digit(2);

when "1101" => seg <= digit(3);

when others => seg <= digit(4);

end case;

end if;

end process;

process(timeProgress)

begin

if (timeProgress=0) then

ProgressLed <= "0000";

elsif (timeProgress=1) then

ProgressLed <= "1000";

elsif (timeProgress=2) then

ProgressLed <= "1100";

elsif (timeProgress=3) then

ProgressLed <= "1110";

else

ProgressLed <= "1111";

end if;

end process;

process(sigtop, clkcount)

begin

digit(1) <= "11111111";

digit(2) <= "11111111";

digit(3) <= "11111111";

digit(4) <= "11111111";

if sigtop = 1 then

if clkcount = '1' then

digit(1) <= int2digit(stackInt(1));

else digit(1) <= "11111111";

end if;

elsif sigtop = 2 then

digit(1) <= int2digit(stackInt(1));

if clkcount = '1' then

digit(2) <= int2digit(stackInt(2));

else digit(2) <= "11111111";

end if;

elsif sigtop = 3 then

digit(1) <= int2digit(stackInt(1));

digit(2) <= int2digit(stackInt(2));

if clkcount = '1' then

digit(3) <= int2digit(stackInt(3));

else digit(3) <= "11111111";

end if;

elsif sigtop = 4 then

digit(1) <= int2digit(stackInt(1));

digit(2) <= int2digit(stackInt(2));

digit(3) <= int2digit(stackInt(3));

if clkcount = '1' then

digit(4) <= int2digit(stackInt(4));

else digit(4) <= "11111111";

end if;

end if;

end process;

end Behavioral;