**“电子设计自动化(EDA)”实验报告(八)**

姓名 刘宗鑫 学号 201710413046 日期 2020.06.28

**一、实验名称：**带闹钟的时分秒可调的数字钟

**二、实验内容和实验目的**

**实验内容：**

1．设计一个带闹钟功能的数字钟,用DE2开发板上的6个数码管(HEX5~HEX0)用来显示时、分、秒。Adjust 按键用来调整时、分、秒。Mode按键用来调整模式，模式的具体含义及数字钟功能要求如表1所示。闹钟设置的时间到了之后，LEDG[7]绿灯亮。

表1 数字钟功能

|  |  |
| --- | --- |
| Mode键 | 功能 |
| 按一下mode键 | 此时可调整“时”，每按一下adjust键，增加1小时，增加到23小时，再按一下adjust键，则变为00小时。调整过程中，数码管HEX5，HEX4闪烁，提示用户此时正在调整“小时”。 |
| 再按一下mode键 | 此时可调整“分”，每按一下adjust键，增加1分钟，增加到59分，再按一下adjust键，则变为00分。调整过程中，数码管HEX3，HEX2闪烁，提示用户此时正在调整“分钟”。 |
| 再按一下mode键 | 此时可调整“秒”，每按一下adjust键，增加1秒，增加到59秒，再按一下adjust键，则变为00秒。调整过程中，数码管HEX1，HEX0闪烁，提示用户此时正在调整“秒”。 |
| 再按一下mode键 | 此时可调整闹钟的“时”，每按一下adjust键，增加1小时，增加到23小时，再按一下adjust键，则变为00小时。调整过程中，数码管HEX5，HEX4用来显示闹钟的“小时”并闪烁，提示用户此时正在调整闹钟“小时”。HEX1，HEX0两个数码管（显示秒）不亮。 |
| 再按一下mode键 | 此时可调整闹钟的“分”，每按一下adjust键，增加1分钟，增加到59分，再按一下adjust键，则变为00分。调整过程中，数码管HEX3，HEX2用来显示闹钟的“分钟”并闪烁，提示用户此时正在调整闹钟“分钟”。HEX1，HEX0两个数码管（显示秒）不亮。 |
| 再按一下mode键 | 回到正常计时和显示状态 |

2．用VHDL语言设计一个奇偶校验器，使用variable.

3. 用VHDL设计一个数据选通器，分别用三种方法设计，用VHDL simple signal assignment, **conditional signal assignment**,和**selected signal assignment**语句设计。

4. 用VHDL状态机设计一个序列信号检测器

5. 用VHDL设计一个8bit双向总线接口。

**实验目的：**

学会分析和设计一个功能较复杂的综合电路，数字钟，带时分秒调整,闹钟及显示，及学会查看编译报告。

进一步掌握并熟练使用VHDL的语句。

**三、实验步骤、实现方法(或设计思想)及实验结果**

**实验步骤：**

1. 分析并理解实验设计的目的和要求。
2. 按照要求编写代码。
3. 建立波形文件。
4. 设置仿真间隔。
5. 进行仿真。

**实现方法：**

1. 设计一个带闹钟功能的数字钟

|  |  |
| --- | --- |
| mode\_adjust | library ieee;  use ieee.std\_logic\_1164.all;  use ieee.std\_logic\_unsigned.all;  entity mode\_adjust is      port(          clk, adjust, mode: in std\_logic;          clk1hz, blink2hz: buffer std\_logic;          min\_h, min\_l : buffer std\_logic\_vector( 3 downto 0 );          hour\_h, hour\_l : buffer std\_logic\_vector( 3 downto 0 );          sec\_h, sec\_l : buffer std\_logic\_vector( 3 downto 0 );          min\_h\_seg, min\_l\_seg : out std\_logic\_vector( 6 downto 0 );          hour\_h\_seg, hour\_l\_seg : out std\_logic\_vector( 6 downto 0 );          sec\_h\_seg, sec\_l\_seg : out std\_logic\_vector( 6 downto 0 );      reset: in std\_logic;      clk\_min\_h, clk\_min\_l : buffer std\_logic\_vector( 3 downto 0 );      clk\_hour\_h, clk\_hour\_l : buffer std\_logic\_vector( 3 downto 0 );      alarm : out std\_logic      );  end mode\_adjust;  architecture arc of mode\_adjust is    component fdiv    port(          clk: in std\_logic;          clk1hz, blink2hz: out std\_logic      );      end component;      component counter24      port(        clk, en: in std\_logic;        cout: out std\_logic;        hh, hl: buffer std\_logic\_vector( 3 downto 0 );        reset: in std\_logic      );    end component;    component counter60      port(        clk, en: in std\_logic;        cout: out std\_logic;        hh, hl: buffer std\_logic\_vector( 3 downto 0 );        reset: in std\_logic      );    end component;    component segment4to7      port(      data:in std\_logic\_vector( 3 downto 0 );      output :out std\_logic\_vector(6 downto 0));    end component;    signal en\_h, en\_m, ens, enh, enm, cout\_hour, cout\_min, cout\_sec: std\_logic;    signal min\_h\_out, min\_l\_out, hour\_h\_out, hour\_l\_out,sec\_h\_out, sec\_l\_out: std\_logic\_vector( 3 downto 0 );    signal blink\_min, blink\_hour, blink\_sec: std\_logic;    signal clk\_hour, clk\_min, clk\_sec: std\_logic;    type state\_type is ( timing, set\_hour, set\_min, set\_sec, set\_clk\_hour, set\_clk\_min );    signal cur\_state: state\_type;      signal clk\_en\_h, clk\_en\_m : std\_logic;  begin    clock: fdiv port map( clk, clk1hz, blink2hz );      process( mode )    begin      if mode'event and mode = '1' then        case cur\_state is          when timing =>            cur\_state <= set\_hour;          when set\_hour =>            cur\_state <= set\_min;          when set\_min =>            cur\_state <= set\_sec;          when set\_sec =>            cur\_state <= set\_clk\_hour;          when set\_clk\_hour =>            cur\_state <= set\_clk\_min;          when set\_clk\_min =>            cur\_state <= timing;        end case;      end if;    end process;      enh <= '1' when cur\_state = set\_hour else '0';    enm <= '1' when cur\_state = set\_min else '0';    ens <= '1' when cur\_state = set\_sec or cur\_state = timing else '0';    clk\_en\_h <= '1' when cur\_state = set\_clk\_hour else '0';    clk\_en\_m <= '1' when cur\_state = set\_clk\_min else '0';      clk\_hour <= adjust when cur\_state = set\_hour else clk1hz;    clk\_min <= adjust when cur\_state = set\_min else clk1hz;    clk\_sec <= adjust when cur\_state = set\_sec else clk1hz;      hour: counter24 port map( clk\_hour, cout\_min or enh, cout\_hour, hour\_h, hour\_l, reset );    min: counter60 port map( clk\_min, cout\_sec or enm, cout\_min, min\_h, min\_l,reset );    sec: counter60 port map( clk\_sec, ens, cout\_sec, sec\_h, sec\_l, reset );    clock\_hour: counter24 port map( adjust, en => clk\_en\_h, hh => clk\_hour\_h, hl => clk\_hour\_l, reset => '1' );    clock\_min: counter24 port map( adjust, en => clk\_en\_m, hh => clk\_min\_h, hl => clk\_min\_l, reset => '1' );      blink\_hour <= '1' when cur\_state = set\_hour or cur\_state = set\_clk\_hour else '0';    blink\_min <= '1' when cur\_state = set\_min or cur\_state = set\_clk\_min  else '0';    blink\_sec <= '1' when cur\_state = set\_sec else '0';      hour\_h\_out <= "1111" when ( (blink2hz = '0' and blink\_hour = '1') or cur\_state = set\_clk\_min ) else            clk\_hour\_h when cur\_state = set\_clk\_hour else            hour\_h;    hour\_l\_out <= "1111" when ( (blink2hz = '0' and blink\_hour = '1') or cur\_state = set\_clk\_min ) else            clk\_hour\_l when cur\_state = set\_clk\_hour else            hour\_l;    min\_h\_out <= "1111" when ( (blink2hz = '0' and blink\_min = '1') or cur\_state = set\_clk\_hour ) else           clk\_min\_h when cur\_state = set\_clk\_min else           min\_h;    min\_l\_out <= "1111" when ( (blink2hz = '0' and blink\_min = '1') or cur\_state = set\_clk\_hour ) else           clk\_min\_h when cur\_state = set\_clk\_min else           min\_l;    sec\_h\_out <= "1111" when (blink2hz = '0' and blink\_sec = '1') or cur\_state = set\_clk\_min or cur\_state = set\_clk\_hour else           sec\_h;    sec\_l\_out <= "1111" when (blink2hz = '0' and blink\_sec = '1') or cur\_state = set\_clk\_min or cur\_state = set\_clk\_hour else           sec\_l;      alarm <= '1' when clk\_hour\_h = hour\_h and clk\_hour\_l = hour\_l and clk\_min\_h = min\_h and clk\_min\_l = min\_l else '0';      hourhout: segment4to7 port map( hour\_h\_out, hour\_h\_seg );    hourlout: segment4to7 port map( hour\_l\_out, hour\_l\_seg );    minhout: segment4to7 port map( min\_h\_out, min\_h\_seg );    minlout: segment4to7 port map( min\_l\_out, min\_l\_seg );    sechout: segment4to7 port map( sec\_h\_out, sec\_h\_seg );    seclout: segment4to7 port map( sec\_l\_out, sec\_l\_seg );    end arc; |
| fdiv | library ieee;  use ieee.std\_logic\_1164.all;  use ieee.std\_logic\_unsigned.all;  entity fdiv is      port(          clk: in std\_logic;          clk1hz, blink2hz: out std\_logic      );  end fdiv;  architecture arc of fdiv is    signal cnt : integer range 0 to 50;  begin    process(clk)    begin      if clk'event and clk = '1' then        cnt <= cnt + 1;        if 0 <= cnt and cnt <5 then          blink2hz <= '1';        elsif 10 <= cnt and cnt < 15 then          blink2hz <= '1';        else          blink2hz <= '0';        end if;          if cnt = 19 then          clk1hz <= '1';          cnt <= 0;        else          clk1hz <= '0';        end if;      end if;    end process;  end arc; |
| counter24 | library ieee;  use ieee.std\_logic\_1164.all;  use ieee.std\_logic\_unsigned.all;  entity counter24 is      port(          clk, en: in std\_logic;          cout: out std\_logic;          hh, hl: buffer std\_logic\_vector( 3 downto 0 );          reset: in std\_logic      );  end counter24;  architecture arc of counter24 is  begin      process( clk )      begin      if reset = '0' then        hh <= "0000";        hl <= "0000";          elsif clk'event and clk = '1' then              if en = '1' then                  if hh = "0010" and hl = "0011" then            hh <= "0000";            hl <= "0000";                  else                      hl <= hl + "0001";                      if hl = "1001" then              hl <= "0000";              hh <= hh + "0001";            end if;                  end if;              end if;          end if;      end process;      cout <= '1' when (hh = "0010" and hl = "0011") else '0';  end arc; |
| counter60 | library ieee;  use ieee.std\_logic\_1164.all;  use ieee.std\_logic\_unsigned.all;  entity counter60 is      port(          clk, en: in std\_logic;          cout: out std\_logic;          hh, hl: buffer std\_logic\_vector( 3 downto 0 );          reset: in std\_logic      );  end counter60;  architecture arc of counter60 is  begin      process( clk )      begin        if reset = '0' then        hh <= "0000";        hl <= "0000";          elsif clk'event and clk = '1' then              if en = '1' then                  if hh = "0101" and hl = "1001" then            hh <= "0000";            hl <= "0000";                  else                      hl <= hl + "0001";                      if hl = "1001" then              hl <= "0000";              hh <= hh + "0001";            end if;                  end if;              end if;          end if;      end process;      cout <= '1' when (hh = "0101" and hl = "1001") else '0';  end arc; |
| Segment4to7 | library ieee;  use ieee.std\_logic\_1164.all;  entity segment4to7 is  port(       data:in std\_logic\_vector( 3 downto 0 );       output :out std\_logic\_vector(6 downto 0));  end segment4to7;  architecture rtc of segment4to7 is  begin    process(data)    begin      output <= "1111111";      if data = "0000" then        output <= "0000001";      elsif data = "0001" then        output <= "1001111";      elsif data = "0010" then        output <= "0010010";      elsif data = "0011" then        output <= "0000110";      elsif data = "0100" then        output <= "1001100";      elsif data = "0101" then        output <= "0100101";      elsif data = "0110" then        output <= "0100000";      elsif data = "0111" then        output <= "0111001";      elsif data = "1000" then        output <= "0000000";      elsif data = "1001" then        output <= "0001100";      else        output <= "1111111";      end if;    end process;  end rtc; |

思想：

* 在上次实验的基础上进行改进。
  + 增加reset。
  + 增加时钟，时钟的控制逻辑和计数器几乎一致。
  + 增加两个新状态，分别表示设置闹钟的时和分。

1. 用VHDL设计一个数据选通器，分别用三种方法设计

|  |
| --- |
| library ieee;  use ieee.std\_logic\_1164.all;  use ieee.std\_logic\_unsigned.all;  entity strobe is      port(      a, b, c: in std\_logic\_vector( 3 downto 0 );      sel : in std\_logic\_vector( 1 downto 0 );      y : out std\_logic\_vector( 3 downto 0 )      );  end strobe;  architecture arc1 of strobe is  begin    process(sel)    begin      if sel = "00" then        y <= a;      elsif sel = "01" then        y <= b;      elsif sel = "10" then        y <= c;      else        y <= "ZZZZ";      end if;    end process;  end arc1;  architecture arc2 of strobe is  begin    y <= a when sel = "00" else       b when sel = "01" else       c when sel = "10" else       "ZZZZ";  end arc2;  architecture arc3 of strobe is  begin    with sel select y <= a when "00",               b when "01",               c when "10",               "ZZZZ" when "11";  end arc3;  configuration cfg of strobe is    for arc3    end for;  end cfg; |

思想：

* 利用三种语句即可，当状态分别为00，01，11选通a，b，c。否则，则被赋予高阻态。

1. 用VHDL状态机设计一个序列信号检测器

|  |
| --- |
| library ieee;  use ieee.std\_logic\_1164.all;  use ieee.std\_logic\_unsigned.all;  entity seq\_check is  port(    a,clk:in std\_logic;    y:out std\_logic;    sm: out std\_logic\_vector( 3 downto 0 )      );  end entity;  architecture arc of seq\_check is  type state\_type is( s0,s1,s2,s3,s4,s5,s6,s7 );  signal state:state\_type;  begin    process(clk)    begin      if clk' event and clk='1' then        y <= '0';        case state is          when s0=>            if a='1' then              sm <= "0001";              state<=s1;            else              sm <= "0000";              state<=s0;            end if;          when s1=>            if a='1' then              sm <= "0010";              state<=s2;            else              sm <= "0000";              state<=s0;            end if;          when s2=>            if a='0' then              sm <= "0011";              state<=s3;            else              sm <= "0010";              state<=s2;            end if;          when s3=>            if a='0' then              sm <= "0100";              state<=s4;            else              sm <= "0001";              state<=s1;            end if;          when s4=>            if a='1' then              sm <= "0101";              state<=s5;            else              sm <= "0000";              state <= s0;            end if;          when s5=>            if a='0' then              sm <= "0110";              state<=s6;            else              sm <= "0010";              state<=s2;            end if;          when s6=>            if a='1' then              sm <= "0111";              state<=s7;              y<='1';            else              sm <= "0000";              state<=s0;              y<='0';            end if;          when s7=>            if a = '1' then              sm <= "0001";              state <= s1;            else              sm <= "0000";              state <= s0;              y<='0';            end if;        end case;      end if;    end process;  end arc; |

列出状态转移图，按照状态转移图进行转移即可。

**实验结果：**

|  |  |
| --- | --- |
| 时钟 |  |
| 序列检测 |  |
| 选通器 |  |

可见时钟之中，可以设置各个状态，并且到达规定时间后，alarm信号也会发生变化。

序列信号检测器可以检测到规定序列。

选通器也可以正常工作。

**四、总结**

* 使用configuration语句可以选择不同的architecture。
* 设计状态转移图时一定要考虑仔细。
* 对于选择语句有三种不同的方式。

**五、思考题**

整个Project 编译成功后，数字钟运行正确后，查看compilation report, 问，

a) 芯片EP2C35F672C6内部共有多少资源，换句话说，共有多少逻辑单元(Logic element)? 多少个寄存器(register)? 数字钟用了多少逻辑单元(Logic element)? 用了总资源的百分之几？数字钟用了多少寄存器(register)? 用了总资源的百分之几？

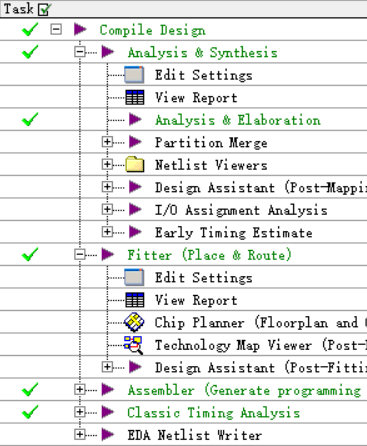
**总共33216个逻辑单元，数字钟用了271个逻辑单元，占总资源的百分之一。**

**共有33216个逻辑单元可以用作寄存器总共用了54个寄存器，占用了总资源的百分之一。**

其中counter24、 counter60、fdivider、mode\_adjust 模块，各用了多少寄存器(regjister)? 可以通过Tools->Netlist viewers-> RTL viewer ,查看编译器根据VHDL文件分析综合后，合成的逻辑框图，双击各个子框图 (例如counter60模块) 数一数用了多少寄存器。然后统计各个模块所用的寄存器的总和，是否与compilation report统计的一样。

**是一样的。**

2） Project编译过程中，编译器做了哪几步重要的工作？分别是什么？可以通过仔细查看下面图3中的编译任务(Tasks)看出来，列表中列出了编译的重要步骤。

* Compile Design 编译设计出来的电路或者代码
  + Analysis & Synthesis 分析和综合
  + Analysis & Elaboration 将分析结果详细站是
  + Fitter(Place & Route) 装配，在芯片中放置各个模块
  + Assembler (Generate Programming) 进行汇编
  + Classic Timing Analysis 进行传统的时序分析