Research on the Design of Electronic Clock with Diversified Functions

Yuyu Dai

School of North China Electric Power University, Baoding, China

3082558439@qq.com

Keywords: Multi-function electronic clock, hourly timekeeping, proofreading function, start/stop counting function.

Abstract: The main purpose of this paper is to realize a multi-function electronic clock, and use six eight-segment digital tubes to display each of the hour, minute and second. We use counters to implement an electronic clock. Then, the function of the hourly timekeeping is realized by "if" judgment of the number of time, tenth, and tenth digits; the proofreading function by setting "1" to the clock signal of the school and calibration; by setting a flag The bit "stop" is counted and the number of "clk" is controlled to implement the start/stop counting function.For the electronic clock, because it has many functions to be implemented, it is not easy to implement in one structure and it is easy to cause confusion. Therefore, in the code design, I used the idea of sub-module "component" to design each function module separately. Then, instantiate through the "port map" in the main program, and connect the functions of each module.Finally, we implement a multi-function electronic clock with hourly timekeeping function, proofreading function and start/stop counting.

1. Introduction

After learning and mastering the experimental principles and design methods of the digital clock, we concentrated the distributed functions and designed this multi-function electronic clock. The timing of this electronic clock ranges from 0 hours 0 minutes 0 seconds to 23 hours 59 minutes 59 seconds, and each digit is displayed by 6 8-segment digital tubes.

The first function of this electronic clock is hourly timekeeping. When the electronic clock is reported, the buzzer sounds 5 times, 1 sound per second. The second function is proofreading. Separate calibration and timing of the electronic clock is achieved by button control. The third is to use the button control to achieve the function of clearing and starting/stopping the electronic clock.

Because it has many functions to be implemented, it is not easy to implement in one structure and it is easy to cause confusion. Therefore, in the code design, the idea of “component” is adopted, and each function module is designed separately. Finally, in the main program, the "port map" is instantiated, and the functions of each module are connected to realize the multi-function of the electronic clock. Implementation of multi-function electronic clock.

2. Implementation of Multi-function Electronic Clock

In order to achieve this multi-function electronic clock without confusion, we divided the entire design into six modules (counting module, decoding module, buzzer module, clear & start/stop & crossover module, proofreading hours & proofreading minute module and debounce module) and a main program. The entire code design is implemented in the assembly language of VHDL.

2.1 Counting Module

We have designed a total of three counters, namely the modulo 60 second counter, the modulo 60 minute counter, and the modulo 24 hour counter. In general, the design of these three counters is basically the same. The basic idea is to nest with two "if" structures, and generate a clock pulse "1" when the tens of the "second" counter and the tens of the "minute" counter are 6. In this way, we can pass the carry signal from the second counter to the minute counter (from the minute counter to the time counter), thus realizing the normal counting function of an electronic clock.

2.2 Decoding Module

The realization of this module is to generate a seven-digit binary number according to the coding law of the common cathode seven-segment digital tube. In this way, we can convert the integer of each digit of the counter into a seven-digit binary number, so that the output is displayed on the digital tube.

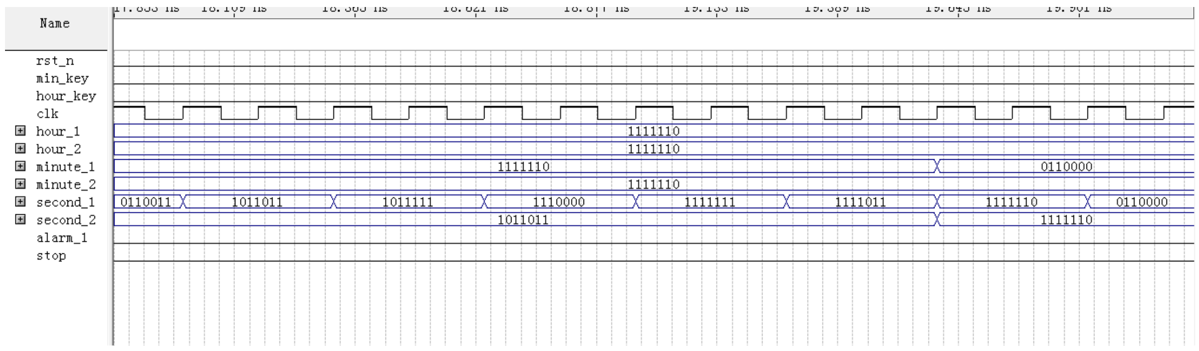

Figure 1: Simulation results of decoding module.

2.3 Buzzer Module

The design idea of this module is to judge whether the ten digits of the "minute" counter, the ones of the "minute" counter, and the ten digits of the "second" counter are 0 at the same time by the "if" statement. If the judgment of the "if" statement is affirmative, and the ones of the "second" counter are 0, 2, 4, 6, and 8, let the output "alarm_1" be "1". At this time, the port is connected to the EDA experiment box to realize the timekeeping function of the buzzer.

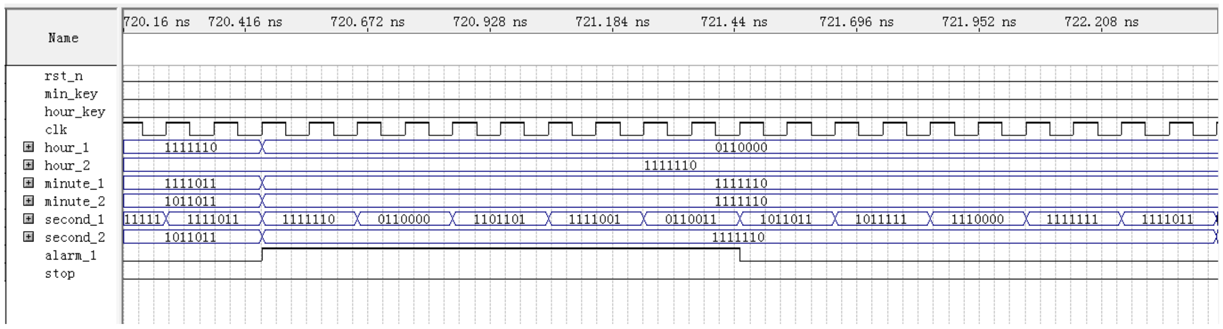

Figure 2: Simulation results of buzzer module.

2.4 Clear & Start/Stop & Crossover Module

The first is the implementation of the clear function. We correspond "rst_n" to the port of the button. There is a setting of "rst_n" in the module of each counter. When "rst_n=1", each bit of the counter is set to "0". Then there is the start/stop module. When the "stop" signal produces a rising edge of the pulse, "q1" is incremented by 1, and then the input clock is set by the condition "q1". When "q1=1", the output clock signal is "0", indicating stop; when "q1=2", the "clk" signal on the experiment box is divided by two (because the clock frequency on the experiment box is the minimum 2 Hz, which is 0.5 sec, and 1 second after the crossover, at which point the count is restarted.)

The program code is:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
--use ieee.std_logic_arith.all;
entity stop1 is
port(clk:in std_logic;
        stop:in std_logic;
        clk_1:out std_logic);
    end entity stop1;

    architecture rtl of stop1 is
   signal q1:integer range 0 to 2;
    begin
   process(stop)is
   begin
   if stop'event and stop='1'then
     q1<=q1+1;
     end if;
     if q1=2 then
     q1<=0;
     end if;
     end process;
   process(clk)is
   variable temp: integer range 0 to 1;
   begin
   if q1=1 then
   clk_1<='0';
   elsif q1=0 then
      if clk'event and clk='1'then
         if temp=1 then
           temp:=0;
           else
           temp:=temp+1;
           end if;
           if temp<1 then
           clk_1<='0';
           else
           clk_1<='1';
--           0.5s->1s
           end if;
           end if;
           end if;
           end process;
           end architecture rtl;
```

2.5 Proofreading Hours & Proofreading Minute Module

This module is implemented using an OR gate circuit. We match the calibration key signal with the clock pulse "1 clk_min" generated when the tens of the "second" counter is 6. When the button is not pressed, the output is “clk_min”, but when the button is pressed, the output becomes “1”. At this time, “clk_min” is manually set to 1, so the “minute” counter The bit will be incremented by one to achieve the function of the calibration (because the count plus 1 is to have a rising edge of the clock, so long press the button will not appear always added). The proofreading hour function is similar to the proofing function.

The program code is:

```
library ieee;
use ieee.std_logic_1164.all;
entity or_2 is
```

```
port(a,b:in std_logic;
        c:out std_logic);
  end entity or_2;

architecture rtl of or_2 is
begin
c<=a or b;
end architecture rtl;
```

2.6 Debounce Module

In the actual debugging process, due to the fact that there is always a large jump in the display and calibration time (that is, the button jitter is relatively large), it is necessary to debounce the button. The method of this implementation is delay debounce, and the proofreading minute and proofreading hour are only performed when the button is pressed for more than 1 second, thus reducing the influence of small jitter on the experimental display.

The program code is:

```
library ieee;
use ieee.std_logic_1164.all;
entity XD is
port( clk: in std_logic;
         input:in std_logic;
         output:out std_logic);
         end entity XD;
   architecture rtl of XD is
   signal a:std_logic;
   signal count:integer range 0 to 2;
   begin
   process(clk)is
   begin
   if input='0'then
   count<=0;
   elsif clk'event and clk='1' then
   if count=2 then
   count<=count;
   else
   count<=count+1;
   end if;
   end if;
   if count=2 then
   a<='1';
   else
   a<='0';
   end if;
   end process;
   output<=a;
   end architecture rtl;
```

2.7 Main Program

The function of each module is connected by a "port map" statement, and the bridge connecting it is a clock signal (a divided clock signal, a carry signal). This will basically realize the function of the entire electronic clock.

The program code is:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity digital_clock is
port(rst_n,clk:in std_logic;
       min_key,hour_key:in std_logic;
       stop:in std_logic;
       alarm_1:out std_logic;
         second_1,second_2:out
   std_logic_vector(6 downto 0);
         minute_1,minute_2:out
  std_logic_vector(6 downto 0);
         hour_1,hour_2:out
  std_logic_vector(6 downto 0));
   end entity digital_clock;

  architecture rtl of digital_clock
 is
  component module_second is
  port(rst_n,clk:in std_logic;
       second1:out integer range 0 to
9;
       second2:out integer range 0 to
9;
       clk_min:out std_logic);
end component module_second;
  component module_minute is
port(rst_n,clk_min:in std_logic;
       minute1:out integer range 0 to
9;
       minute2:out integer range 0 to
9;
       clk_hour:out std_logic);
  end component module_minute;
  component module_hour is
port(rst_n,clk_hour:in std_logic;
       hour1:out integer range 0 to 9;
       hour2:out integer range 0 to 9);
  end component module_hour;
component decode_dis is
port(din:in integer range 0 to 9;
       dout:out std_logic_vector(6
downto 0));
end component decode_dis;
component or_2 is
port( a,b:in std_logic;
        c:out std_logic);
end component or_2;
```

```
    component alarm is
    port(min_show_1,min_show_2:in
    integer range 0 to 9;
           sec_show_1,sec_show_2:in
    integer range 0 to 9;
           alarm1:out std_logic);
    end component alarm;
    component stop1 is
    port(clk:in std_logic;
           stop:in std_logic;
           clk_1:out std_logic);
      end component stop1;
      component XD is
    port( clk: in std_logic;
            input:in std_logic;
            output:out std_logic);
                                                    end component XD;
                                            Signal
d1,d2,d3,d4,clk_1,hour_key_1,min_key_1:std_logic;
                                            signal
second1,second2,minute1,minute2,hour1,hour2:integer range 0 to 9;

u1:module_second port map(rst_n=>rst_n,
                                                                  clk=>clk_1,
                                                                  second1=>second1,
                                                                  second2=>second2,
                                                                  clk_min=>d1);
                                              u2:module_minute port map(rst_n=>rst_n,
                                                                  clk_min=>d2,
                                                                  minute1=>minute1,
                                                                  minute2=>minute2,
                                                                  clk_hour=>d3);
                                              u3:module_hour port map(rst_n=>rst_n,
                                                                  clk_hour=>d4,
                                                                  hour1=>hour1,
                                                                  hour2=>hour2);
                                              u4:decode_dis port map( din=>hour1,
                                                                dout=>hour_1);
                                              u5:decode_dis port map( din=>hour2,
                                                                dout=>hour_2);
                                              u6:decode_dis port map( din=>minute1,
                                                                dout=>minute_1);
                                              u7:decode_dis port map( din=>minute2,
                                                                dout=>minute_2);
                                              u8:decode_dis port map( din=>second1,
                                                                dout=>second_1);
                                              u9:decode_dis port map( din=>second2,
                                                                dout=>second_2);

 u10:or_2 port map( a=>min_key_1,
                             b=>d1,
                             c=>d2);
 u11:or_2 port map( a=>hour_key_1,
                              b=>d3,
                              c=>d4);

   --   minute hour corrcet
   u12:XD port map( clk=>clk,
                            input=> min_key,

    output=>min_key_1);
   u13:XD port map( clk=>clk,
                            input=> hour_key,

    output=>hour_key_1);
   u14:alarm port
      map( min_show_1=>minute1,

    min_show_2=>minute2,

    sec_show_1=>second1,

    sec_show_2=>second2,

    alarm1=>alarm_1);
   u15:stop1 port map( clk=>clk,
                                stop=>stop,
                                clk_1=>clk_1);
      end architecture rtl;
```

3. Conclusion

How to implement multiple functions on one electronic clock at the same time is the main direction of our research. In order to prevent confusion in this structure, when we use EDA's VHDL language to design code, we use the idea of sub-modules to design each functional module, including a total of six modules(counting module, decoding module, buzzer module, clear & start/stop & crossover module, proofreading hours & proofreading minute module and debounce module) . Then the main program is instantiated by "port map", which is equivalent to connecting the functions of each module. Finally, we implemented a multi-function electronic clock with hourly timekeeping function, proofreading function and start/stop counting function. The electronic clock can also add a lot of functions, and we can further study on this basis to realize more complicated and more perfect electronic clocks.

References

[1] Zhao De-sheng, Li De-cang. Design and implementation of Digital Electronic Clock System based on AT89S52[J]. Railway Computer Application,2010,19(12):43-46.

[2] Hong Shuliang, Wang Fugiang, Liu Junwei. Design of a Multi-function Electronic Clock of Infrared Remote Control Based on MCU. Electronic Science and Technology,2015,28(05):85-86.

[3] W.J Llope, N Adams, K.K Kainz. An electronic clock for correlated noise corrections[J]. Nuclear Inst. and Methods in Physics Research, A,2000,443(2).