DIGITAL ALARM CLOCK

ABSTRACT

Most people in the whole world use digital clocks with various functionalities. This is a technical report for this project - DIGITAL ALARM CLOCK. The proposed system will count the hour, minute, and second, along with 4 modes of operation like resetting and setting the time, setting alarm, changing hour format. The system was modeled in VHDL (Very-high-speed integrated circuit Hardware Description Language) in Quartus.

ANALYSIS

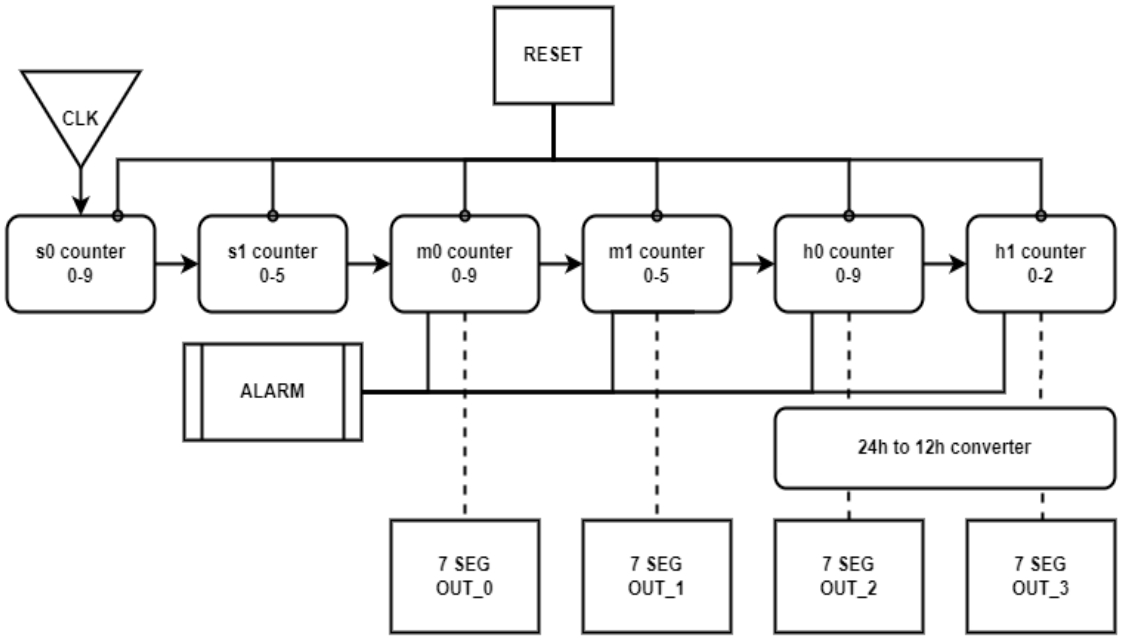
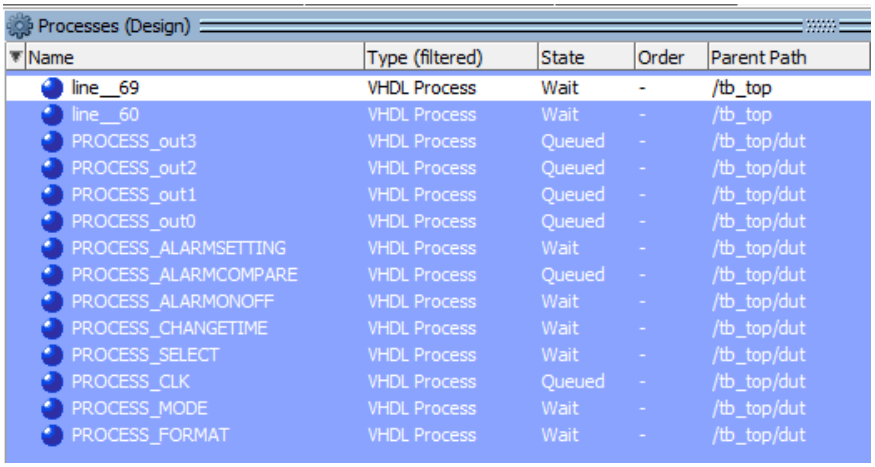


Figure.1: Block diagram of the digital alarm clock

Features

- Processes



- Four Operating Modes:
  - **Mode Control - Using Q(4bit)** : It goes like *0001 -> 0010 -> 0100 -> 1000*
  - **Mode0 - Display Mode :** 7 segment decoders are used. The following is the behavior of the decoder. I have implemented the decoder at the algorithmic level:

```
-----------------display 7 SEG-----------------
   PROCESS_m0 : PROCESS(r_m0, CLK) is
      begin
         if (rising_edge(clk)) then
            r_m0 <= m0;
            if (Q = "0100") then
               r_m0 <= a_m0;
            elsif(alarmOut = "1111111") then
               OUT_0 <= transport alarmOut after 60sec;
            end if;
            case r_m0 is
               when "0000"=> OUT_0 <="0111111";   -- '0'
               when "0001"=> OUT_0 <="0000110";   -- '1'
               when "0010"=> OUT_0 <="1011011";   -- '2'
               when "0011"=> OUT_0 <="1001111";   -- '3'
               when "0100"=> OUT_0 <="1100110";   -- '4'
               when "0101"=> OUT_0 <="1101101";   -- '5'
               when "0110"=> OUT_0 <="1111101";   -- '6'
               when "0111"=> OUT_0 <="0000111";   -- '7'
               when "1000"=> OUT_0 <="1111111";   -- '8'
               when "1001"=> OUT_0 <="1101111";   -- '9'
               when "1010"=> OUT_0 <="1110111";   -- 'A'
               when "1011"=> OUT_0 <="1111100";   -- 'b'
               when "1100"=> OUT_0 <="1111001";   -- 'C'
               when "1101"=> OUT_0 <="1011110";   -- 'd'
               when "1110"=> OUT_0 <="1111001";   -- 'E'
               when "1111"=> OUT_0 <="1110001";   -- 'F'
               when others => NULL;
            end case;
         end if;
      end PROCESS PROCESS_m0;
```

Figure.2: 7 segment decoder code of m0

- **Mode1 - Change hour format:** The default hour format of the project is 12h. We can change it to 24h or 12h using a converter. *T_Format* is given as an internal signal which is triggered when we go to this mode.
- **Mode2 - SET alarm:** When Q = “0010”, We can set alarm time at PROCESS_ALARMSETTING. When current time reaches the alarm time, “alarmOut” convert “00000000” to “11111111” and 7 segment outputs FF:FF for 1 min. We can turn off the alarm using button “A” which change signal of “alarmOn” 1 to 0 also vice versa.
- **Mode3 - SET Current time**: We can change the time according to the time zone. With the help of UP and DOWN, the time can be adjusted.

- **RESET:** When reset the clock starts from 00:00 or 12 AM. And the alarm is set at 11:11.
- From the structural model of the clock in Figure.1, it is observed that we can embed the output of all the hour, minute, and second in 4-bit binary.
- Many FSM were analyzed to have the complete design. The UP and DOWN function values are stored and are given for display only when SET is high.
- The alarm set vector and time vectors of all hour bits and minute bits are compared. When all equal then the display vector values become “1111”, making it display FF.

SIMULATION RESULTS

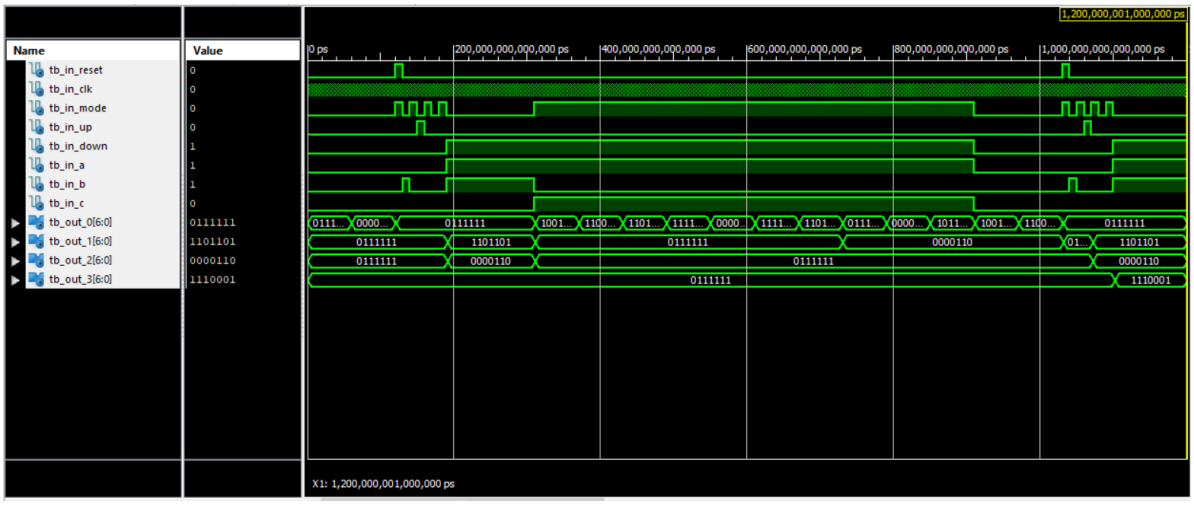


Figure.5: Output waveform

User Guide

- Reset: Time is set to 00:00 in 12h format and alarm is set to 11:11
- Mode:
    - Mode_0: Display time
    - Mode_1: Time format change, you can change the format by changing C(additional input)
    - Mode_2: Set alarm using UP and DOWN. At this time you can see the alarm time to set in the display and using B(additional input) you can select the set(hour *00~24 or 12~00* /min *00~59~00*) to change.
    - Mode_3: Set time. Similarly, here change the time to set using UP and DOWN. The display will show the time to set during this mode.
- UP/DOWN: To set the time and alarm.
- A: The user can set the alarm off by keeping A high(‘1’).
- B: This is for set selection. It goes like *minute -> hour*
- C: This is for changing the Time format when in Mode_2.

CONCLUSION

In this project, I have implemented a digital clock with an alarm using VHDL in Quartus. I learned more fundamental principles of designing digital systems. The digital clock can be reset and we can adjust the time according to different time zones too. It has 4 displays that make it user-friendly.

REFERENCES

[1] Oyewale Ademola S., ‘*Digital Alarm Clock SIWES 300 Project*’, GRID LABORATORY, OBAFEMI AWOLOWO UNIVERSITY ILE-IFE

[2] Tesfamichael Molla, ‘*Digital Clock Project*’, Technical Report · January 2017
DOI: 10.13140/RG.2.2.27401.90724

[3] https://vhdlguru.blogspot.com/2010/03/digital-clock-in-vhdl.html

[4] https://electronics-course.com/digital-clock#hide

[5] https://www.fpga4student.com/2016/11/vhdl-code-for-digital-clock-on-fpga.html