University of Montenegro
Faculty of Electrical Engineering

Subject: Automated design of electronic circuits and systems

Topic: Real car alarm

Authors:
Milica Bastrica
Svetlana Zecevic

Date and location:
11.24.2018
Podgorica, Montenegro

Contents

Summary

Using FPGA board DE2 – 70 we are simulating the work of a real car alarm. Clock of 1Hz is gained using a clock divider and the fpga board's implemented clock of 50MHz. Switch SW[0] is used for turning on the alarm. However alarm is still not active. After 30s the alarm is active and LED[0] ,which is used for implication that the alarm is active, is lit up. As a consequence of that if any of the doors, which are represented by switches SW[1], SW[2], SW[3], SW[4], SW[5], are opened the siren will turn on.Siren will be activated after output delay of 30s. Siren is represented by LED[1] and it stays on for 60s, unless in the mean time the alarm gets deactivated.

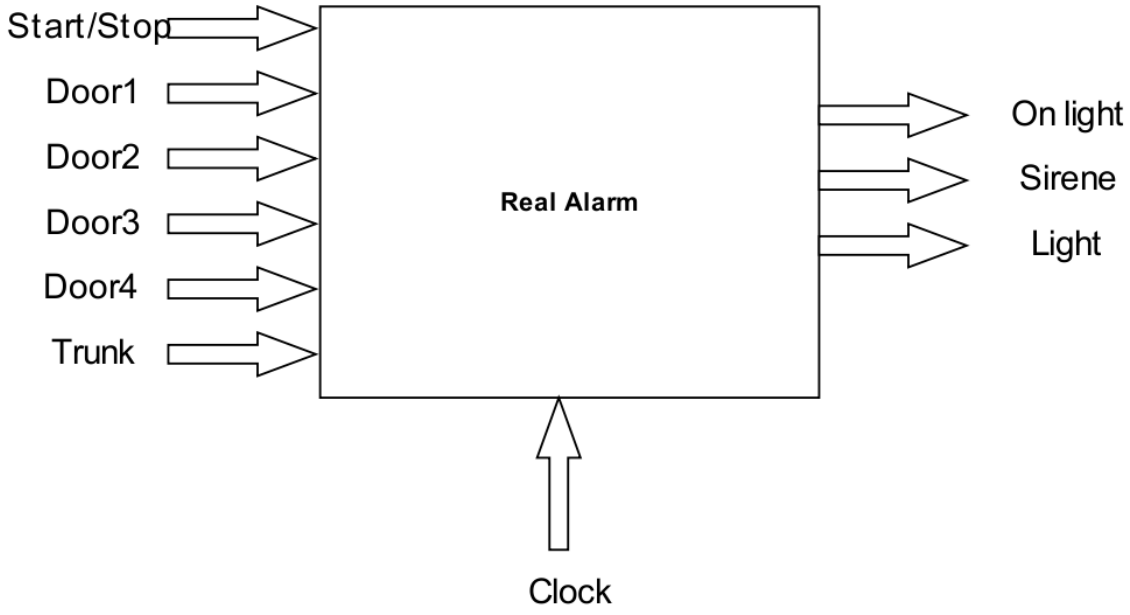
Description of the problem

The assignment is to make a real car alarm. Alarm is activated by opening any of the 5 sensors. Alarm has a switch start/stop which after activating activates an input delay of 30s.After the input delay all the sensors can be activated. When sensors get activated, the siren becomes active after output delay of 30s.The siren is on for 1 minute. By turning off the alarm the entire system is reset. LED diodes are used for illustrating the state of activated alarm, input and output delay …

Hardware – software solution and simulation

Altera Quartus 9.1 is used as a software solution. Code is in VHDL and simulations are represented in a WaveForm file.

Simple scheme:



Code:

Divider:

```
Library Ieee;
Use ieee.std_logic_1164.all;
Use Ieee.std_logic_arith.all;
Use ieee.std_logic_unsigned.all;

entity clk_div is
generic(N:integer:=50000000);
port
```

```
        ( clk: in std_logic;
        clk_new : out std_logic);
end clk_div;

architecture clk_div_behav of clk_div is
signal clk_temp : std_logic;
signal temp : integer range 0 to N-1;
begin
        process(clk, clk_temp)
        begin
                if(clk'event and clk='0') then
                        if(temp=N/2-1)then
                                temp<=temp+1;
                                clk_temp<='1';
                        elsif (temp=N-1) then
                                temp <= 0;
                                clk_temp<='0';
                        else
                                temp<=temp+1;
                        end if;
                                clk_new<=clk_temp;
                end if;
        end process;
end clk_div_behav;
```

Alarmled:

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

entity alarmled is
port(clk,button:IN STD_LOGIC;
        pulse,pulseoff,ledout,onlight:OUT STD_LOGIC);
end alarmled;

architecture alarmled_arh of alarmled is
constant count_max:INTEGER:=30;
constant bin_active:STD_LOGIC:='1';
signal count:INTEGER:=0;
signal temp:STD_LOGIC:='0';
type state_type is (off, wait_time, trigger);
signal state : state_type :=off;
begin
   process(button)
   begin
                if (button='1') then
                        onlight <= '1';
                else
                        onlight <= '0';
                end if;
        end process;
        pulseoff <= not button;
```

```
process (button,clk)
begin
        if (button = '0') then
                state <= off;
                count <= 0;
                pulse <= '0';
                temp <= '0';
        elsif (clk'event and clk='1') then
                case state is
                                when off =>
                                                if( button = bin_active) then
                                                        state <= wait_time;
                                                        count <= count + 1;
                                                else
                                                        state <= off;
                                                end if;
                                                   pulse <= '0';
                                when wait_time =>
                                                if(count = count_max) then
                                                        count <= 0;
                                                        state <= trigger;
                                                elsif (button ='0') then
                                                        count <= 0;
                                                else
                                                        count <= count + 1;
                                                end if;
                                when trigger =>
                                                pulse <= '1';
```

```
                                                    state <= off;
                                                    temp <= '1';

                                end case;
                            end if;
                        end process;
            with temp select ledout <=
    '1' when '1',
    '0' when '0';
end alarmled_arh;
```

Alarm:

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

entity alarm is
port(clk,turnon,turnoff,in1,in2,in3,in4,in5:IN STD_LOGIC;
pulse,pulseoff:OUT STD_LOGIC);
end alarm;

architecture alarm_arch of alarm is
 type state_type is (activated,deactivated);
 signal ps:state_type;
 signal temp:STD_LOGIC:='0';
 begin
            sync_proc:process(turnon,turnoff)
            begin
                        if (turnoff = '1') then
```

```
                ps <= deactivated;
        elsif (turnon = '1') then
                ps <= activated;
        end if;
    end process sync_proc;

    comb_proc:process(ps)
    begin
        pulseoff <= '0';
        case ps is
            when activated =>
                if ( in1='1' or in2='1' or in3='1' or in4='1' or in5='1')  then
                    pulseoff <= '0';
                    temp <= '1';
                end if;
            when deactivated =>
                    temp <= '0';
                    if (turnoff = '1') then
                        pulseoff <= '1';
                    else
                        pulseoff <= '0';
                    end if;
        end case;
    end process comb_proc;
        with temp select pulse <=
   '1' when '1',
   '0' when '0';
end alarm_arch;
```

Outcounter:

```
library ieee;
use ieee.std_Logic_1164.all;
use ieee.std_Logic_unsigned.all;
entity outcounter is
        generic(N:integer:=5);
        port(clk:in std_logic;
        output:out std_logic_vector(N-1 downto 0);
        start:in std_logic;
        pulse:in std_logic);
end outcounter;
architecture arh of outcounter is
signal br: std_logic_vector(N-1 downto 0);
begin
        process(clk,start)
        begin
                if (clk'event and clk ='1') then
                        if (start='1') then
                                if (pulse='0')then
                                        br<="00000";
                                elsif (br="11110" and pulse='1') then
                                        br<="11110";
                                else
                                        br<=br+1;
                                end if;
                        elsif (start='0') then
```

```
                    br<="00000";
                end if;
            end if;
        end process;
        output<=br;
end arh;
```

Comparator:

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

entity komparator is
port(input:in std_logic_vector(4 downto 0);
        output: out std_logic;
        pulse: in std_logic);
end komparator;
architecture arh of komparator is
begin
process(input,pulse)
begin
        if(pulse='1') then
            case  input is
                when "11110" => output<='1';
                when others => output<='0';
            end case;
        elsif(pulse='0') then
            case input is
```

```
                when "00000" => output<='0';
                when others => output<='0';
        end case;
    end if;
end process;
end arh;
```

Monostable:

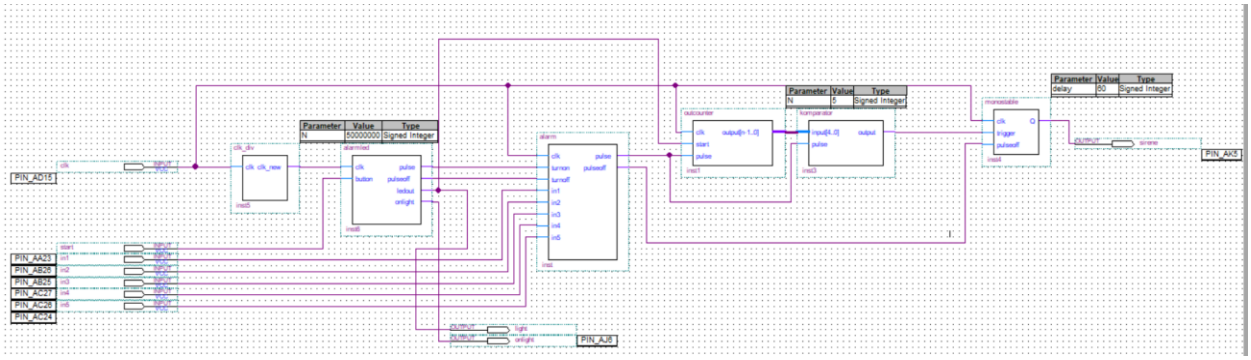
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity monostable is
generic(delay : INTEGER := 60);
port(clk,trigger,pulseoff:IN STD_LOGIC;
    Q:OUT STD_LOGIC);
end monostable;
architecture monostable_arch of monostable is
begin
    process(clk)
    variable count:INTEGER := 0;
    variable trigger_was : BIT := '0';
    begin
    if(clk 'event and clk='1')then
        if (trigger = '1' and trigger_was = '0') then
            count := delay;
```

```
                trigger_was := '1';
            elsif count = 0 then
                count := 0;
            else
                count := count-1;
            end if;
            if trigger = '0' then
                trigger_was := '0';
            end if;
            if pulseoff = '1' then
                count := 0;
            end if;
        end if;
        if count = 0 then
            Q <= '0';
        else
            Q <= '1';
        end if;
end process;
end monostable_arch;
```

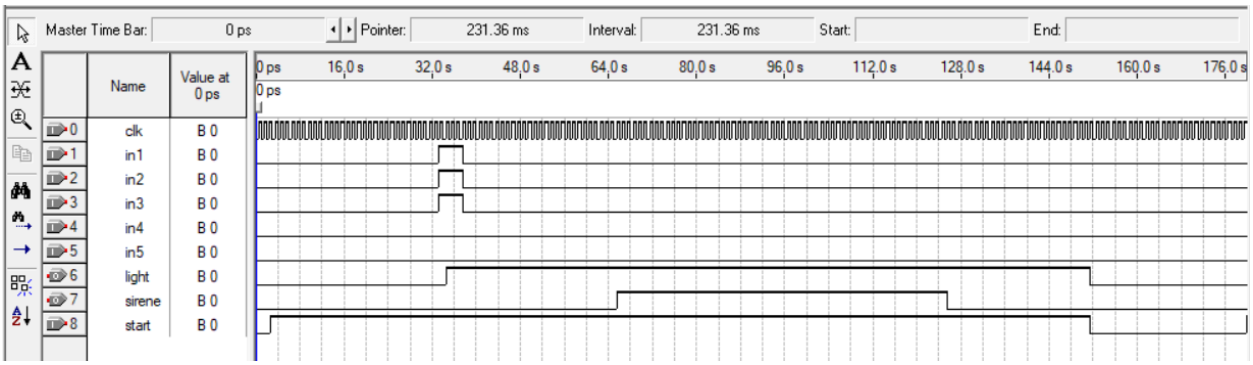
For converting the frequence of 50MHz into frequency of 1Hz we used the clock divider. Component alarmled is used for getting an input delay of 30s. The way it works is that the alarm system gets activated by turning on the switch and the counter starts incrementing by one on every rising edge of the clock until it reaches the value of 30. When that value is reached one small impulse is sent to activate the next component,as an indication that the alarm is active,the LED1 of the altera board is turned on. The next component is alarm. When any of the switches that are respresenting doors are turned on , we have one as output. That one is than brought on to the input of the counter , our next component, which counts to 30. The information about the moment 30s is reached we get by using the comparator which sends that information to the monostable multivibrator via impulse. That way we generate an output delay. The function of the

monostable multivibrator is to generate an impulse that last for 60s, that impulse is in our case the sirene of the alarm(LED2).

Bdf file:



Simulation as a wvf file:



FPGA board DE2 – 70 (Cyclone II) is used as a hardware solution

Pinout:

clk PIN_AD15 (50MHz)

onlight PIN_AJ6 (LED[0])

ledout PIN_AK5(LED[1])

sirene PIN_AJ5 (LED[2])

start/stop PIN_AA23(SW[0])

in1 PIN_AB26(SW[1])

in2 PIN_AB25(SW[2])

in3 PIN_AC27(SW[3])

in4 PIN_AC26(SW[4])

in5 PIN_AC24(SW[5])

Verification of the work on the board

Attached link.

Video link

https://youtu.be/VFKUFAtwY1E

Literature used

AUTOMATIZOVANO PROJEKTOVANJE DIGITALNIH SISTEMA (VHDL i FPGA)

DE2_70_User_manual_v109