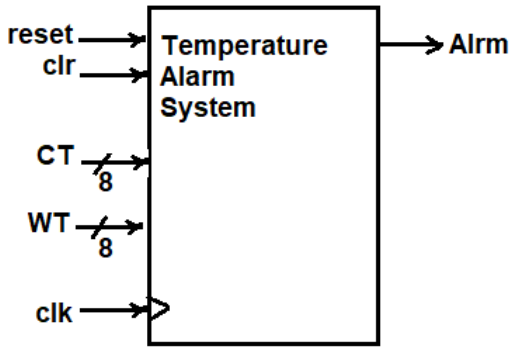
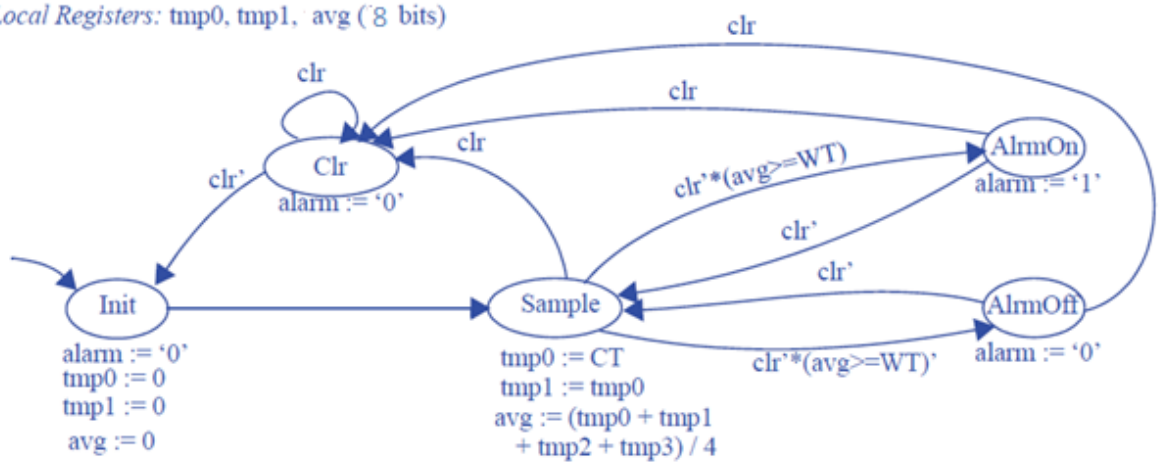
EEL4740 - Embedded Computing Systems Final Exam Review Problems Part 2- Solutions

Design a single purpose processor to create an alarm system that sets a single-bit output alarm to '1' when the average temperature of two consecutive samples meets or exceeds a user-defined threshold value. An 8-bit unsigned input CT indicates the current temperature, and an 8-bit unsigned input WT indicates the warning threshold. Samples should be taken every millisecond. A single-bit input clr when 1 disables the alarm and the sampling process. Start by capturing the desired system behavior as an HLSM, draw the state diagram and then write the VHDL model.



Step 1 - Capture a high-level state machine

Inputs: CT, WT (8 bits); clr (bit)
Outputs: alarm (bit)
Local Registers: tmp0, tmp1, avg (8 bits)



Step 2 – VHDL Model

Input are 0 when they are supposed to be 1 and vice versa, so the program will run on the Elbert V2 board.

```
21  use IEEE.STD_LOGIC_1164.ALL;
22  use IEEE.NUMERIC_STD.ALL;
23  use IEEE.STD_LOGIC_UNSIGNED.ALL;
24
25
26  entity TemperatureAlrm is
27      Port ( clk : in  STD_LOGIC;
28             reset : in  STD_LOGIC;
29             clr : in  STD_LOGIC;
30             CT : in  STD_LOGIC_VECTOR (7 downto 0);
31             WT : in  STD_LOGIC_VECTOR (7 downto 0);
32             Alrm : out  STD_LOGIC);
33  end TemperatureAlrm;
34
35  architecture Behavioral of TemperatureAlrm is
36  type statetype is (Init, Clear, Sample, AlrmOn, AlrmOff);
37  signal state, statenext: statetype;
38  signal temp0, temp0Next: STD_LOGIC_VECTOR(7 downto 0);
39  signal temp1, temp1Next: STD_LOGIC_VECTOR(7 downto 0);
40  signal ave, aveNext: STD_LOGIC_VECTOR(7 downto 0);
41  signal count_1ms, t: unsigned(24 downto 0);
42  signal sumReg, asumReg : STD_LOGIC_VECTOR(7 downto 0);
43  signal bT_1ms: STD_LOGIC;
44
45  constant U_Zero: STD_LOGIC_VECTOR(7 downto 0) := "00000000";
46  constant U_ONE: STD_LOGIC_VECTOR(7 downto 0) := "00000001";
47
48  begin
```

```
t <= "00000000000000000000000000";

   reg: process(clk) is
   begin
      if rising_edge(clk)then
         count_1ms <= count_1ms + 1;

         if count_1ms > 6000 then   --1 milisecond period
            if bT_1ms = '0' then
               bT_1ms <= '1';
            else
               bT_1ms<= '0';
            end if;
            count_1ms <= t;
         end if;
      end if;
   end process reg;

   regs: process (bT_1ms, reset, clr) is
   begin
   if (reset ='0') then
   state <= Init;
      temp0 <= U_Zero;
      temp1 <= U_Zero;
      ave <= U_Zero;
   elsif (clr = '0') then
   state <= Clear;
   elsif rising_edge(bT_1ms) then
```

```
77      state <= statenext;
78      temp0 <= temp0Next;
79      temp1 <= temp1Next;
80      ave <= aveNext;
81      end if;
82      end process regs;
83
84      CombLogic: process (state, CT, WT) is
85      begin
86         case (state) is
87            when Init =>
88            Alrm <='0';
89            temp0Next <= U_Zero;
90            temp1Next <= U_Zero;
91            aveNext <= U_Zero;
92            statenext <= Sample;
93
94            when Sample =>
95            temp0Next <= CT;
96            temp1Next <= temp0;
97            sumReg <= temp0 + temp1;
98            asumReg <= SHR(sumReg, U_ONE);
99            aveNext <= asumReg;
100           if (clr = '0') then
101              statenext <= Clear;
102           elsif (ave < WT) then
103              statenext <= AlrmOff;
104           elsif (ave >= WT) then
```

```
105             statenext <= AlrmOn;
106          end if;
107
108          when AlrmOff =>
109          Alrm <= '0';
110          if (clr = '0') then
111             statenext <= Clear;
112          end if;
113
114          when AlrmOn =>
115          Alrm <= '1';
116          if (clr = '0') then
117             statenext <= Clear;
118          end if;
119
120          when Clear =>
121          Alrm <= '0';
122          if (clr = '1') then
123             statenext <= Init;
124          else
125             statenext <= Clear;
126          end if;
127
128          when others =>
129          statenext <= Clear;
130       end case;
131    end process;

132
133 end Behavioral;
```