ECE 448
Midterm Exam
Wednesday, February 27, 2019

Problem 1 (18%)

Assuming the following library, package and signal declarations:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
. . .
signal s1,s2,s3,s4,s5: std_logic_vector(3 downto 0);
signal u1,u2,u3,u4,u5: unsigned(3 downto 0);
signal sg: signed(3 downto 0);
```

please correct the following code, which gives a syntax error for each of its lines:

Incorrect code (with type mismatches):

```
s1 <= sg;
s2 <= -5;
s3 <= s4 - sg;
sg <= u1 – u2;
u3 <= s5 - 1;
u4 <= 7;
```

Corresponding correct code:

Problem 2 (22%)

Fill in the blanks in the code of the universal N-bit binary counter:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity univ_bin_counter is
   generic(……: …… := 8);
   port(
      clk      : in  std_logic;
      reset    : in  std_logic;
      syn_clr  : in  std_logic;
      load     : in  std_logic;
      en       : in  std_logic;
      up       : in  std_logic;
      d        : in  std_logic_vector(…… downto 0);
      max_tick : out std_logic;
      min_tick : out std_logic;
      q        : out std_logic_vector(…… downto 0)
   );
 end univ_bin_counter;
architecture arch of univ_bin_counter is
   signal r_reg  : unsigned(…… downto 0);
   signal r_next : unsigned(…… downto 0);
begin
   -- register
   process(……)
   begin
      if (reset='1') then
         r_reg <= ……;
      elsif (……) then
         r_reg <= ……;
      end if;
   end process;
   -- next-state logic
   r_next <= …… when syn_clr='1' else
             …… when load='1' else
```

```
                .....  when  en  ='1'  and  up='1'
else
                .....  when  en  ='1'  and  up='0'
else
                .....;
   -- output logic
   q <= .....;
   ..... <= '1' when r_reg=(2**N - 1) else '0';
   ..... <= '1' when r_reg=0 else '0';
end arch;
```

Problem 3 (20%)

Draw a block diagram of the circuit that calculates a minimum of three 4-bit signed inputs A, B, and C built of comparators and multiplexers.

Problem 4 (40%)

Draw a block diagram and an ASM chart described by the following code:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity debounce is
   port(
      clk, reset : in  std_logic;
      sw         : in  std_logic;
      db_level   : out std_logic;
      db_tick    : out std_logic
   );
end debounce;

architecture exp_fsmd_arch of debounce is
   constant N: integer:=21;  -- 2^N * 10ns = 20ms
   type state_type is (zero, wait0, one, wait1);
   signal state_reg, state_next : state_type;
   signal q_reg, q_next  :unsigned(N - 1 downto 0);
   signal q_load, q_dec, q_zero : std_logic;
begin
   -- FSMD state & data registers
   process(clk,reset)
   begin
      if reset='1' then
         state_reg <= zero;
         q_reg <= (others=>'0');
      elsif (clk'event and clk='1') then
         state_reg <= state_next;
         q_reg <= q_next;
      end if;
   end process;
   -- FSMD data path (counter) next-state logic
   q_next <= (others=>'1') when q_load='1' else
             q_reg - 1 when q_dec='1' else
             q_reg;
   q_zero <= '1' when q_next=0 else '0';
   -- FSMD control path next-state logic
   process(state_reg,sw,q_zero)
   begin
      q_load  <= '0';
      q_dec   <= '0';
      db_tick <= '0';
      state_next <= state_reg;
      case state_reg is
         when zero =>
            db_level <= '0';
            if (sw='1') then
               state_next <= wait1;
               q_load <= '1';
```

```
            end if;
         when wait1=>
            db_level <= '0';
            if (sw='1') then
               q_dec <= '1';
               if (q_zero='1') then
                  state_next <= one;
                  db_tick    <= '1';
               end if;
            else -- sw='0'
               state_next <= zero;
            end if;
          when one =>
            db_level <= '1';
            if (sw='0') then
               state_next <= wait0;
               q_load     <= '1';
            end if;
           when wait0=>
               db_level <= '1';
               if (sw='0') then
                  q_dec <= '1';
                  if (q_zero='1') then
                     state_next <= zero;
                  end if;
               else -- sw='1'
                  state_next <= one;
               end if;
      end case;
   end process;
end exp_fsmd_arch;
```