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):

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

Corresponding correct code:

Problem 2 (22%)

Fill in the blanks in the code of the universal N-bit shift register:

```
library ieee;
use ieee.std_logic_1164.all;
entity univ_shift_reg is
   ……(N: …… := 8);
   port(
      clk   : in  std_logic;
      reset : in  std_logic;
      ctrl  : in std_logic_vector(…… downto 0);
      d     : in std_logic_vector(…… downto 0);
      q     : out std_logic_vector(…… downto 0)
   );
end univ_shift_reg;
architecture arch of univ_shift_reg is
   signal r_reg: std_logic_vector(…… downto 0);
   signal r_next: std_logic_vector(…… downto 0);
begin
   -- register
   process(……)
   begin
      if (reset='1') then
         r_reg <= ……;
      elsif (……) then
         r_reg <= ……;
      end if;
   end process;
   -- next-state logic
   with ctrl select
    r_next <=
      …… when "00", --no op
      …… when "01", --shift left;
      …… when "10", --shift right;
      …… when others; -- load
   -- output
   q <= ……;
end arch;
```

Problem 3 (20%)

Draw a block diagram of the circuit that calculates the minority function of four variables, x3, x2, x1, x0, built using a ROM. On a separate diagram, show the full contents of the ROM.

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 db_fsm is
   port( clk, reset: in std_logic;
         sw    : in std_logic;
         db    : out std_logic );
end db_fsm;

architecture arch of db_fsm is
   constant N: integer := 20;  -- 2^N * 10ns = 10ms tick
    type db_state_type is
      (zero,wait1_1,wait1_2,wait1_3,one,wait0_1,wait0_2,wait0_3);
   signal q_reg, q_next : unsigned(N-1 downto 0);
   signal m_tick        : std_logic;
   signal state_reg     : db_state_type;
   signal state_next    : db_state_type;
begin
```

```
--*************************************************
-- counter to generate 10 ms tick
--*************************************************
process(clk,reset)
begin
   if (clk'event and clk='1') then
      q_reg <= q_next;
   end if;
end process;
-- next-state logic
q_next <= q_reg + 1;
--output tick
m_tick <= '1' when q_reg=0 else '0';
--*************************************************
-- debouncing FSM
--*************************************************

-- state register
process(clk,reset)
begin
   if (reset='1') then
      state_reg <= zero;
   elsif (clk'event and clk='1') then
      state_reg <= state_next;
   end if;
end process;
-- next-state logic
process(state_reg,sw,m_tick)
begin
   state_next <= state_reg; --default: back to same state
   db <= '0';   -- default 0
   case state_reg is
      when zero =>
         if sw='1' then
            state_next <= wait1_1;
         end if;
      when wait1_1 =>
         if sw='0' then
            state_next <= zero;
         else
            if m_tick='1' then
               state_next <= wait1_2;
            end if;
         end if;
      when wait1_2 =>
         if sw='0' then
            state_next <= zero;
         else
            if m_tick='1' then
               state_next <= wait1_3;
            end if;
         end if;
```

```
         when wait1_3 =>
            if sw='0' then
               state_next <= zero;
            else
               if m_tick='1' then
                  state_next <= one;
               end if;
            end if;
         when one =>
            db <='1';
            if sw='0' then
               state_next <= wait0_1;
            end if;
         when wait0_1 =>
            db <='1';
            if sw='1' then
               state_next <= one;
            else
               if m_tick='1' then
                  state_next <= wait0_2;
               end if;
            end if;
         when wait0_2 =>
            db <='1';
            if sw='1' then
               state_next <= one;
            else
               if m_tick='1' then
                  state_next <= wait0_3;
               end if;
            end if;
         when wait0_3 =>
            db <='1';
            if sw='1' then
               state_next <= one;
            else
               if m_tick='1' then
                  state_next <= zero;
               end if;
            end if;
      end case;
   end process;
end arch;
```