

Massachusetts Institute of Technology
 Department of Electrical Engineering and Computer Science
 6.111 – Introductory Digital Systems Laboratory
 Problem Set 4 solutions

March 14, 2003

Part I

```
-- VHDL code for fourbitmult.vhd
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity fourbitmult is port(
    multA      : in std_logic_vector(3 downto 0);
    multB      : in std_logic_vector(3 downto 0);
    multOUT    : out std_logic_vector(7 downto 0));
end fourbitmult;

architecture multarch of fourbitmult is

    -- component declarations
    component fulladd
    port (
        ina, inb, inc : in  std_logic;    -- inputs
        sumout, outc  : out std_logic);    -- outputs
    end component;

    component halfadd
    port (
        a, b  : in  std_logic;            -- inputs
        sum, c : out std_logic);          -- outputs
    end component;

    signal x0y1, x0y2, x0y3, x1y0, x1y1, x1y2, x1y3, x2y0, x2y1, x2y2,
    x2y3, x3y0, x3y1, x3y2, x3y3, hc0, hc1, hc2, hc3, fc0, fc1,
    fc2, fc3, fc4, fc5, fc6, fc7, hs1, fs0, fs1, fs2, fs3, fs4 : std_logic;
    -- internal signals

begin
    -- anding bits together
    multOUT(0) <= multA(0) and multB(0);
    x0y1 <= multA(0) and multB(1);
    x0y2 <= multA(0) and multB(2);
    x0y3 <= multA(0) and multB(3);

    x1y0 <= multA(1) and multB(0);
    x1y1 <= multA(1) and multB(1);
    x1y2 <= multA(1) and multB(2);
    x1y3 <= multA(1) and multB(3);

    x2y0 <= multA(2) and multB(0);
    x2y1 <= multA(2) and multB(1);
    x2y2 <= multA(2) and multB(2);
```

```
x2y3 <= multA(2) and multB(3);

x3y0 <= multA(3) and multB(0);
x3y1 <= multA(3) and multB(1);
x3y2 <= multA(3) and multB(2);
x3y3 <= multA(3) and multB(3);

--structural instantiation of half adders and full adders\\
h0 : halfadd port map (
  a  => x0y1,
  b  => x1y0,
  sum => multOUT(1),
  c  => hc0);
h1 : halfadd port map (
  a  => x3y1,
  b  => fc1,
  sum => hs1,
  c  => hc1);
h2 : halfadd port map (
  a  => x0y2,
  b  => fs0,
  sum => multOUT(2),
  c  => hc2);
h3 : halfadd port map (
  a  => x0y3,
  b  => fs2,
  sum => multOUT(3),
  c  => hc3);
f0 : fulladd port map (
  ina => x1y1,
  inb => x2y0,
  inc => hc0,
  sumout => fs0,
  outc  => fc0);
f1 : fulladd port map (
  ina  => x2y1,
  inb  => x3y0,
  inc  => fc0,
  sumout => fs1,
  outc  => fc1);

f2 : fulladd port map (
  ina  => x1y2,
  inb  => fs1,
  inc  => hc2,
  sumout => fs2,
  outc  => fc2);

f3 : fulladd port map (
  ina  => x2y2,
  inb  => hs1,
  inc  => fc2,
  sumout => fs3,
  outc  => fc3);

f4 : fulladd port map (
  ina  => x3y2,
```

```

    inb  => hc1,
    inc  => fc3,
    sumout => fs4,
    outc => fc4);

f5 : fulladd port map (
    ina  => x1y3,
    inb  => fs3,
    inc  => hc3,
    sumout => multOUT(4),
    outc => fc5);

f6 : fulladd port map (
    ina  => x2y3,
    inb  => fs4,
    inc  => fc5,
    sumout => multOUT(5),
    outc => fc6);

f7 : fulladd port map (
    ina  => x3y3,
    inb  => fc4,
    inc  => fc6,
    sumout => multOUT(6),
    outc => multOUT(7));
end multarch;

--full adder and half adder
-- This comment is before the library and use clauses.
library ieee;
use ieee.std_logic_1164.all;
-- here is the entity
entity fulladd is
    port (ina, inb, inc : in std_logic;
          sumout, outc : out std_logic);
end fulladd;
-- here is the architecture
architecture top of fulladd is
    -- this is the declaration of components and signals.
    component halfadd
        port (a, b : in std_logic;
              sum, c : out std_logic);
    end component;
    signal s1, s2, s3 : std_logic;
    -- here is the body of the architecture
begin
    -- the order is irrelevant, all are computed concurrently
    -- a structural instantiation of two half adders
    h1: halfadd port map( a => ina, b => inb,
                          sum => s1, c => s3);
    h2: halfadd port map( a => s1, b => inc,
                          sum => sumout, c => s2);
    -- a concurrent statement implementing the or gate
    outc <= s2 or s3;
end top;

-- This comment is before the library and use clauses.

```

```

library ieee;
use ieee.std_logic_1164.all;
-- here is the entity
entity halfadd is
  port (a, b : in std_logic;
        sum, c : out std_logic);
end halfadd;
-- here is the architecture
architecture comp of halfadd is
begin
  -- a concurrent statement implementing the and gate
  c <= a and b;
  -- a concurrent statement implementing the xor gate
  sum <= a xor b;
end comp;

```

Part II

- (a) \$11.25
- (b) \$45.50
- (c) \$42
- (d) 9.375%
- (e) 28.125%
- (f) 37.5%
- (g) binary: 010000111 decimal: 1.0546875

Part III

```

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

entity grandtotalcalc is

  port (
    clk      : in  std_logic;
    go       : in  std_logic;
    taxrate  : in  std_logic_vector(3 downto 0);
    subtotal : in  std_logic_vector(7 downto 0);
    grandtotal, add : out std_logic_vector(7 downto 0);
    done     : out std_logic;
  calc      : out std_logic);

end grandtotalcalc;

architecture top of grandtotalcalc is

  component salestaxcalc
    port (
      clk      : in  std_logic;
      go       : in  std_logic;
      taxrate  : in  std_logic_vector(3 downto 0);
      subtotal : in  std_logic_vector(7 downto 0);
      tax      : out std_logic_vector(11 downto 0); -- done
      done     : out std_logic);

```

```

end component;

signal tax : std_logic_vector(11 downto 0);
signal sub, tax_add : std_logic_vector(7 downto 0);
signal done_calc : std_logic;
begin

  st : salestaxcalc port map (
    clk      => clk,
    go       => go,
    taxrate  => taxrate,
    subtotal => subtotal,
    tax      => tax,
    done     => done_calc);

  calc <= done_calc;
  tax_add <= "000" & tax(11 downto 7);
  add <= tax_add;
  process(go, done_calc)
  begin
  if rising_edge(clk) then
    if (go = '1') then
      grandtotal <= "00000000";
    sub <= subtotal;
    done <= '0';
    elsif (done_calc = '1') then
      grandtotal <= sub + tax_add;
      done <= '1';
    end if;
  end if;
end process;
end top;

```

Part IV

Since the subtotal is a negative number, first invert the lower 7 significant bits and then add 1. Then append a '0' in front of the 7 bit number before sending it into the cash register. The same is done for the grand total, but append a '1' in front instead of a '0'.

