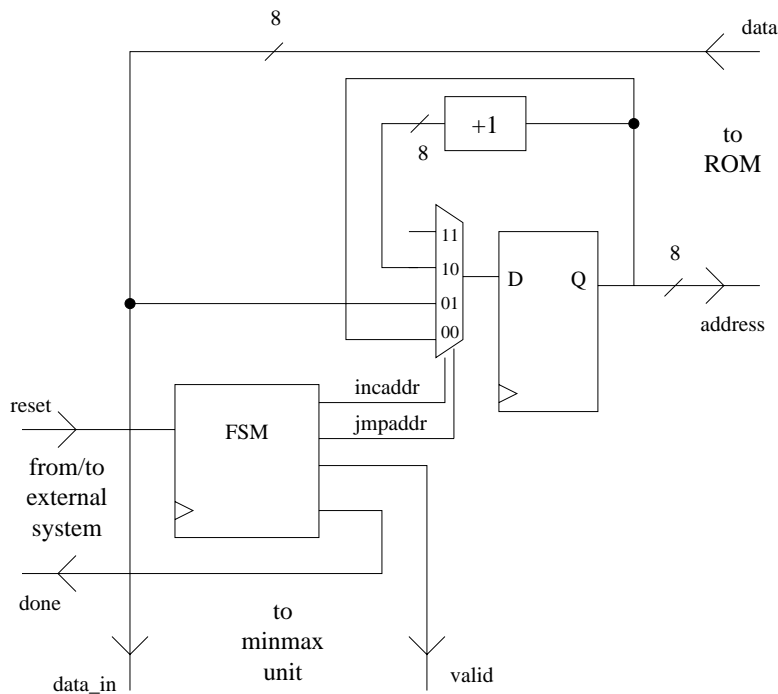


Massachusetts Institute of Technology
 Department of Electrical Engineering and Computer Science
 6.111 – Introductory Digital Systems Laboratory

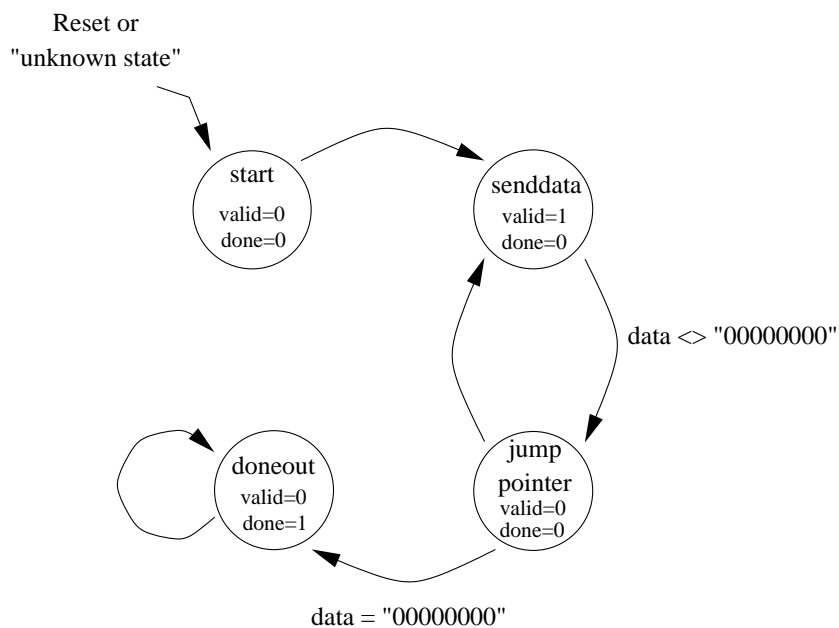
Problem Set 5 Solutions

Issued: March 21, 2003

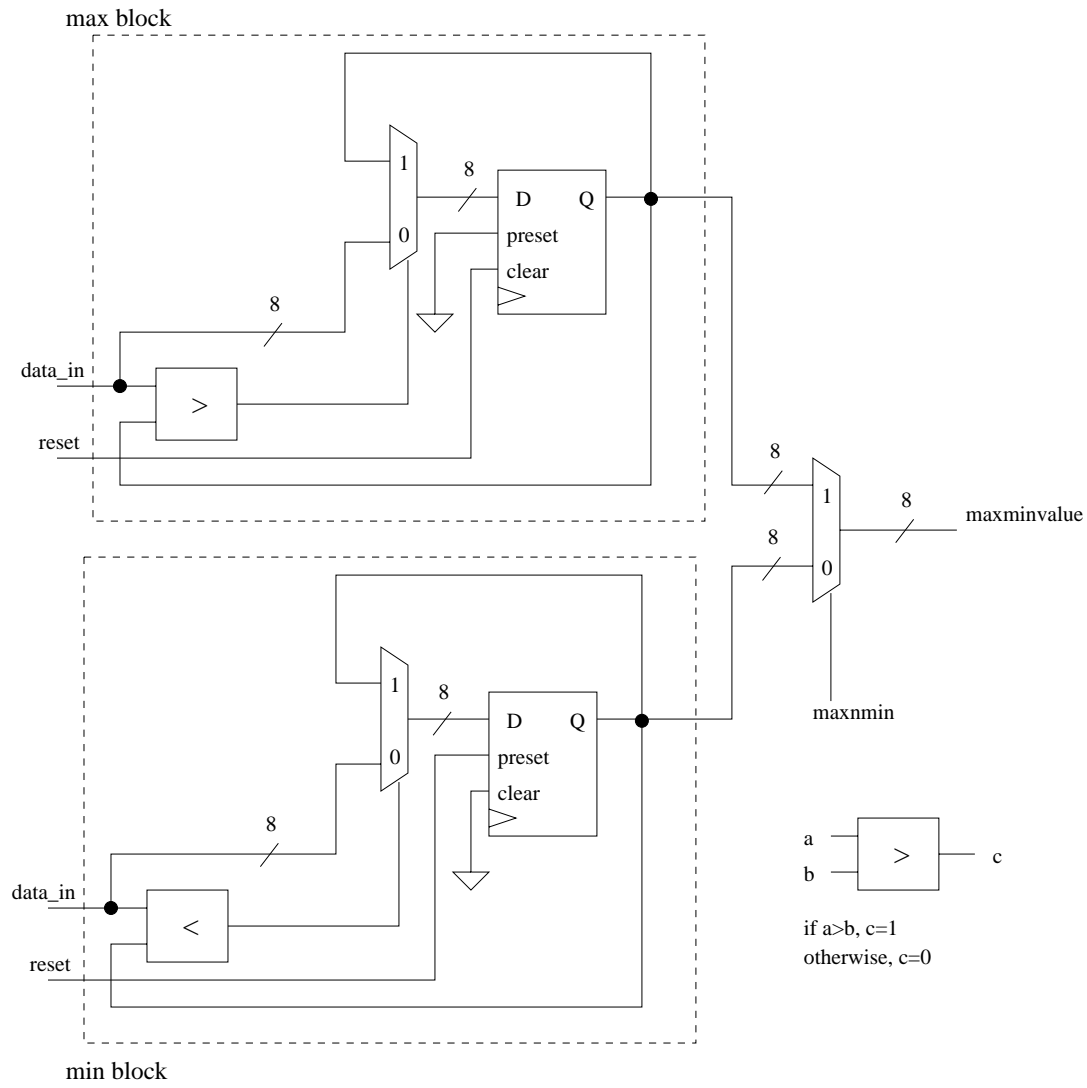
Block diagram for addressunit



FSM for addressunit



Block diagram for minmaxunit



addressunit.vhd

```

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

entity addressunit is port(
    clk          : in std_logic;
    reset        : in std_logic;
    data         : in std_logic_vector(7 downto 0);
    address      : out std_logic_vector(7 downto 0);
    data_in      : out std_logic_vector(7 downto 0);
    valid        : out std_logic;
    done         : out std_logic);
end addressunit;

architecture addressarch of addressunit is
    signal incaddr      : std_logic;
    signal jmpaddr      : std_logic;
    signal curaddr      : std_logic_vector(7 downto 0);
    signal present_state : std_logic_vector(1 downto 0);
    signal next_state   : std_logic_vector(1 downto 0);

    constant start      : std_logic_vector(1 downto 0) := "00";
    constant senddata   : std_logic_vector(1 downto 0) := "01";
    constant jumppointer : std_logic_vector(1 downto 0) := "10";
    constant doneout    : std_logic_vector(1 downto 0) := "11";

begin
    data_in <= data;
    address <= curaddr;

    clocked:process(clk)
    begin
        if reset='1' then
            present_state <= start;
        elsif rising_edge(clk) then
            present_state <= next_state;

            -- need to handle clocked signal, address
            if incaddr = '1' then
                curaddr <= curaddr + 1;
            elsif jmpaddr = '1' then
                curaddr <= data;
            else
                curaddr <= curaddr;
            end if;
        end if;
    end process;

    statecomb:process(present_state, data)
    begin
        case present_state is
            when start =>
                -- set up first pointer
                incaddr <= '0';
                jmpaddr <= '0';
        end case;
    end process;
end architecture addressarch;

```

```

    done <= '0';
    valid <= '0';
    next_state <= senddata;
when senddata =>
    -- curaddr will in incremented at the end of the cycle
    -- old data will be sampled at next clock edge
    incaddr <= '1';
    jmpaddr <= '0';
    done <= '0';
    valid <= '1';
    next_state <= jumppointer;
when jumppointer =>
    incaddr <= '0';
    jmpaddr <= '1';
    done <= '0';
    valid <= '0';
    if data = "00000000" then
        next_state <= doneout;
    else
        next_state <= senddata;
    end if;

when doneout =>
    incaddr <= '0';
    jmpaddr <= '0';
    done <= '1';
    valid <= '0';
    next_state <= doneout;
when others =>
    incaddr <= '0';
    jmpaddr <= '0';
    done <= '0';
    valid <= '0';
    next_state <= start;
end case;
end process;
end addressarch;

```

minmaxunit.vhd

```

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

entity minmaxunit is port(
    clk          : in std_logic;
    reset        : in std_logic;
    data_in      : in std_logic_vector(7 downto 0);
    valid        : in std_logic;
    maxnmin      : in std_logic;
    maxminvalue  : out std_logic_vector(7 downto 0));
end minmaxunit;
architecture minmaxarch of minmaxunit is
    signal min   : std_logic_vector(7 downto 0);
    signal max   : std_logic_vector(7 downto 0);
begin

```

```

maxminvalue <= max when (maxnmin = '1') else min;

clocked:process(clk)
begin
  if reset='1' then
    min <= "11111111";
    max <= "00000000";
  elsif rising_edge(clk) then
    if valid = '1' then
      if min > data_in then
        min <= data_in;
      end if;
      if max < data_in then
        max <= data_in;
      end if;
    end if;
  end if;
end process;
end minmaxarch;

```

Simulation for minmax

