```
architecture my_gcd of cmp_add_ctrl is
  -- enumeration type for states: "state"
  type state is (start,
                 read1, read2, load_l, load_r,
                 finished,
                 store_load_0, store_load_1,
                 load_add_l_0);
  signal current_state, next_state: state;
begin
  seq: process(clk, reset)
  begin
    if reset = '1'
    then
      current_state <= start;
      req <= '1';
      ready <= '0';
    elsif rising_edge(clk)
    then
      current_state <= next_state;

      -- request for second operand of GCD while storing first in
      -- memory; request for first operand when finished with previous
      -- computation
      if (next_state = read1) or (next_state = finished)
      then
        req <= '1';
      else
        req <= '0';
      end if;

      if next_state = finished
      then
        ready <= '1';
      else
        ready <= '0';
      end if;
    end if;
  end process seq;

  new_state: process(current_state, equal, greater)
  begin
    case current_state is
      when start => next_state <= read1;
      when read1 => next_state <= read2;
      when read2 => next_state <= load_l;
      when load_l => next_state <= load_r;
      when load_r =>
        if equal = '1'
        then
          next_state <= finished;
        elsif greater = '1'
        then
          next_state <= store_load_0;
        else
          next_state <= load_add_l_0;
        end if;
      when finished => next_state <= read1;
      when store_load_0 =>
        if equal = '1'
        then
          next_state <= finished;
        elsif greater = '1'
        then
          next_state <= store_load_0;
        else
          next_state <= load_add_l_0;
        end if;
      when load_add_l_0 => next_state <= store_load_1;
      when store_load_1 =>
        if equal = '1'
        then
          next_state <= finished;
        elsif greater = '1'
        then
          next_state <= store_load_0;
        else
          next_state <= load_add_l_0;
        end if;
    end case;
  end process new_state;

  outputs: process(next_state)
  begin
    case next_state is
      when start =>
        -- no meaningful activity in data path; all signals don't care
        add_l_sel <= '-'; add_l_en <= '-'; add_r_sel <= '-'; add_r_en <= '-';
        sub <= '-';
        cmp_l_sel <= '-'; cmp_l_en <= '-'; cmp_r_sel <= '-'; cmp_r_en <= '-';
        rd_addr <= "--"; wr_addr <= "--"; wr_sel_en <= "--";
      when read1 =>
        -- copy from data_in to memory address 0;
        -- read addr is set to 00 because current state may be "finished"
        -- and ready may therefore be high
        add_l_sel <= '-'; add_l_en <= '-'; add_r_sel <= '-'; add_r_en <= '-';
        sub <= '-';
        cmp_l_sel <= '-'; cmp_l_en <= '-'; cmp_r_sel <= '-'; cmp_r_en <= '-';
        rd_addr <= "00"; wr_addr <= "00"; wr_sel_en <= "11";
      when read2 =>
        -- copy from data_in to memory address 1; rest is don't care
        add_l_sel <= '-'; add_l_en <= '-'; add_r_sel <= '-'; add_r_en <= '-';
        sub <= '-';
        cmp_l_sel <= '-'; cmp_l_en <= '-'; cmp_r_sel <= '-'; cmp_r_en <= '-';
        rd_addr <= "--"; wr_addr <= "01"; wr_sel_en <= "11";
      when load_l =>
        -- copy from memory address 0 to cmp_l register
        add_l_sel <= '0'; add_l_en <= '1'; add_r_sel <= '-'; add_r_en <= '-';
        sub <= '-';
        cmp_l_sel <= '0'; cmp_l_en <= '1'; cmp_r_sel <= '-'; cmp_r_en <= '-';
        rd_addr <= "00"; wr_addr <= "--"; wr_sel_en <= "00";
      when load_r =>
        -- copy from memory address 1 to cmp_r register
        add_l_sel <= '-'; add_l_en <= '0'; add_r_sel <= '0'; add_r_en <= '1';
        sub <= '-';
        cmp_l_sel <= '-'; cmp_l_en <= '0'; cmp_r_sel <= '0'; cmp_r_en <= '1';
        rd_addr <= "01"; wr_addr <= "--"; wr_sel_en <= "00";
      when finished =>
        -- next state is "finished"; "ready" is not yet high;
        -- read address is don't care
        add_l_sel <= '-'; add_l_en <= '-'; add_r_sel <= '-'; add_r_en <= '-';
        sub <= '-';
        cmp_l_sel <= '-'; cmp_l_en <= '-'; cmp_r_sel <= '-'; cmp_r_en <= '-';
        rd_addr <= "--"; wr_addr <= "--"; wr_sel_en <= "00";
      when store_load_0 =>
        -- subtract and copy result of subtraction to memory address 0
        add_l_sel <= '1'; add_l_en <= '1'; add_r_sel <= '-'; add_r_en <= '0';
        sub <= '1';
        cmp_l_sel <= '1'; cmp_l_en <= '1'; cmp_r_sel <= '-'; cmp_r_en <= '0';
        rd_addr <= "--"; wr_addr <= "00"; wr_sel_en <= "01";
      when load_add_l_0 =>
        -- copy from memory address 1 to add_l
        add_l_sel <= '0'; add_l_en <= '1'; add_r_sel <= '1'; add_r_en <= '1';
        sub <= '1';
        cmp_l_sel <= '-'; cmp_l_en <= '-'; cmp_r_sel <= '-'; cmp_r_en <= '-';
        rd_addr <= "10"; wr_addr <= "--"; wr_sel_en <= "00";
      when store_load_1 =>
        -- subtract and copy result of subtraction to memory address 1
        add_l_sel <= '0'; add_l_en <= '1'; add_r_sel <= '1'; add_r_en <= '1';
        sub <= '1';
        cmp_l_sel <= '0'; cmp_l_en <= '1'; cmp_r_sel <= '1'; cmp_r_en <= '1';
        rd_addr <= "00"; wr_addr <= "01"; wr_sel_en <= "01";
    end case;
  end process outputs;
end my_gcd;
```