1. Consider the synchronous ripple carry counter on page 5-8 of the notes. Assume that the flip flops have a setup time requirement of 2 ns and that the gates in the circuit all have a propagation delay of 1 ns. If the enable input changes from 0 to 1 at time 0, what is the earliest time that the clock can change from 0 to 1 without violating setup time requirements, if the initial value of the counter is 1? What if the initial value of the counter is 3? What if it is 7? If we extend the counter to 16 bits, how much time must there be between a change to the enable input and the next rising clock edge if we are to avoid setup time violations? If the flip flops have a hold time requirement of 1 ns and the maximum clock skew is 0.5 ns, what is the fastest clock frequency at which you can use this counter, without violating setup or hold time constraints?

If the counter value is 1 initially, then a change to the enable input will ripple through to the bit 1 flip flop, but not to flip flops 2 and 3. So, if the clock rises 4 ns after the enable changes, we can still meet the setup time constraint of 2 ns. Otherwise, we will get a violation. If the initial value is 3, then there must be a delay of 5 ns from a change to the enable to the clock edge. If the initial value is 7, the required delay is 6 ns.

For a 16 bit counter, we will need 18 ns in the worst-case (which occurs when the counter bits are all 1).

To determine the shortest clock period, note that the worst-case situation arises when clock to bit 0 of the counter occurs as late as possible, relative to the clock going to bit 15 of the counter. Suppose the clock rises at time 0 at bit 15 and it rises 0.5 ns later at bit 0. Then the enable can rise at time 0.5 without violating any hold time constraints (but not before). Now the carry must propagate to bit 15 of the counter before the next rising clock edge at bit 15. So the data input of the bit 15 flip flop will be stable at time 0.5+16=16.5. Adding in the setup time requirement gives 18.5, so the shortest clock period we can have is 18.5 ns and the maximum clock frequency is 54 MHz.
2. Write a VHDL specification for a simple synchronous binary counter with 4 bits, an enable input, a clock input and a carry_out. Do a unit delay simulation with the simulation precision set to 1 ns. Your simulation should sequence through all values of the counter and at each value, keep the enable low for one clock period and then raise it high for one clock period (causing the counter to increment to the next value). What is the worst-case delay that you get from when the enable changes to when the carry_out is stable? Is this delay dependent on the value of the counter? What does this tell you about the circuit that the synthesizer created from your VHDL specification?

The simulation output appears below. The worst-case delay is 6 ns. The carry_out only goes high when the counter value is all 1s and the enable is high. The fact that the delay is 6 ns suggests that the synthesizer created a circuit using a ripple carry chain, since a circuit with a lookahead carry circuit would have a delay of just one or two gate delays.
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

everity counter is
  port (  
    clk, en: in STD_LOGIC;
    c_out: out STD_LOGIC;
    d: out STD_LOGIC_VECTOR (3 downto 0) 
  );
end counter;

architecture counter_arch of counter is
  signal reg: STD_LOGIC_VECTOR(3 downto 0);
begin
  process (clk) begin
    if clk'event and clk = '1' then
      if en = '1' then
        reg <= reg + "0001";
      end if;
    end if;
  end process;
  d <= reg;
  c_out <= '1' when en = '1' and reg = "1111" else '0';
end counter_arch;
3. Design a 3 digit BCD counter. This counter will have 4 groups of flip flops, with each group representing a single BCD digit. Section 5-6 describes how to design the circuit for each group of 4 flip flops. Augment this design to generate a carry_out signal that can be used to link the different BCD digits together. Implement your design using the schematic editor and simulate it.
4. Mano and Kime 6-4. A 32K×8 RAM chip uses coincident decoding by splitting the internal decoder into row select and column select. (a) Assuming that the RAM cell array is square, what is the size of each decoder, and how many AND gates are required for decoding an address? (b) Determine the row and column selection lines that are enabled when the input address is the binary equivalent of 2100010.

(a) Since the RAM has 256K bits, the array will have 512 rows and 512 columns. So the row decoder will decode 9 bits and the column decoder will decode 6 bits. The number of AND gates needed in the two decoders is 512+64=576.

(b) 21,000 = 101 0010 0000 1000 in binary. Assuming the high order address bits go to the row decoder then row select line 14816=32810 will be active. Also, column select line 8 will be active. If the low order address bits go to the row decoder, then row select line 8 will be active and column select line 41 will be active.

5. Mano and Kime 6-8. A DRAM has a refresh interval of 128 ms and has 4096 rows. What is the interval between refreshes for distributed refresh? What is the minimum number of address pins on the DRAM?

The interval between refreshes is 128/4096 = .03125 ms or 31.25 µs.

The minimum number of address pins that such a RAM can have is 12. However, if it had 4096 columns, as well as 4096 rows, it would need to have a 4096 bit wide word if it just had 12 address pins.