1. Consider the sequential circuit shown below. Assume that the flip flops have a setup time requirement of 2 ns, a hold time requirement of 1 ns, a minimum propagation delay of 1 ns and a maximum of 3 ns. Also, assume that the maximum clock skew in the circuit is 2 ns and that all gates have a minimum propagation delay of 1 ns and a maximum propagation delay of 3 ns. If this circuit is operated at 50 MHz, will all the setup time constraints be satisfied? Explain. What about if the circuit is operated at 100 MHz? What is the maximum clock frequency for which all the setup time constraints are satisfied? Does the circuit satisfy all hold time requirements? What is the largest clock skew for which the circuit satisfies all hold time requirements?

At 50 Mhz, the clock period is 20 ns. So, to meet setup time requirements, we need to ensure that the worst-case flip flop propagation delay, plus the worst-case circuit delay, plus the maximum clock skew plus the setup time is no more than 20 ns. Here, the sum of these values is 3+9+2+2=16 ns (9 ns is the maximum circuit delay, which is for the circuit path from the output of the bottom flip flop back to its input). Since this is less than 20 ns, the circuit will meet all setup time constraints at 50 Mhz. However, it will not meet them at 100 MHz, since here the clock period is just 10 ns. The maximum clock frequency for which setup requirements are met is 62.5 MHz.

The hold time requirements are not all satisfied. In particular, if we have the minimum propagation delay through the bottom flip flop and the circuit path to the top flip flop, we get just 2 ns, which is less than the hold time requirement plus the maximum clock skew (these sum to 3 ns). In order to satisfy the hold time requirements, we would need to reduce the clock skew to 1 ns. Or, we could increase the circuit delay along the path from the output of the second inverter to the input of the first, by adding a pair of inverters.
2. Write a VHDL specification for the sequential circuit from problem 5 of homework 7 using a process statement. Simulate your program and turn in the VHDL listing and simulator output.

library IEEE;
use IEEE.std_logic_1164.all;

entity mod3 is
   port (d, en, clk : in STD_LOGIC;
         z: out STD_LOGIC);
end mod3;

architecture mod3_arch of mod3 is
begin
   process(clk) begin
      if clk'event and clk = '1' then
         if en = '0' then
            state <= div0;
         else
            if state = div0 and d = '1' then
               state <= div1;
            elsif state = div1 and d = '1' then
               state <= div2;
            elsif state = div2 and d = '1' then
               state <= div0;
            end if;
         end if;
      end if;
   end process;
   z <= '1' when state = div0 else '0';
end mod3_arch;
1. Write a VHDL specification for a serial adder/subtractor. Your circuit will have two data inputs, $A$ and $B$, a clock input $CLK$, an enable input $EN$, a mode input $M$, and a data output $Z$. When the enable input is low, $Z=0$. When the enable input is high and $M=0$, the data inputs are added together. The incoming values arrive with a new bit at every rising clock edge, with the least significant bits first. The sum bits are sent out serially, with a new bit on each clock tick. When $M=1$, the circuit subtracts $B$ from $A$, instead of adding the values together.

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;

entity addsub is
    port (
        clk, en, a, b, m: in STD_LOGIC;
        z: out STD_LOGIC
    );
end addsub;

architecture addsub_arch of addsub is
    signal carry_borrow: std_logic;
begin
    process(clk) begin
        if clk'event and clk = '1' then
            if en = '0' then
                carry_borrow <= '0';
            elsif m = '0' then
                carry_borrow <= (a and b) or 
                    (a and carry_borrow) or 
                    (b and carry_borrow);
            elsif m = '1' then
                carry_borrow <= ((not a) and b) or 
                    ((not a) and carry_borrow) or 
                    (b and carry_borrow);
            end if;
        end if;
    end process;
    z <= en and (a xor b xor carry_borrow);
end addsub_arch;
```
4. (MK 5-4) The content of a 4 bit register is initially 1111. The register is shifted eight times to the left, with the sequence 00101011 as the serial input. The leftmost bit of the sequence is applied first. What is the content of the register after each shift?

After the first shift it’s 1110, then 1100, 1001, 0010, 0101, 1010, 0101, 1011.

5. (MK 5-11) Draw the logic diagram of a 4 bit register with mode selection inputs $S_1$ and $S_0$. When $S_1S_0=00$, there is to be no change in the register contents. When $S_1S_0=01$, the value of the register is to be changed to zero. When $S_1S_0=10$ the register is to load a new value. When $S_1S_0=11$, the value is to be complemented. Your logic diagram should use only edge-triggered $D$ flip flops and simple gates (AND, OR, NOT). Do NOT gate the clock signal.