Union College
ECE318
HW5 Solutions

Problem 1

This is a positive edge triggered 8b register and there is an active-high enable. The characteristic table is

Enable	Clk	$D_{7..0}$	$Q_{next[7..0]}$
0	X	X	$Q_{7..0}$
1	1	X	$Q_{7..0}$
1	0	X	$Q_{7..0}$
1	↑	$D_{7..0}$	$D_{7..0}$

Problem 2

I will be including two versions of the VHDL code. The first is a solution that does not work with our current hardware and version of Quartus. The second version does work correctly with our current DE2 boards and Quartus II vers 9.1 SP2. Please note the differences and we will be talking about them in class.

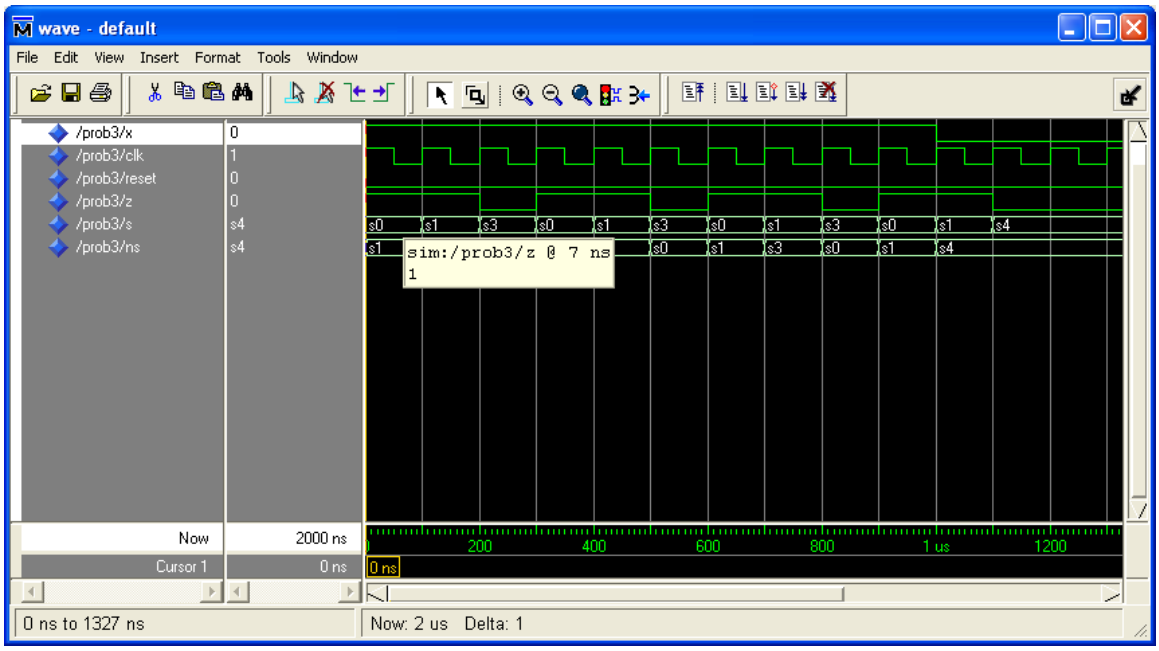
Code (Does not synthesize correctly in Quartus 9.1 SP2):

```
library ieee;
use ieee.std_logic_1164.all;

entity prob3 is
        port( x, clk, reset : in std_logic;
                        z : out std_logic);
end prob3;
```

```
architecture seq of prob3 is
type state is ( S0, S1, S2, S3, S4, S5 );
signal s, ns : state;
begin

        ns_logic: process (x,s)
        begin
                case s is
                        when S0 =>
                                if (x = '1') then
                                        ns <= S1;
                                else
                                        ns <= S2;
                                end if;
                        when S1 =>
                                if (x = '1') then
                                        ns <= S3;
                                else
                                        ns <= S4;
                                end if;
                        when S2 =>
                                if (x = '1') then
                                        ns <= S3;
                                else
                                        ns <= S2;
                                end if;
                        when S3 =>
                                if (x = '1') then
                                        ns <= S0;
                                else
                                        ns <= S3;
                                end if;
                        when S4 =>
                                if (x = '1') then
                                        ns <= S3;
                                else
                                        ns <= S4;
                                end if;
                        when others =>
                          null;
                end case;
        end process ns_logic;

        Z <= '0' when s = S3 or s = S4 else '1';

        ff: process (clk)
```

```
        begin
                if reset = '1' then
                        s <= S0;
                elsif rising_edge(clk) then
                        s <= ns;
                end if;
        end process ff;

end architecture seq;
```

Simulation using ModelSim:



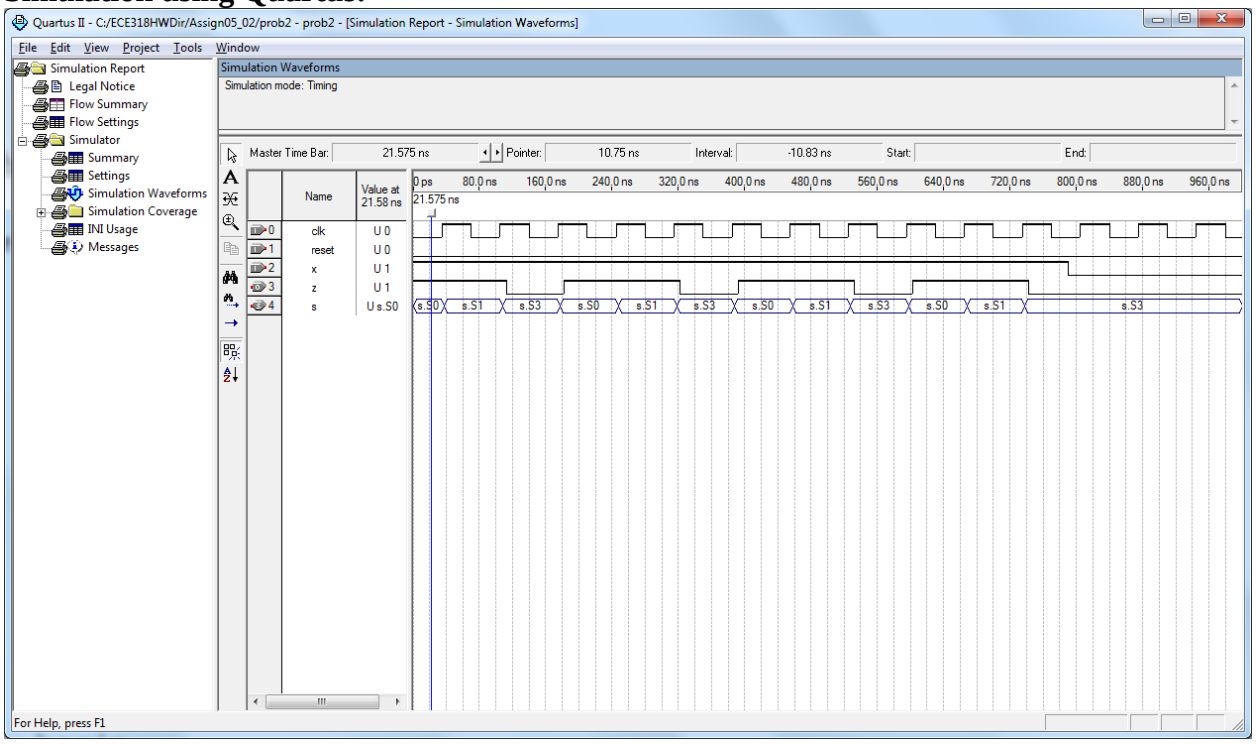
Problem 2 (code works with Quartus II vers 9.1 SP 2)

```
library ieee;
use ieee.std_logic_1164.all;

entity prob2 is
        port( x, clk, reset : in std_logic;
                        z : out std_logic);
end prob2;

architecture seq of prob2 is
type STATE_TYPE is ( S0, S1, S2, S3, S4, S5 );
signal s, ns : STATE_TYPE;
begin

        ns_logic: process (x,s)
```

```
begin
 -- if (clk'EVENT AND clk = '1') THEN
        case s is
                when S0 =>
                        if (x = '1') then
                                ns <= S1;
                        else
                                ns <= S2;
                        end if;
                when S1 =>
                        if (x = '1') then
                                ns <= S3;
                        else
                                ns <= S4;
                        end if;
                when S2 =>
                        if (x = '1') then
                                ns <= S3;
                        else
                                ns <= S2;
                        end if;
                when S3 =>
                        if (x = '1') then
                                ns <= S0;
                        else
                                ns <= S3;
                        end if;
                when S4 =>
                        if (x = '1') then
                                ns <= S3;
                        else
                                ns <= S4;
                        end if;
                when others =>
                   ns <= s0;
        end case;
--      END if;
end process ns_logic;

Z <= '0' when s = S3 or s = S4 else '1';

ff: process (clk, reset)
begin
        if reset = '1' then
                s <= S0;
        elsif rising_edge(clk) then
```

```
                s <= ns;
            end if;
        end process ff;

end architecture seq;
```

Simulation using Quartus:



Problem 3

Code:

Counter

```
library ieee;
use ieee.std_logic_1164.all;

entity counter is
     port(clk : in std_logic;
          reset : in std_logic;
          count : out std_logic_vector(2 downto 0));
end entity counter;

architecture seq of counter is
begin
```

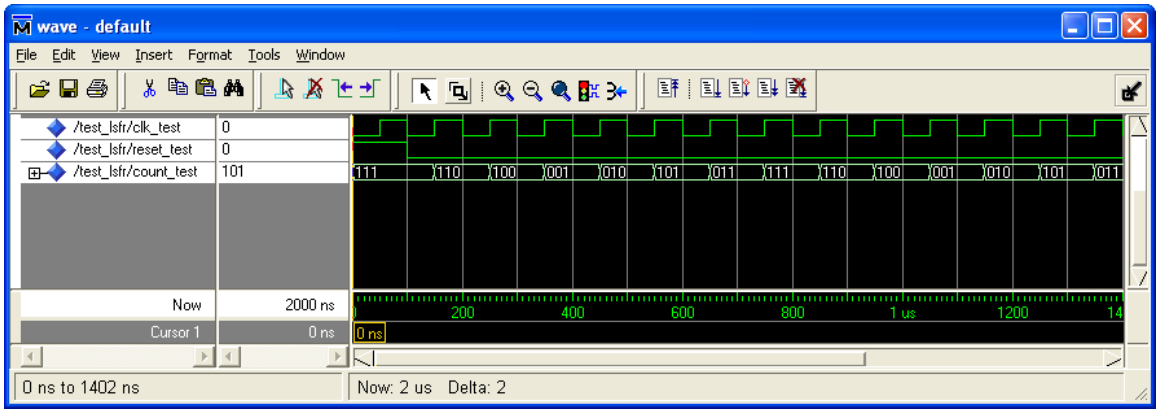
```
          p0: process (clk,reset) is
               variable reg : std_logic_vector(2 downto 0);
          begin
               if reset = '1' then
                    reg := (others => '1');
               elsif rising_edge(clk) then
                    reg := reg(1 downto 0) & (reg(2) xor
reg(1));
               end if;
               count <= reg;
          end process p0;
     end architecture seq;
```

Testbench

```
     library ieee;
     use ieee.std_logic_1164.all;

     entity test_lsfr is
     end entity test_lsfr;

     architecture tb of test_lsfr is

     -- Component declaration
     component counter is
          port(clk : in std_logic;
                    reset : in std_logic;
                    count : out std_logic_vector(2 downto
     0));
     end component counter;

     -- Signal declaration
     signal clk_test, reset_test : std_logic;
     signal count_test : std_logic_vector(2 downto 0);

     begin
     -- Device under test

     DUT: counter port map ( clk_test, reset_test,
     count_test);

     tim: process is
         begin
             clk_test <= '0';
```

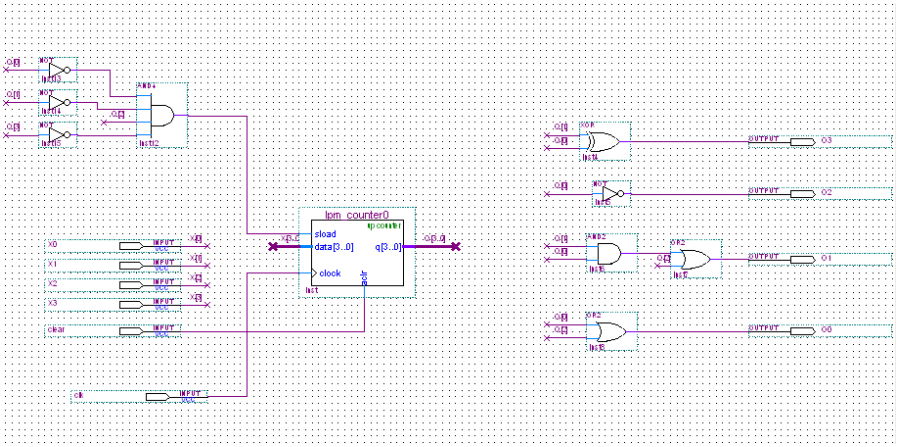
```
        wait for 50 NS;
           clk_test <= '1';
        wait for 50 NS;
    end process tim;

    rst: process is
    begin
        reset_test <= '1';
        wait for 100 NS;
        reset_test <= '0';
        wait;
    end process rst;


    end architecture tb;
```

Simulation:

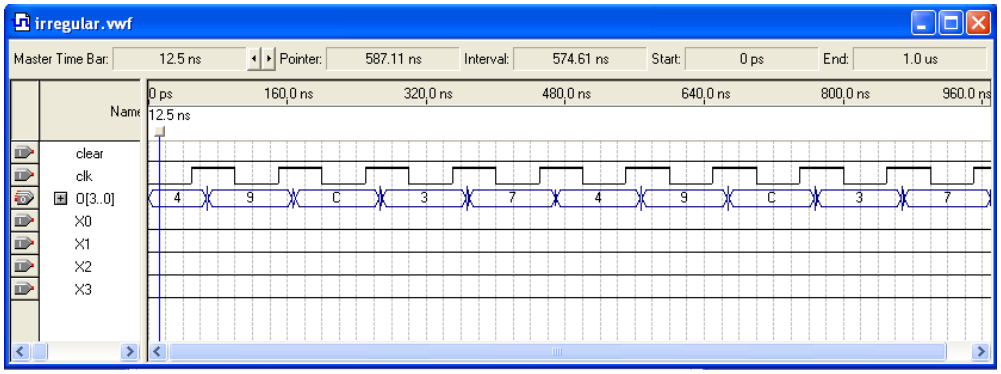


Problem 5 Results

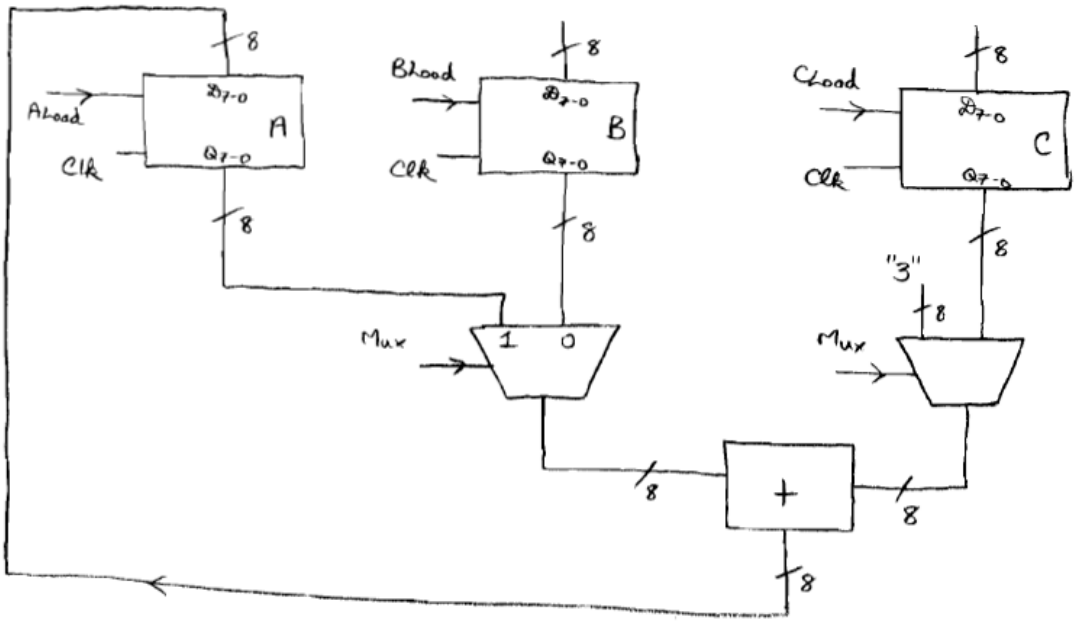
Block Diagram:



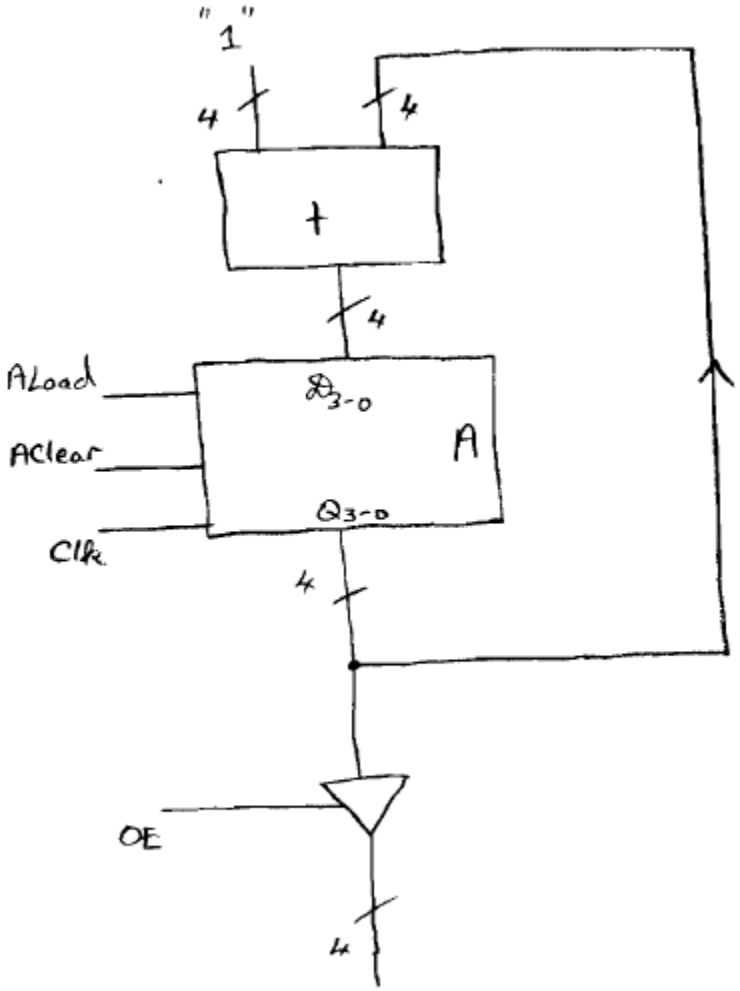
Simulation:



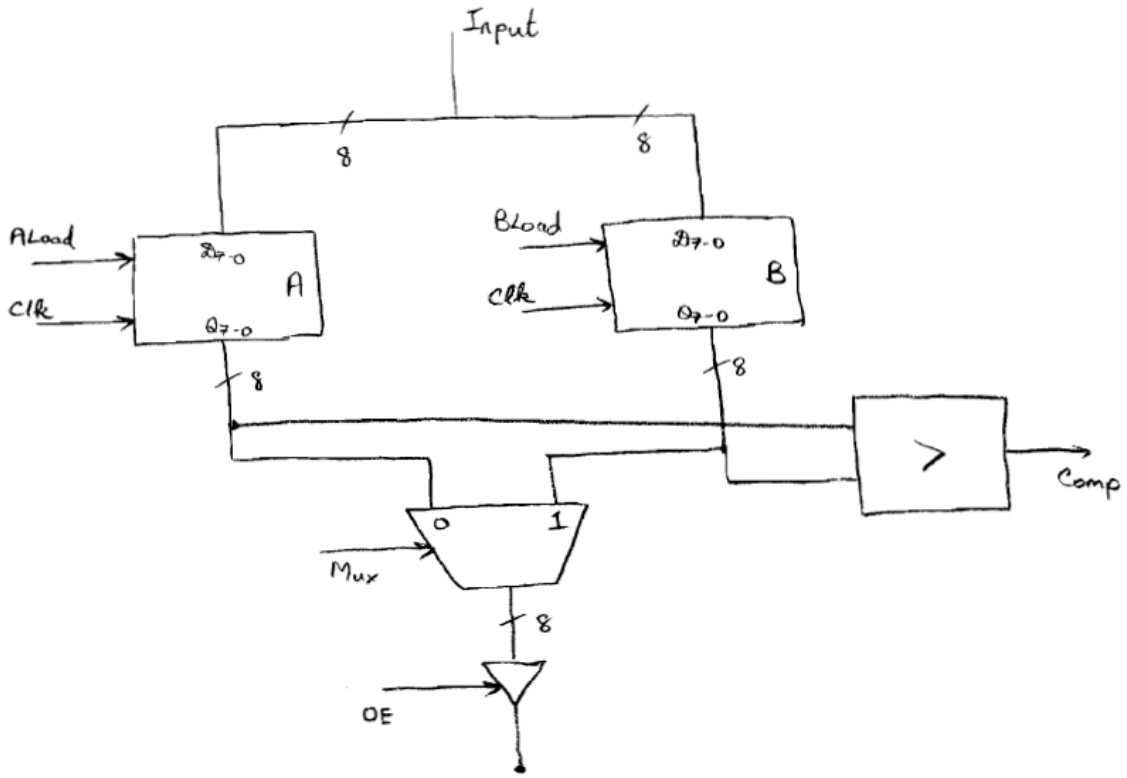
6. Design a datapath that can execute the two statements A=B+C and A=A+3 using just one adder. (Note that A, B, C are all 8 bits.)



7. Construct a 4 bit wide dedicated datapath to generate and output the numbers from 1 to 10. Indicate any inputs from an external source and a controller as well as any status signals generated in the datapath but there is no need to design the controller.



8. Design a dedicated datapath for inputting two 8-bit unsigned numbers (each presented one clock cycle after the other at the input port) and then output the larger number. The datapath should have only one input port and one output port. Label clearly all of the control and status signals (You can assume that the "greater than" comparison block is given to you. The LPM_COMPARE megafunction in Quartus could be used to generate this output.)



Controller :-

Loads A, then B, and next waits for the comp signal [comp = 1 ⇒ A > B ; comp = 0 ⇒ B > A].

If comp = 1, MUX i/p 0 is selected, otherwise MUX i/p 1 is selected. The output is enabled.