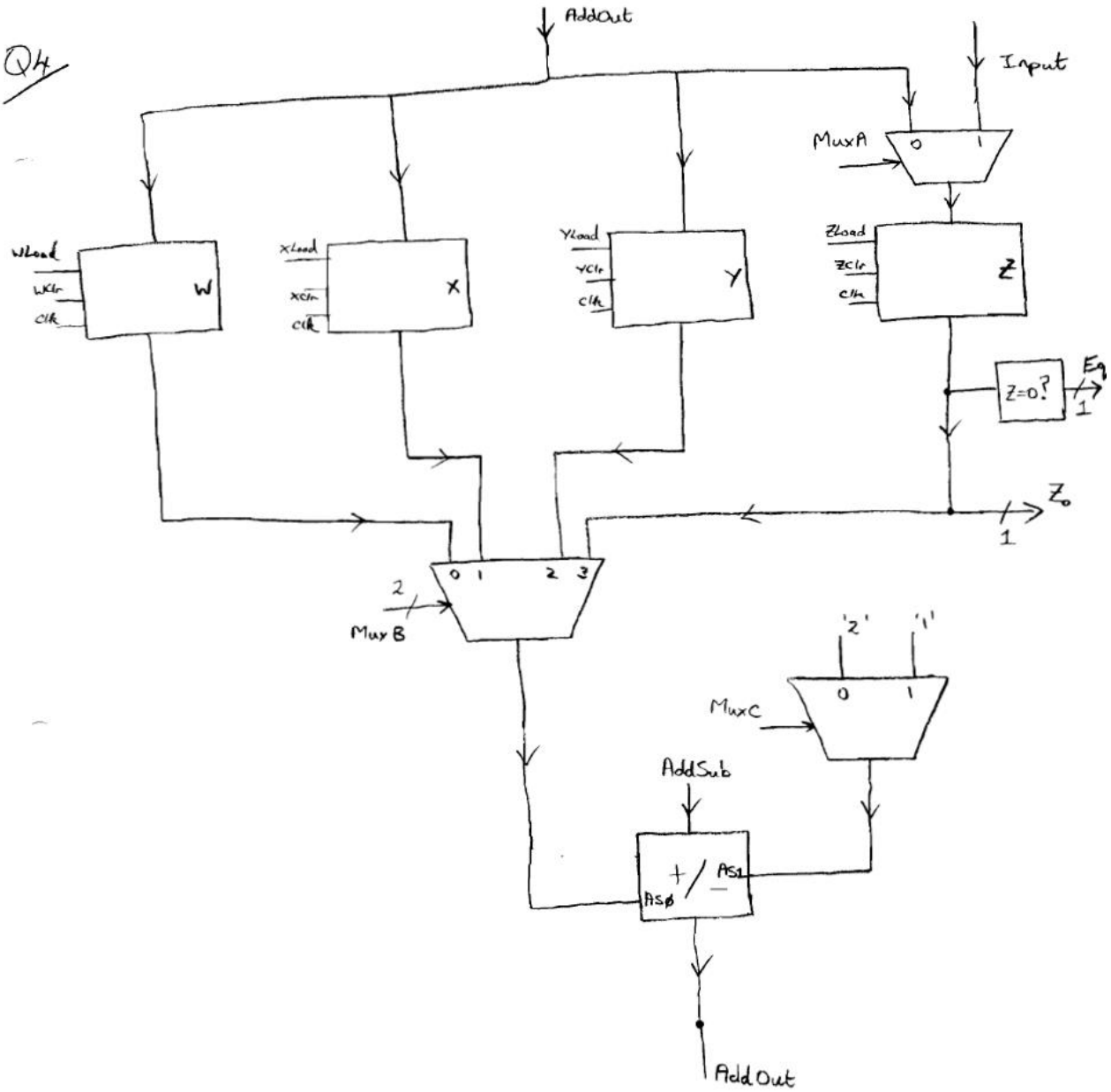
Union College
ECE318/CSC318
Assignment 6 Solution

Note: submit both the code and the simulation output for each problem requiring simulation.

Problem 1 Design an 8-bit dedicated datapath for the following algorithm and write the control words for it. Use only one adder-subtractor unit for all of the addition and subtraction operations. Label clearly all of the control and status signals

```
W=0

X=0

Y=0

INPUT Z

WHILE (Z /= 0) {

        W = W – 2

        IF (Z is an odd number) THEN

                X = X + 2

        ELSE

                Y = Y + 1

        END IF

        Z = Z - 1

}
```

Q4



STATUS SIGNALS :-

Z_0 : bit 0 of Z

Eq : 1 if z = 0 ; 0 otherwise

Note:

All lines 8 bits unless otherwise marked

Control Signals :-

WLoad, XLoad, YLoad, ZLoad : load W, X, Y, Z

WClr, XClr, YClr, ZClr : clear W, X, Y, Z

Mux A : controls i/p to Z

{ 0 - take i/p from adder output (AddOut)
{ 1 - " " " input port

Mux B :

{ 00 - pass W to adder
{ 01 - " X " "
{ 10 - " Y " "
{ 11 - " Z " "

Mux C :

{ 0 - pass 2 to adder
{ 1 - " 1 " "

AddSub :

{ 0 - AddOut = AS0 + AS1
{ 1 - AddOut = AS0 - AS1

Problem 2 A digital system has an 8-bit input, X, and an 11-bit output, Z, which is computed from X by the equation:

Z = 2X + 4(X-1) + 2(X-2)

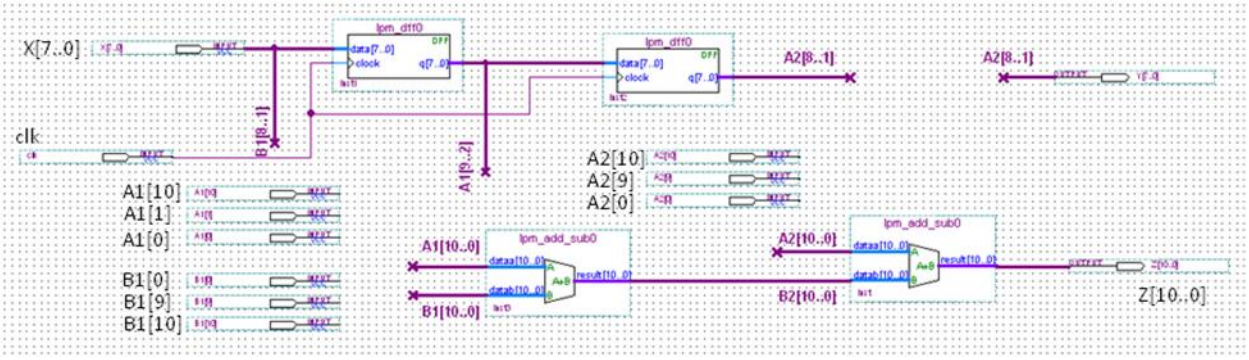
NOTE: (X-1) is the 8-bit value of X of the PREVIOUS CLOCK CYCLE. (X-2) is the 8-bit value of X of TWO CLOCK CYCLES AGO. A new X input is applied at every clock cycle, and after 2 clock cycles, a new Z output is generated every clock cycle.

a. If the X input sequence is the simple incrementing sequence shown below, give the Z output for each of the first 10 clock cycles. We assume the initial values of X-1 and X-2 are 0, and the first two are done for you.

clock cycle	X(binary)	Z(decimal)
1	00000001	2 + 0 + 0 = 2
2	00000010	4 + 4 + 0 = 8
3	00000011	6 + 8 + 2 = 16
4	00000100	
5	00000101	
6	00000110	
7	00000111	
8	00001000	
9	00001001	
10	00001010	

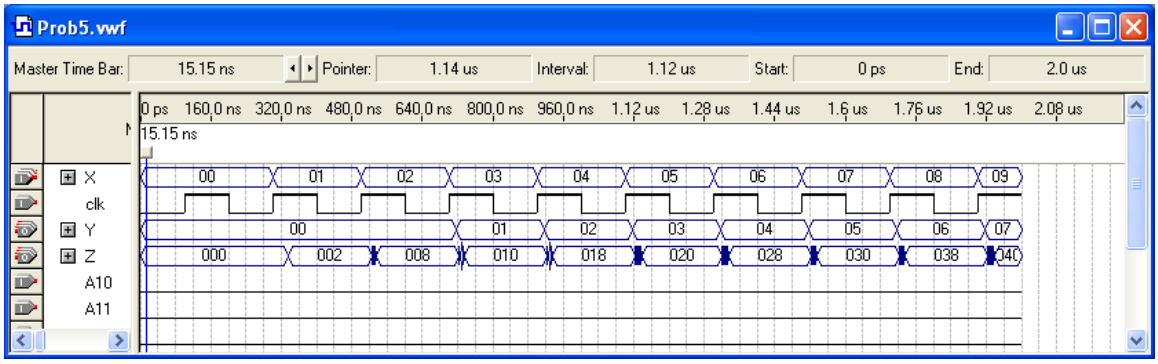
This circuit can be implemented with only a datapath, there is no controller required.

b. Draw a block diagram of the datapath. Clearly label both the ports of the components and the signals between components.



This is a clever way of solving the problem where we shift the output by 2 or 4 by simply renaming the bus lines and inserting zeros. This problem can also be solved by using multipliers but this will introduce more delay. Carefully review this solution to make sure that you understand how it works.

Simulation (you were not asked to do this but is included to show that it works):



Problem 3 Come up with a control unit for assignment 5 problem 7. You were not asked to implement and simulate this in Quartus but I have included this to show that it works.

The first step is to put the datapath block diagram together using the standard Quartus components from the library. This is shown as figure 1 below. Then, the datapath is tested with some input waveforms to make sure that it does what we want it to do. The results are shown in figure 2.

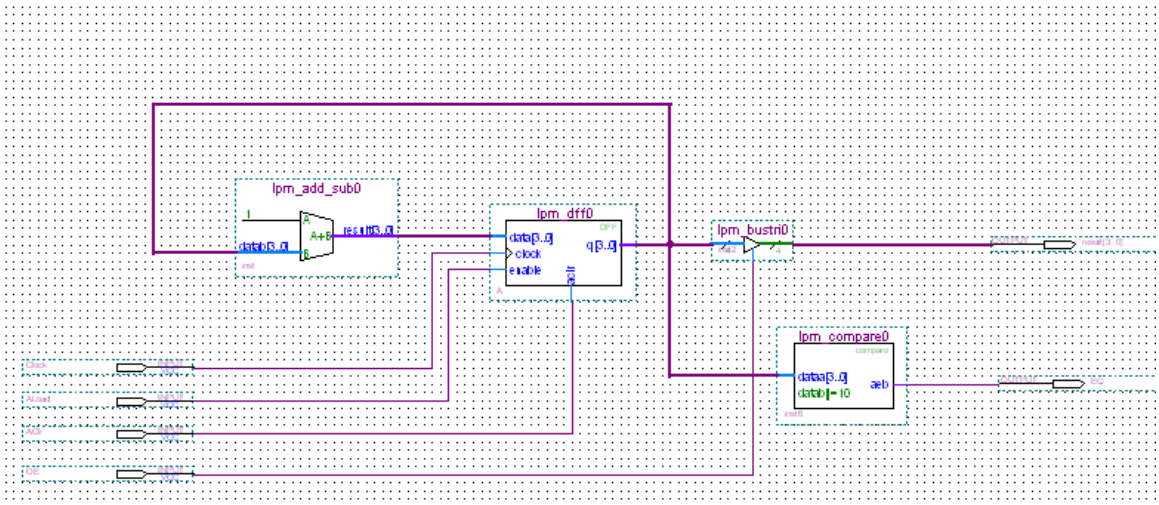


Figure 1. Datapath Block Diagram

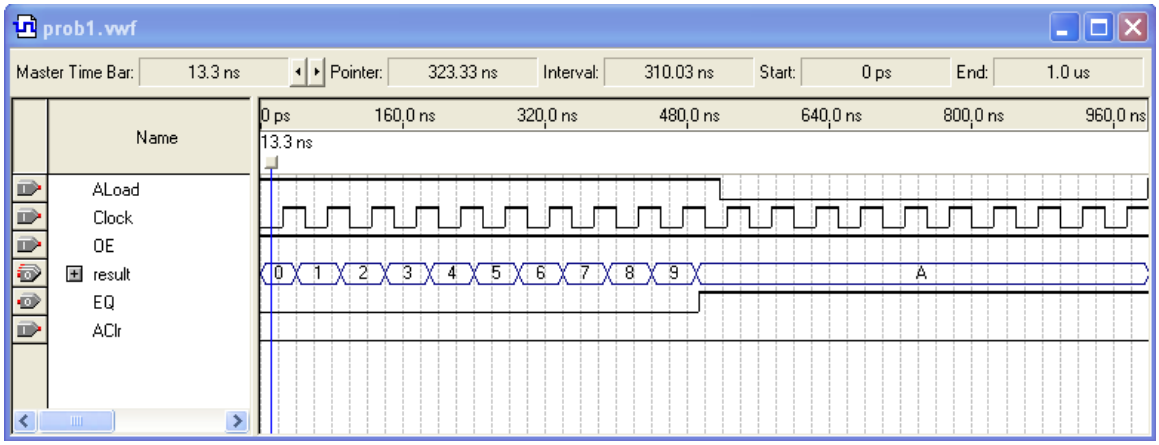


Figure 2. Simulation on Datapath

The controller must now be coded and so we use a simple state machine to determine the signals sent to the datapath. Here is one possible way of coding the state machine.

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

entity prob1_control is
port (clk,EQ,reset : in std_logic;
      AClr, ALoad, OE : out std_logic);
end entity prob1_control;

architecture seq of prob1_control is
type state_type is (S0, S1);
signal state, next_state : state_type;
begin

ctrl_ff: process (clk,reset) is
begin
      if reset = '1' then
            state <= S0;
      elsif rising_edge(clk) then
            state <= next_state;
      end if;
end process ctrl_ff;
```

```
ctrl: process (state,EQ)
begin
      AClr <= '0';
      ALoad <= '0';
      OE <= '0';

    case state is
           when S0 =>
                    AClr <= '1';
                next_state <= S1;

            when S1 =>

                        if (EQ = '0') then
                              ALoad <= '1';
                              OE <= '1';
                        else
                              ALoad <= '0';
                              OE <= '0';
                end if;
      end case;
  end process ctrl;

end architecture seq;
```

Figure 3. Controller State Machine

Again, it is a good idea to make sure that this is functioning as we expect it to and so the behavior is again simulated. The results are given in figure 4.

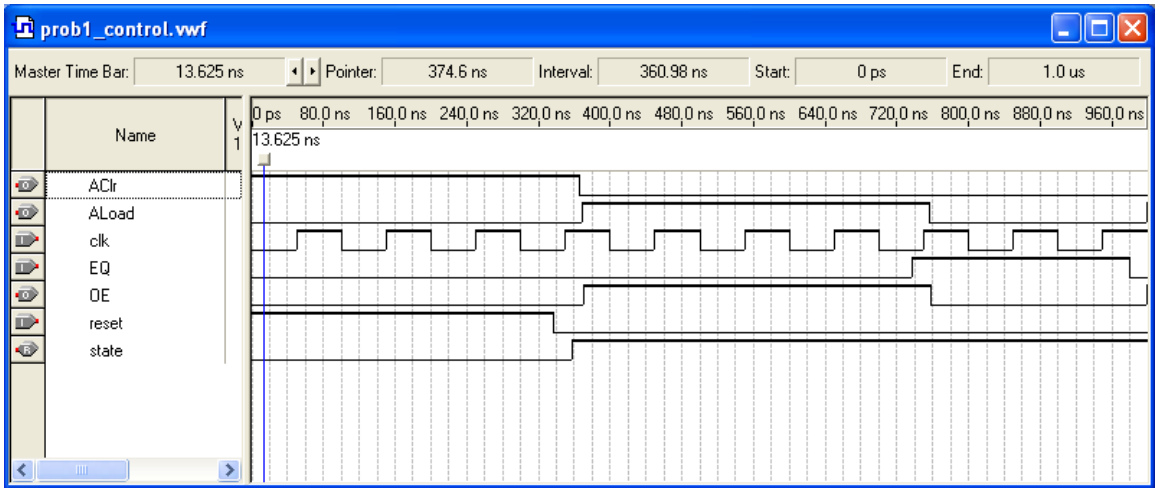


Figure 4. Controller Simulation Results

Finally, the controller is placed in the datapath circuit and the circuit is modified accordingly as shown in figure 5. The final simulation is then run and the results are given in figure 6.

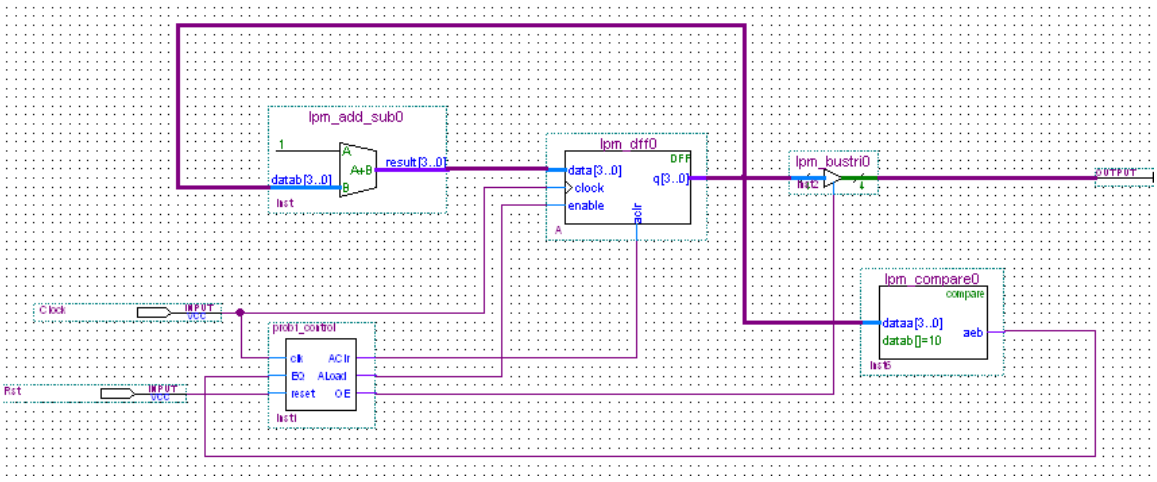
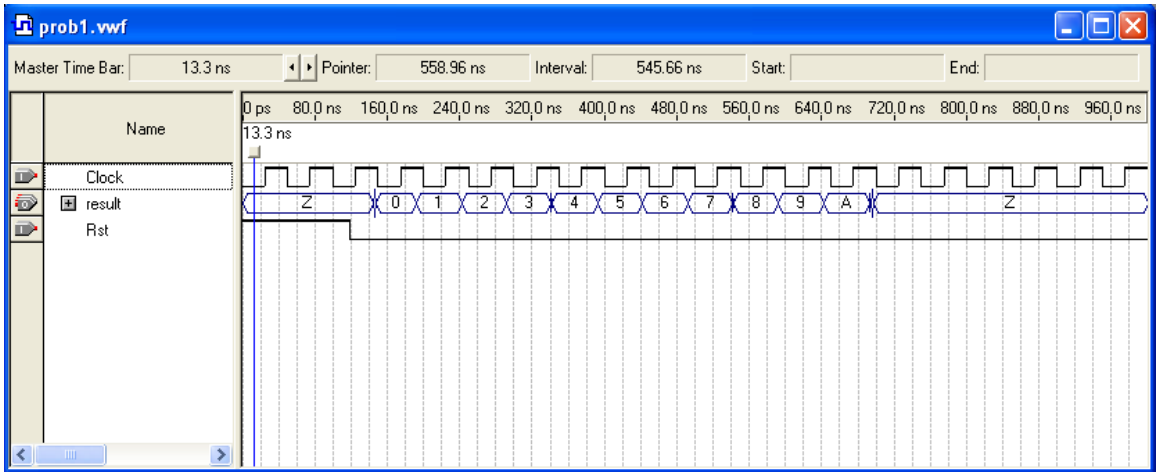
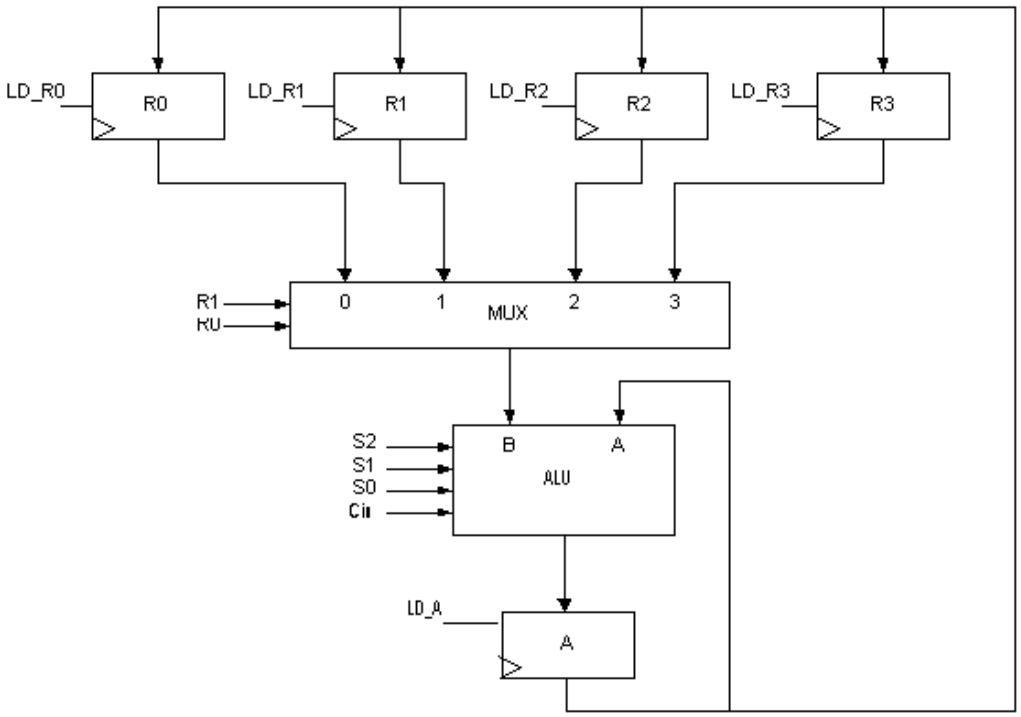


Figure 5. Final Complete Logic Circuit



Problem 4 **.** The following datapath can implement operations on the 8-bit A register and the four registers R0 - R3 (also 8 bits) and store the results back to the R0-R3 registers. The ALU has output F that is defined based on the select inputs as follows:

S2	S1	S0	ALU OUTPUT F
0	0	0	F = A + B + Cin
0	0	1	F = A + B' + Cin
0	1	0	F = A
0	1	1	F = 0
1	0	0	F = A OR B
1	0	1	F = A AND B
1	1	0	F = A'
1	1	1	F = A XOR B



.

(a) Give the values of the control signals for each state, S0-S4, that would be needed to implement the following sequence of operations on this datapath. State S0 is done for you.

Sequence of operations	Control signals
S0: A <---- 0	S2,S1,S0 = 011, LD_A = 1, LD_R0-LD_R3 = 0, R1,R0, Cin = X
S1: A <---- A + R3	
S2: R0 <---- A	
S3: A <---- A XOR R1	
S4: A <---- A - R2	

(b) For initial values: R0=F2, R1=CA, R2=03, R3=88, find the final value of A after this sequence of states is executed.

Winter 2013