EE 209 Homework 4

Name: __Solutions_____

Due: **Score: _____**

Show work to get full credit. Remember, use on only one side of the paper and staple them together. Only use a calculator to CHECK your work, not to DO your work.

1. (32 pts.) Perform the following addition and subtraction problems assuming 2's complement numbers. State whether overflow does or does not occur for each problem. Justify your answer for why overflow does or does not occur. (You can easily check your work by converting to decimal.)

a.)
```
 1010 0110
+1101 1011
-------------
1000 0001
```

n+n=n
$c_{out}=1, c_{in}=1$
No Overflow

b.)
```
 0010 0001
+0111 1001
-------------
 1001 1010
```

p+p=n
$c_{out}=0, c_{in}=1$
Overflow

c.)
```
 1000 0010
-1010 1111
-------------
     ||
 1000 0010
+0101 0000
+         1
-------------
 1101 0011
```

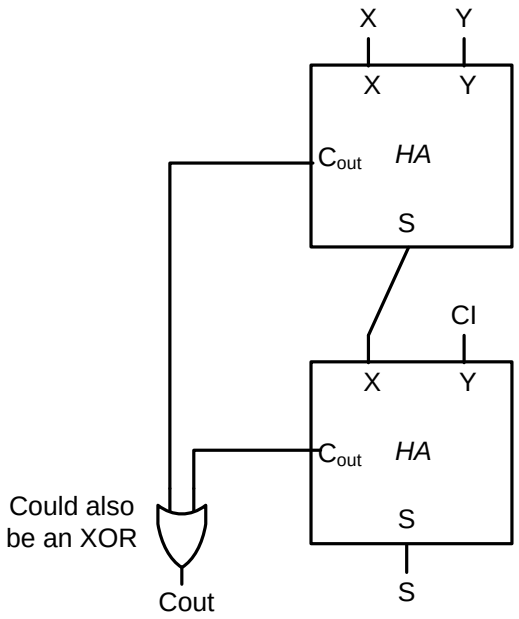
n+p=n
$c_{out}=0, c_{in}=0$
No Overflow

d.)
```
 0101 1001
-1010 0101
-------------
     ||
 0101 1001
+0101 1010
+         1
-------------
 1011 0100
```

n+n=p
$c_{out}=0, c_{in}=1$
Overflow

2. (10 pts.) Build an equivalent full-adder using two half-adders as building blocks along with additional gate(s) if needed.

Answer:



Note: We could use an XOR gate in place of the OR gate since both Cout's will never be 1 at the same time.

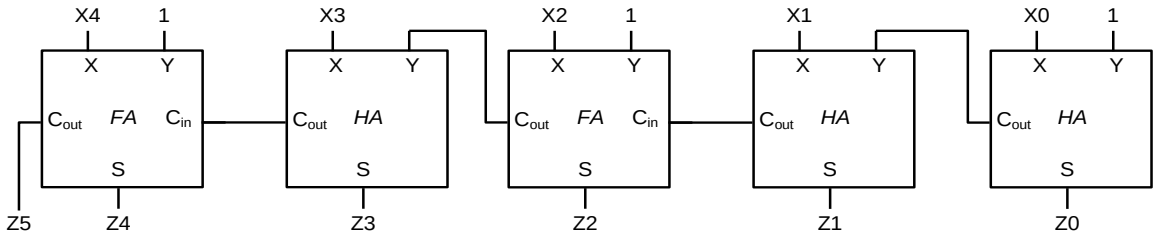
```
module hw4_q2(
    input X,
    input Y,
    input Cin,
    output S,
    output Cout
    );
     wire Stemp, CA, CB;
     // Can add either X+Y, X+Cin or Y+Cin on this adder
     //  with the other input and Stemp on the 2nd HA
     ha a1(.A(X), .B(Y), .S(Stemp), .Cout(CA));
     ha a2(.A(Stemp), .B(Cin), .S(S), .Cout(CB));
     // can also be XOR
     or u0(Cout, CA, CB);

endmodule
```

3. (10 pts.) Using half-adders and full-adders design a circuit that takes in a 5-bit unsigned number, X ($X_4..X_0$) and produces an output equal to X + 21_{10}. Your design should minimize the area required (i.e. use half-adders where possible.)

Answer:

$$Y = X + 21 = X_4\, X_3\, X_2\, X_1\, X_0 + 10101$$

The first bit addition does not have a carry in so we can use a half-adder ($X_0 + 1$). The second bit only adds the possible carry from previous step since the second bit in 21 is 0 (same for fourth bit). The third bit requires a full adder since ($X_2 + 1 +$ CarryFromBit2), same for fifth bit. In total we use 2 full-adders and 3 half-adders.

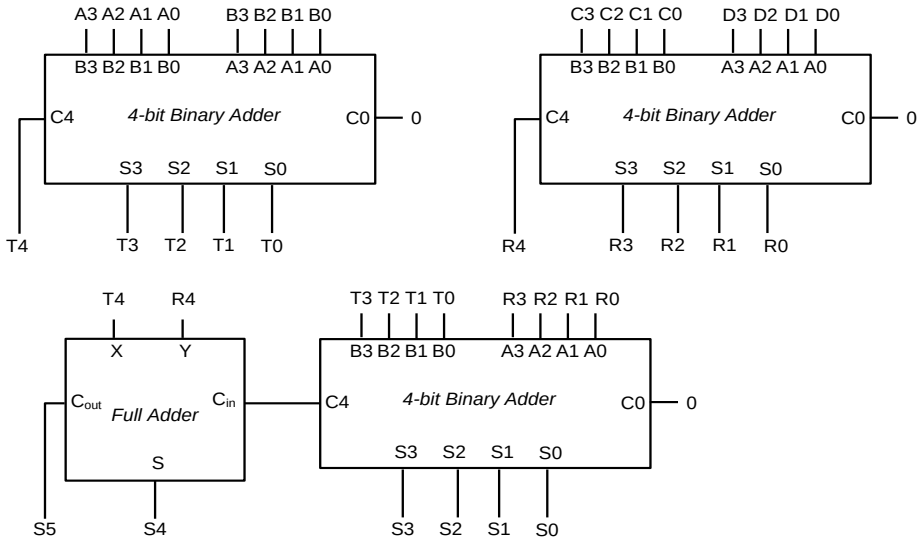



```
module hw4_q3(
    input [4:0] X,
    output [5:0] S
    );
     // Important note:  Constant's can be attaced by writing
     //   1'b0 = Logic 0
     //   1'b1 = Logic 1

     wire [4:1] C;
     // Since 21 dec. = 10101 binary we don't need FA's
     // where there are constant 0 inputs
  ha u0(.A(X[0]), .B(1'b1), .S(S[0]), .Cout(C[1]));
  ha u1(.A(X[1]), .B(C[1]), .S(S[1]), .Cout(C[2]));
  fa u2(.A(X[2]), .B(1'b1), .Cin(C[2]), .S(S[2]), .Cout(C[3]));
  ha u3(.A(X[3]), .B(C[3]), .S(S[3]), .Cout(C[4]));
  fa u4(.A(X[4]), .B(1'b1), .Cin(C[4]), .S(S[4]), .Cout(S[5]));

endmodule
```

4. (20 pts.) Design a circuit that takes in four 4-bit unsigned numbers, A ($A_3..A_0$), B ($B_3..B_0$), C ($C_3..C_0$), and D ($D_3..D_0$) and produces the 6-bit unsigned sum of those numbers. You may use any number of 4-bit adder blocks (74LS283's), single-bit full adders or half adders that we have studied in class. You should organize your adder circuits to perform as many additions in parallel (at the same time) as possible.

ANSWER:

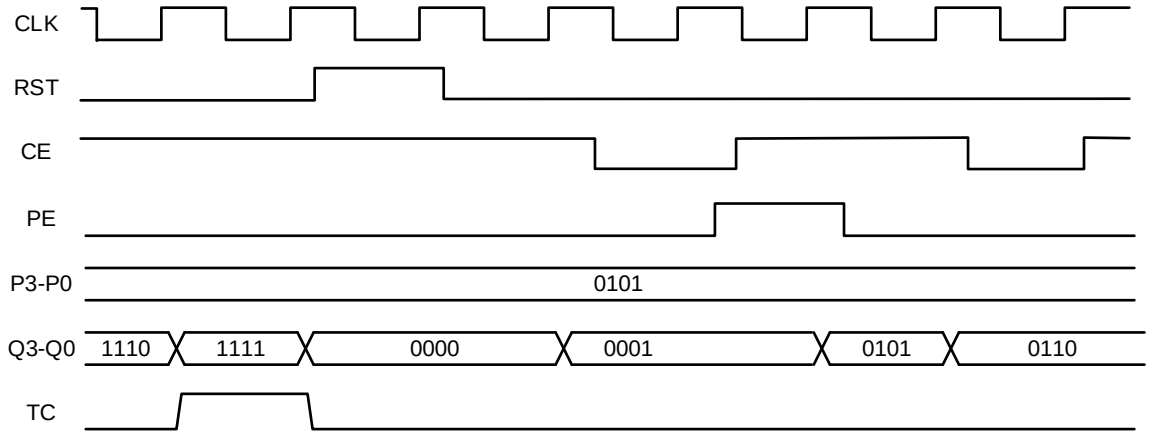
First we will perform two additions in parallel (A+B) and (C+D). Later we add the results of these two additions with another 4-bit adder. One of the carries is attached to this second level adder. The remaining carry from level I and the newly created carry from level II can be added by a half-adder. The result of these carries affects only the higher bits thus it does not affect the lower bits.



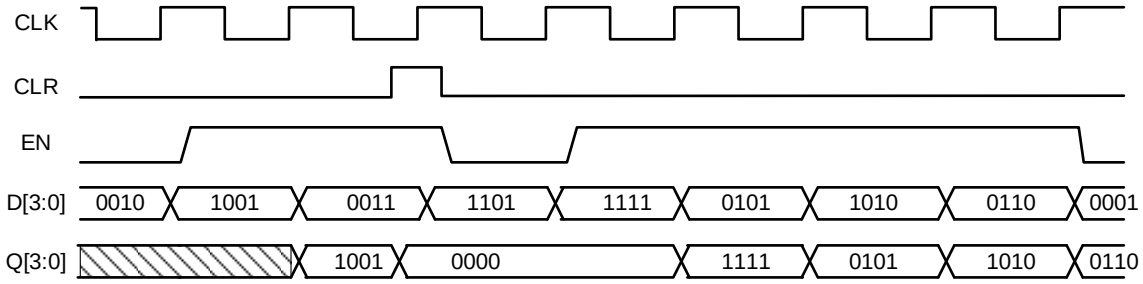
Note: In the Verilog below we name rename T and R to T1 and T2 as well as different names for the carries

```
module hw4_q4(
    input [3:0] A,
    input [3:0] B,
    input [3:0] C,
    input [3:0] D,
    output [5:0] F
    );

       // You may attach bit slices of a signal/output to another module by simply
       //  writing signal[x:y].  For example if a module outputs 3-bits that we
       //  want to drive the lower 3-bits of a 5-bit signal F, we could connect
       //  only F[2:0] to the output of that module.  Then another module or gates
       //  could drive F[5], F[4], F[3]
       wire [3:0] T1;
       wire [3:0] T2;
       wire CA, CB, CC;

       adder4 u0(.A(A), .B(B), .C0(1'b0), .S(T1), .C4(CA));
       adder4 u1(.A(C), .B(D), .C0(1'b0), .S(T2), .C4(CB));
       adder4 u2(.A(T1), .B(T2), .C0(1'b0), .S(F[3:0]), .C4(CC));
       fa u3(.A(CA), .B(CB), .Cin(CC), .S(F[4]), .Cout(F[5]));
endmodule
```

5.


CLK
RST
CE
PE
P3-P0
0101
Q3-Q0
1110
1111
0000
0001
0101
0110
TC

6.


CLK
CLR
EN
D[3:0]
0010
1001
0011
1101
1111
0101
1010
0110
0001
Q[3:0]
1001
0000
1111
0101
1010
0110