Name _____

CS/COE 0447 Example Problems for Exam 2
Spring 2011

1) Show the steps to multiply the 4-bit numbers 3 and 5 with the "fast shift-add multipler". Use the table below. List the multiplicand (M) and product (P) in binary. In the field "step", write "ADD" when the multiplicand is added. Write "SHIFT" to indicate when the product is shifted. In the iteration "Start" write the initial values for the mutiplicand and product. You may not need all steps (rows) in the table.

Iter.	Multiplicand (M)	Product (P)	Step
Start	*0011*	*0000 0101*	*set product=0s:R*
1	*0011*	*0011 0101* *0001 1010*	*lsb=1 => +M* *shift right 1*
2	*0011*	*0001 1010* *0000 1101*	*lsb=0 => +0* *shift right 1*
3	*0011*	*0011 1101* *0001 1110*	*lsb=1 => +M* *shift right 1*
4	*0011*	*0001 1110* *0000 1111*	*lsb=0 => +0* *shift right 1*
5	*NOT NEEDED*		
6	*NOT NEEDED*		
7	*NOT NEEDED*		

Name _____

2) Show the steps to multiply an 6-bit number 17 and 3 with Booth's algorithm. Use the table below. List the multiplicand and product in binary. In the field "step", write "ADD", "SUB", or "NO OP" to indicate which operation is done on each iteration.

Iter.	Multiplicand (M)	Product (P)	Step
Start	*010001* *(negation is 101111)*	*000000 00001**1 0***	*set P, with pad bit*
1	*010001*	*101111 000011 0* *110111 10000**1 1***	*lsbs=10: -M* *shift right arithmetic*
2	*010001*	*110111 100001 1* *111011 11000**0 1***	*lsbs=11: +0* *shift right arithmetic*
3	*010001*	*001100 110000 1* *000110 01100**0 0***	*lsbs=01: +M* *shift right arithmetic*
4	*010001*	*000110 011000 0* *000011 00110**0 0***	*lsbs=00: +0* *shift right arithmetic*
5	*010001*	*000011 001100 0* *000001 100110 0*	*lsbs=00: +0* *shift right arithmetic*
6	*010001*	*000001 100110 0* ***000000 110011** 0*	*lsbs=00: +0* *shift right arithmetic*
7	*NOT NEEDED*	*Final answer is:* ***000000 110011***	*N/A*

3) Explain how the "fast shift-add multiply" improves over the original "slow shift-add multiply". Be sure to indicate what hardware changes make the "fast version" faster than the "slow version".

combines registers, reduces size of ALU & does 3-steps in paralle. last two make it fast.

Name _____

4) Consider restoring division with hardware design #3 (the design with a 32-bit divisor and a 64-bit remainder register that holds the remainder and quotient). Assume the quotient and divisor are 5 bit unsigned numbers. Fill in the table below for 17 / 3. For each step, indicate what shift, subtraction, and addition operations are done in the "Step Notes" column.

Iteration	**Divisor (D)**	**Remainder (R)**	**Notes**
Init	*00011*	*00000 10001*	*Initial values*
1	*00011*	*11101 10001* *00001 00010*	*R=R-D* *R<0: +D, left shift 0 into lsb*
2	*00011*	*11110 00010* *00010 00100*	*R=R-D* *R<0: +D, left shift 0 into lsb*
3	*00011*	*11111 00100* *00100 01000*	*R=R-D* *R<0: +D, left shift 0 into lsb*
4	*00011*	*00001 01000* *00010 10001*	*R=R-D* *R>0: left shift 1 into lsb*
5	*00011*	*11111 10001* *00101 00010*	*R=R-D* *R<0: +D, left shift 0 into lsb*
6	*00011*	*00010 00010* *00100 00101*	*R=R-D* *R>0: left shift 1 into lsb*
Done	*00011*	*00010 00101*	*Right shift the left half of R by 1* *Result = 5, Remainder = 2*

5) Suppose we want to do the computation S = A + B. A and B are positive 2's complement 8-bit binary numbers. Give a boolean expression that indicates whether there was an overflow when these numbers are added. To represent a certain bit *i* in *A*, *B* or *S*, use A_i, B_i or S_i. E.g., bit position 3 in *A* is A_3. Assume the bits are numbered 0 to 7 (right to left).

overflow happens when input values have same sign but output has different one

$Overflow = (A_7 \wedge B_7 \wedge \neg S_7)\ OR\ (\neg A_7 \wedge \neg B_7 \wedge S7)$

6) Give the negation in one's complement binary representation (5 bit numbers) for the decimal numbers:

5d Negation (in one's complement binary) ___*11010*_____

10d Negation (in one's complement binary) ___*10101*_____

-15d Negation (in one's complement binary) ___*01111*_____

Name _____

7) Give the negation in two's complement binary representation (5 bits) for the decimal numbers:

   11d Negation (in two's complement binary) ___*10101*_____

   15d Negation (in two's complement binary) ___*10001*_____

   -13d Negation (in two's complement binary) ___*01101*_____

8) Give Booth's encoding for the 8-bit numbers:

   -19d Booth's encoding _____*00-11 0-11-1*_____

   -19 in two's comp: 1110 1101
   -19 in two's comp with 0 pad: 1110 1101 0
   Booth's encoding: 00-11 0-11-1
   check yourself: $-2^5+2^4-2^2+2^1-2^0 = -32+16-4+2-1=-19$

   27d Booth's encoding _____ *0010 -110-1*_____

   27 in two's comp: 0001 1011
   27 in two's comp with 0 pad: 0001 1011 0
   Booth's encoding: 0010 -110-1
   check yourself: $2^5-2^3+2^2-2^0 = 32 - 8 + 4 -1 = 27$

   62d Booth's encoding _____*0100 00-10*_____

   62 in two's comp: 0011 1110
   62 in two's comp with 0 pad: 0011 1110 0
   Booth's encoding: 0100 00-10
   check yourself: $2^6-2^1 = 62$

9) Give *two reasons* to use Booth's algorithm (encoding) to improve the multiplication hardware.

   1. it can reduce the number of addition operations
   2. it handles signed multiplication

10) Floating-point numbers represent a "richer" set of values than integer numbers. Nevertheless, processors support integer numbers and programs frequently use them. What primary advantage does integer numbers and operations offer over floating point numbers and operations?

   integer operations are significantly faster, programs frequently use discrete values thus, using integer for common operations/values offers a big performance benefit.

Name _____

11) Using 1-bit adders, draw the circuit for a 4-bit ripple-carry addition unit.
See book / class lecture slides.

12) Using 1-bit adders and 1-bit inverters (i.e., the *not* of a bit), draw a circuit for a 4-bit ripple-carry subtract unit.
See book / class lecture slides (drawn on the board during class).

13) Consider the sum of products boolean equation: A'BC + ABC + A'B'C. Give the truth table representation for this boolean equation. (A' is NOT A and + is OR)

This truth table has eight rows with three input values (A, B, C) and one output. Label the rows by counting in binary from 0 to 7. Row 011 (i.e., where A=0, B=1, C=1) has a 1 in the output, row 111 has a 1 in the output, and 001 has a 1 in the output. All other output values are 0.

14) Suppose you want to design a hardware circuit that has two inputs A and B, and one output O. The output O has the value 1 when exactly one input (A/B) is a 1. Give the truth table for this circuit design.

INPUTS		***OUTPUT***
A	***B***	***O***
0	*0*	*0*
0	*1*	*1*
1	*0*	*1*
1	*1*	*0*

(this is exclusive-OR!)

15) For question 15, what is the boolean equation for the truth table?

O = A'B + AB'

16) Here is a truth table. What is the boolean equation for O?

INPUTS			**OUTPUT**
A	**B**	**C**	**O**
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

O=A'B'C + A'BC' + AB'C'

Name _____

17) Let's consider a 4-bit adder. If each 1-bit adder takes 2ns to compute an output (1-bit result and carry-out), how long does it take to compute the full 4-bit result?

*it takes 4 * 2ns = 8ns (Note: We'll have more to say about this soon.)*

18) Consider problem 17 again. This time, let's compute the time for subtraction (A-B). Suppose the 1-bit inverter (used to complement the B input of each 1-bit adder) takes 1ns. How long does it take to compute an answer with this subtraction unit? (*Be careful*: Think about whether the invert operations can be done simultaneously.)

it takes 1ns additional for the first invert (9ns total); the rest are done in parallel.
the first choice is actually faster, so I'd pick this one, assuming "costs" are the same.

19) Consider the logic function: F = A'BC' + A'BC + AB'C + ABC. Draw a K-map for this function, circle the minimum terms (1s), and give the simplified Boolean equation.

		C	
		0	1
AB	00		
	01	1	1
	11		1
	10		1

Simplified form: F = A'B + AC

20) Given the K-map below, what is the simplified Boolean equation?

		CD			
		00	**01**	**11**	**10**
AB	**00**	0	0	0	0
	01	0	0	0	0
	11	0	1	1	0
	10	0	1	1	0

F = AD

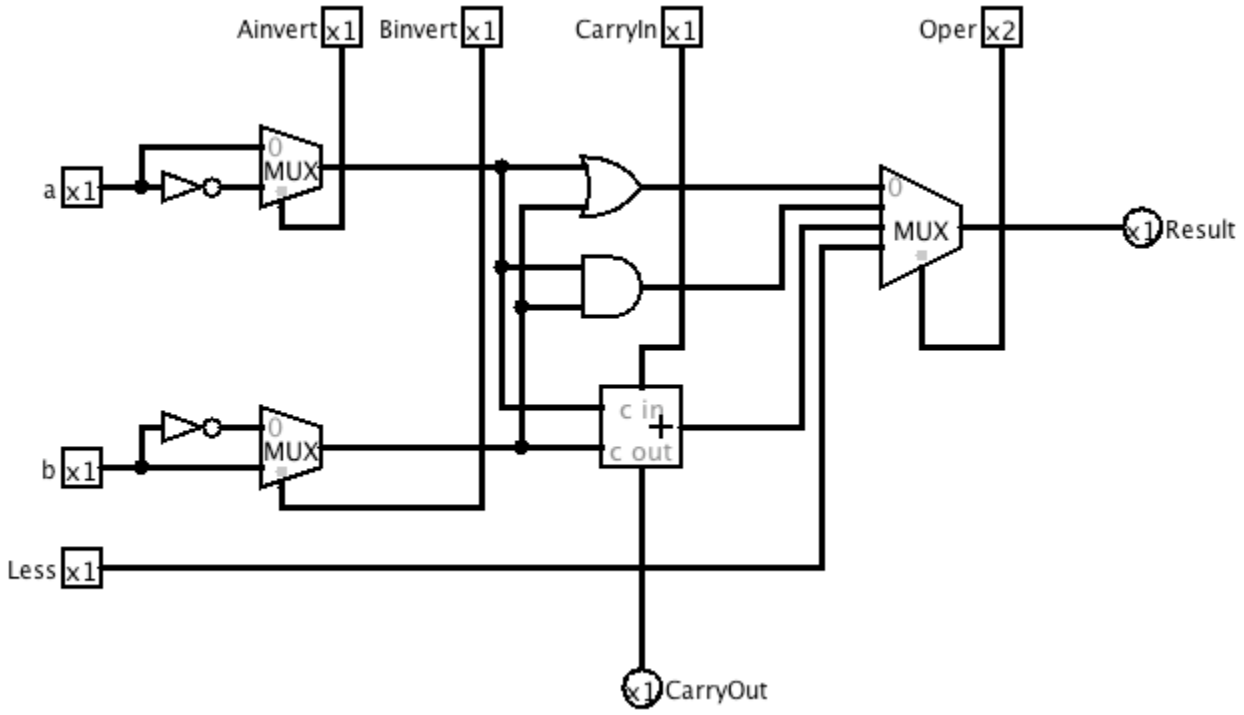
Name _____

21) Given the K-map below, what is the simplified Boolean equation?

		CD			
		00	**01**	**11**	**10**
AB	**00**	1	1	0	0
	01	0	0	1	1
	11	0	0	1	1
	10	0	0	0	0

F = A'B'C' + BC

22) Consider the ALU below. Note this ALU has some minor wiring differences than the one described in class. Answer the questions below for this ALU.



To perform addition, how should Ainvert, Binvert, CarryIn and Oper be set (if the input doesn't matter, use an 'X')?

Name _____

Ainvert=0, Binvert=1, CarryIn=0, Oper=10

To perform an OR operation, how should Ainvert, Binvert, CarryIn and Oper be set (if the input doesn't matter, use an 'X')?

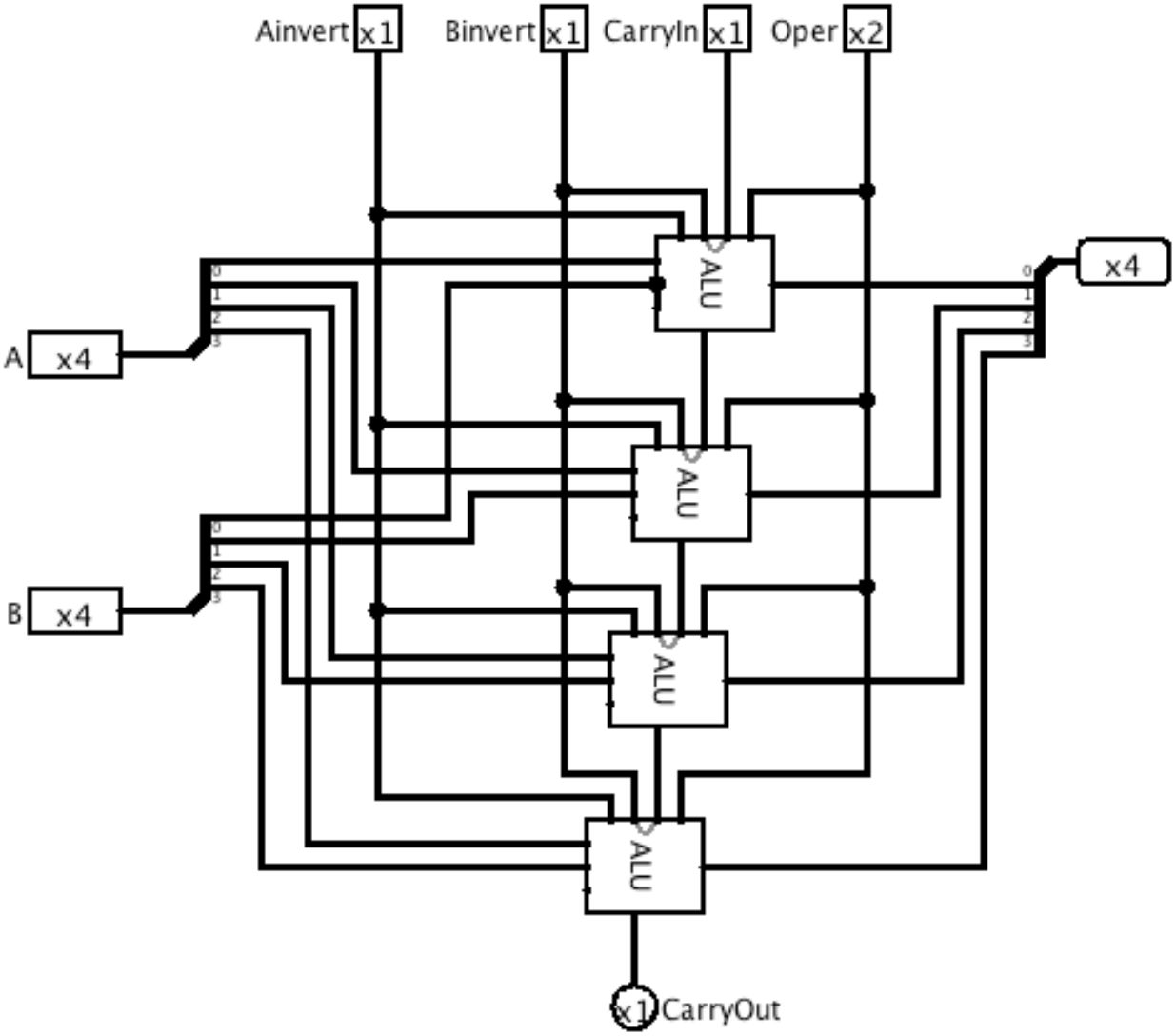
Ainvert=0, Binvert=1, CarryIn=X, Oper=00

To perform a NAND operation, how should Ainvert, Binvert, CarryIn and Oper be set (if the input doesn't matter, use an 'X')?

Ainvert=1, Binvert=0, CarryIn=X, Oper=00

To perform a NOR operation, how should Ainvert, Binvert, CarryIn and Oper be set (if the input doesn't matter, use an 'X')?

Ainvert=1, Binvert=0, CarryIn=X, Oper=01



Name _____

23) Consider the 4-bit ALU made with the 1-bit ALU from the previous questions. Suppose, Ainvert=0, Binvert=0, CarryIn=1, A=0100, B=0011, and Oper=10. What is Result?

Result = 0001 (this is subtract)

24) Now, suppose, Ainvert=0, Binvert=0, CarryIn=1, A=0101, B=0011, and Oper=00. What is the result?

Result = 1101 (Binvert causes B to be inverted to 1100, which is OR'ed with A)