ECE 124 digital circuits and systems
Assignment #4

Q1: Convert the following unsigned numbers to decimal:

(a) $(2314)_5$
$= 2 \times 5^3 + 3 \times 5^2 + 1 \times 5^1 + 4 \times 5^0 = 334$

(b) $(498)_{12}$
$= 4 \times 12^2 + 9 \times 12^1 + 8 \times 12^0 = 680$

(c) $(0111011110)_2$
$= 0 \times 2^9 + 1 \times 2^8 + 1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 478$

(d) $(1011100111)_2$
$= 1 \times 2^9 + 0 \times 2^8 + 1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 743$

(e) $(3742)_8$
$= 3 \times 8^3 + 7 \times 8^2 + 4 \times 8^1 + 2 \times 8^0 + = 2018$

(f) $(A26E)_{16}$
$= A \times 16^3 + 2 \times 16^2 + 6 \times 16^1 + E \times 16^0 + = 10 \times 16^3 + 2 \times 16^2 + 6 \times 16^1 + 14 \times 16^0 + = 41546$

(g) $(F0F0)_{16}$
$= F \times 16^3 + 0 \times 16^2 + F \times 16^1 + 0 \times 16^0 + = 15 \times 16^3 + 0 \times 16^2 + 15 \times 16^1 + 0 \times 16^0 + = 61680$

Q2: Express the following unsigned numbers in decimal:

(a) $(10110.0101)_2 = 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 6 \times 2^0 + 1 \times 2^{-1} + 2 \times 2^{-2} + 2 \times 2^{-3} + 2 \times 2^{-4} = 22.3125$

(b) $(16.12)_{16} = 1 \times 16^1 + 6 \times 16^0 + 1 \times 16^{-1} + 2 \times 16^{-2} = 20.070312$

(c) $(26.24)_7 = 2 \times 7^1 + 6 \times 7^0 + 2 \times 7^{-1} + 4 \times 7^{-2} = 20.36735$

Q3: Determine the base of the unsigned numbers in each of the following cases for the operations to be true:

(a) $14 \div 2 = 5$

Consider the base is the unknown "r". We can write $(24)_r \div (17)_r = (40)_r$ or, alternatively, $(5)_r \times (2)_r = (14)_r$. Now, we can convert the representations to values as follows:

$$(0 \times r + 5) \times (0 \times r + 2) = 10 = 1 \times r + 4.$$

This is true when $r = 6$. Therefore, these numbers must be represented in base 6.

(b) $54 \div 4 = 13$

Consider the base is the unknown "r". We can write $(54)_r \div (4)_r = (13)_r$ or, alternatively, $(13)_r \times (4)_r = (54)_r$. Now, we can convert the representations to values as follows:

$$(1 \times r + 3) \times (0 \times r + 4) = 4 \times r + 12 = 5 \times r + 4.$$

This is true when $r = 8$. Therefore, these numbers must be represented in octal.

(c) $24 + 17 = 40$

Consider the base is the unknown "r". We can write:

$$(24)_r + (17)_r = (40)_r$$

Converting to values we get:

$$(2 \times r + 4) + (1 \times r + 7) = 3 \times r + 11 = (4 \times r + 0)$$

This can be solved to give $r = 11$. We can check:

$$(24)_{11} + (17)_{11} = 2 \times 11 + 4 + 1 \times 11 + 7 = 22 + 4 + 11 + 7\text{‘} = 54 = (54)_{10}.$$

Also,

$$(40)_{11} = 4 \times 11 + 0 = 44 = (44)_{10}.$$

Q4: Convert each of the following unsigned binary numbers to octal and hexidecimal:

(a) $(0111011110)_2$
$= \overbrace{000}^{0} + \overbrace{111}^{7} + \overbrace{011}^{3} + \overbrace{110}^{6} = (0736)_8$
$= \overbrace{0001}^{1} + \overbrace{1101}^{D} + \overbrace{1110}^{E} = (ED1)_{16}$

(b) $(1011100111)_2 = \overbrace{001}^{1} + \overbrace{011}^{3} + \overbrace{100}^{4} + \overbrace{111}^{7} = (1347)_8$
$= \overbrace{0010}^{2} + \overbrace{1110}^{E} + \overbrace{0111}^{7} = (2E7)_{16}$

Q5: Determine the decimal values of the following 2s complement numbers:

(a) $(0111011110)_2$

This number has a leading 0 which, for 2s complement means the number is positive and can be converted to decimal directly.

$$= 0 \times 2^9 + 1 \times 2^8 + 1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + = 478$$

(b) $(1011100111)_2$

This number has a leading 1 which, for 2s complement means the number is negative. To determine its decimal value, we need to 2s complement it, find the positive value and then add a negative sign. The 2s complement is 0100011001. This has a value of 281. Therefore, the number is -281.

(c) $(1111111110)_2$

This number has a leading 1 which, for 2s complement means the number is negative. To determine its decimal value, we need to 2s complement it, find the positive value and then add a negative sign. The 2s complement is 0000000010. This has a value of 2. Therefore, the number is -2.

Q6: Convert the decimal numbers 73, 1906, -95 and -1630 into signed 12-bit binary numbers assuming 2s complement representation.

For each number, convert its absolute value (i.e., the positive number) to a 12-bit binary number. Append leading 0s if required to ensure the number uses 12 bits. Then, if the number is negative take the 2s complement.

					Remainder	
73	÷	2	=	36	1	↑
36	÷	2	=	18	0	↑
18	÷	2	=	9	0	↑
9	÷	2	=	4	1	↑
4	÷	2	=	2	0	↑
2	÷	2	=	1	0	↑
1	÷	2	=	0	1	↑

The number 73 is $(000001001001)_2$.

					Remainder	
1906	÷	2	=	953	0	↑
953	÷	2	=	476	1	↑
476	÷	2	=	238	0	↑
238	÷	2	=	119	0	↑
119	÷	2	=	59	1	↑
59	÷	2	=	29	1	↑
29	÷	2	=	14	1	↑
14	÷	2	=	7	0	↑
7	÷	2	=	3	1	↑
3	÷	2	=	1	1	↑
1	÷	2	=	0	1	↑

The number 1906 is $(011101110010)_2$.

					Remainder	
95	÷	2	=	47	1	↑
47	÷	2	=	23	1	↑
23	÷	2	=	11	1	↑
11	÷	2	=	5	1	↑
5	÷	2	=	2	1	↑
2	÷	2	=	1	0	↑
1	÷	2	=	0	1	↑

The number 95 is $(000001011111)_2$.

Therefore, -95 is the 2s complement of this which is $(111110100001)_2$.

					Remainder	
1603	$\div$	2	=	801	1	$\uparrow$
801	$\div$	2	=	400	1	$\uparrow$
400	$\div$	2	=	200	0	$\uparrow$
200	$\div$	2	=	100	0	$\uparrow$
100	$\div$	2	=	50	0	$\uparrow$
50	$\div$	2	=	25	0	$\uparrow$
25	$\div$	2	=	12	1	$\uparrow$
12	$\div$	2	=	6	0	$\uparrow$
6	$\div$	2	=	3	0	$\uparrow$
3	$\div$	2	=	1	1	$\uparrow$
1	$\div$	2	=	0	1	$\uparrow$

The number 1603 is $(011001000011)_2$.

Therefore, -1603 is the 2s complement of this which is $(100111101101)_2$.

Q7: Perform the following operations involving eight bit 2s complement numbers. Indicate whether or not overflow occurs. Check your answers by converting the numbers to decimal.

00110110	01110101	11011111
+ 01000101	+ 11011110	+ 10111000

00110110	01110101	11010011
- 00101011	+ 11010110	+ 11101100

With 8-bits, we can represent the numbers -128 up to and including 127.

Shown below are each addition, along with the carries generated. Then, there is some comment about overflow.

0	0	0	0	0	1	0	0	
	0	0	1	1	0	1	1	0
+	0	1	0	0	0	1	0	1
	0	1	1	1	1	0	1	1

There is no overflow. Both numbers are positive. The addtion is $54 + 69 = 123$.

1	1	1	1	1	1	0	0	
	0	1	1	1	0	1	0	1
+	1	1	0	1	1	1	1	0
	0	1	0	1	0	0	1	1

There is no overflow. The first number is positive and the second number is negative. The result is positive. There is no overflow since the carry in and carry out of the most-significant bit are the same. We can check. Since the second number is negative, we need to 2s complement to determine its value. This gives 00100010. Therefore, the addition is $117 + (-34) = 83$.

1	1	1	1	1	0	0	0	
	1	1	0	1	1	1	1	1
+	1	0	1	1	1	0	0	0
	1	0	0	1	0	1	1	1

There is no overflow since the carry in and carry out at the most-significant bit are the same. Both numbers are negative and the result is negative. The addition is $(-33) + (-72) = (-105)$.

$$\begin{array}{cccccccc} & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ - & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ \hline \end{array}$$

Since this is subtraction, we need to 2s complement and add.

$$\begin{array}{ccccccccc} 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & \\ & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ + & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ \hline & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}$$

There is no overflow since the carry in and carry out at the most-significant bit are the same. Both numbers are positive and the result is positive. The subtraction is $(54) - (43) = (11)$.

$$\begin{array}{cccccccc} & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ - & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ \hline \end{array}$$

Since this is subtraction, we need to 2s complement and add.

$$\begin{array}{ccccccccc} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & \\ & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ + & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ \hline & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{array}$$

There **is overflow** since the carry in and carry out at the most-significant bit are different. We can also see this from the addition; both numbers are positive (leading 0), but the result is negative (leading 1). The subtraction is $(117) - (-32) = (117) + (32) = (149)$ which exceeds the limit of 127. The actual result obtained is -97.

$$\begin{array}{cccccccc} & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ - & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ \hline \end{array}$$

Since this is subtraction, we need to 2s complement and add.

$$\begin{array}{ccccccccc} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ + & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ \hline & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \end{array}$$

There is no overflow since the carry in and carry out at the most-significant bit are the same. The subtraction is $(-45) - (-20) = (-45) + (20) = -25$.

Q8: Prove the validity of finding a 2s complement of a number by scanning the number from right to left, copying all the 0s and the first 1 and then flipping all the remaining bits.

The 2s complement of a binary number can be found by flipping the bits and adding 1.

Consider the n-bit number $A = a_{n-1}a_{n-2}\cdots a_{k+1}a_k a_{k-1}\cdots a_1 a_0$.

Next, assume that bits $a_{k-1}\cdots a_0$ are all 0 and that $a_k = 1$. If we flip the bits, we will find that $a_{k-1}\cdots a_0$ will all be 1 while $a_k = 0$. Higher bits $a_{n-1}\cdots a_{k+1}$ will also flip.

Next, consider adding 1. We find that $a_{k-1}\cdots a_0 + 1 = 1\cdots 1 + 1 = 10\cdots 0$. Here, all the bits are 0 except for the carry out which will occur in the k-th position. Since the flipped value of a_k is a 0, we will find that in the k-th position we have $1 + 0 = 1$ with *no carry out*.

Hence, we obtain our result — the bits $a_k a_{k-1}\cdots a_0$ which were $10\cdots 0$ remain the same while the higher bits get flipped.

Q9: Prove that the carry out signal, c_k, from the $k-1$ position of an adder can be generated as $c_k = x_k \oplus y_k \oplus s_k$ where x_k and y_k are the inputs and s_k is the sum bit.

We know that $s_k = x_k \oplus y_k \oplus c_k$. We can therefore say that

$$s_k \oplus (x_k \oplus y_k) = x_k \oplus y_k \oplus c_k \oplus (x_k \oplus y_k).$$

Since $\oplus$ is associative, we can remove the parentheses as well as rearrange the terms:

$$s_k \oplus x_k \oplus y_k = x_k \oplus y_k \oplus c_k \oplus x_k \oplus y_k = (x_k \oplus x_k) \oplus (y_k \oplus y_k) \oplus c_k.$$

Since $A \oplus A = 0$, we find that:

$$s_k \oplus x_k \oplus y_k = (0) \oplus (0) \oplus c_k.$$

Since $0 \oplus B = B$, we find that:

$$s_k \oplus x_k \oplus y_k = \oplus c_k$$

which is what we are trying to prove.

Q10: Determine the number of gates required to implement an n-bit carry-lookahead adder. Use AND, OR and XOR gates with any number of inputs.

We need to generate the sums and the carry outs. Recall that the carry out, c_{i+1} of the i-th bit is

$$c_{i+1} = x_i y_i + c_i(x_i + y_i) = x_i y_i + c_i(x_i \oplus y_i)$$

So we can define $g_i = x_i y_i$ and $p_i = x_i + y_i$ (or $p_i = x_i \oplus y_i$ — it doesn't matter for the question asked). For full carry lookahead, we find that

$$c_{i+1} = g_i + p_i c_i = g_i + p_i g_{i-1} + \cdots + p_i p_{i-1} \cdots p_1 p_0 c_0.$$

We can write down a couple of the carry outs to get an idea:

$$\begin{aligned} c_1 &= g_0 + p_0 c_0 \\ c_2 &= g_1 + p_1 g_0 + p_1 p_0 c_0 \\ c_3 &= g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0 \\ &\cdots \end{aligned}$$

Therefore, to generate all the carry outs for n-bits requires computing $c_n, c_{n-1}, \cdots c_1$. Each g_i requires 1 AND gate for a total of **n gates for the $\mathbf{g_i}$**. Each p_i requires 1 OR gate (or 1 XOR gate) for a total of **n gates for the $\mathbf{p_i}$**. The carry out c_1 requires 2 gates, c_2 requires 3 gates, c_3 requires 4 gates, $\cdots$, and c_n will therefore require $n+1$ gates. We can find a formula which says that

$$\sum_{i=1}^{n} i = 1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}.$$

Here, we have the summation $2+3+4+\cdots+(n+1) = (1+2+3+\cdots+n)+n = \frac{n(n+1)}{2} + n$. Therefore, we need an additional $\mathbf{\frac{n(n+1)}{2} + n}$ **gates to generate all the carry out signals**.

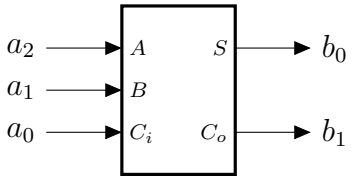
Finally, we need to generate the sum outputs. The sum output can be written as $s_i = x_i \oplus y_i \oplus c_i$. Therefore, we need an additional **n gates to generate the $\mathbf{s_i}$**.

Finally, in total, we need the following total number of gates:

$$n + n + \frac{n(n+1)}{2} + n + n = 4n + \frac{n(n+1)}{2} = \frac{n(n+9)}{2}.$$

Q11: Design a simple circuit that determines how many bits are equal to 1 in a 3-bit unsigned number.

Assume the 3 bits of the unsigned number are a_2, a_1 and a_0. Let the number of bits which are equal to 1 be represented by a binary number $B = b_1 b_0$ (two bits is sufficient). The solution is to use a single full adder; the number of bits which are equal to 1 is available at the output of the full adder.

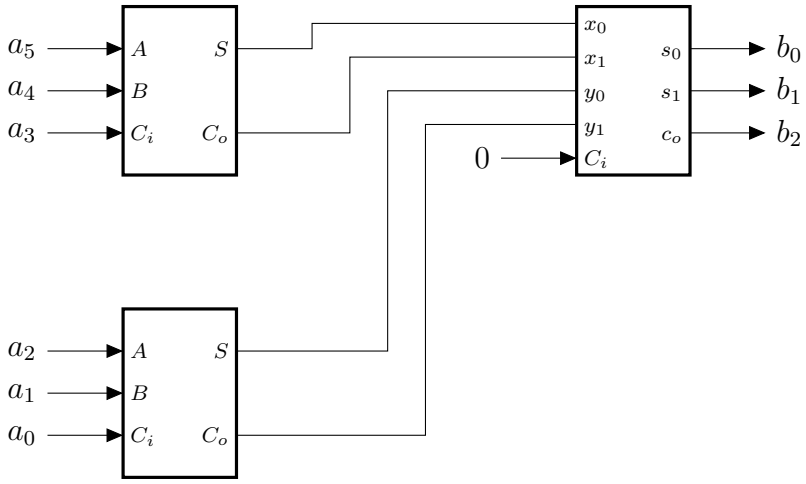


Q12: Repeat question 11 assuming a 6-bit unsigned number.

Assume the 6 bits of the unsigned number are a_5, a_4, a_3, a_2, a_1 and a_0. Let the number of bits which are equal to 1 be represented by a binary number $B = b_2b_1b_0$ (three bits is sufficient).

We can divide the inputs into two groups of three, determine the number of bits which are equal to 1 in each group and then add those numbers together.

So, we can accomplish our task with two full adders and then one, 2 bit full adder. Be careful — where each signal is connected is important!



Q13: Design a simple circuit that can add 3, 4-bit unsigned numbers. You may use only 4-bit adders and any other required logic gates.

Let the numbers be:

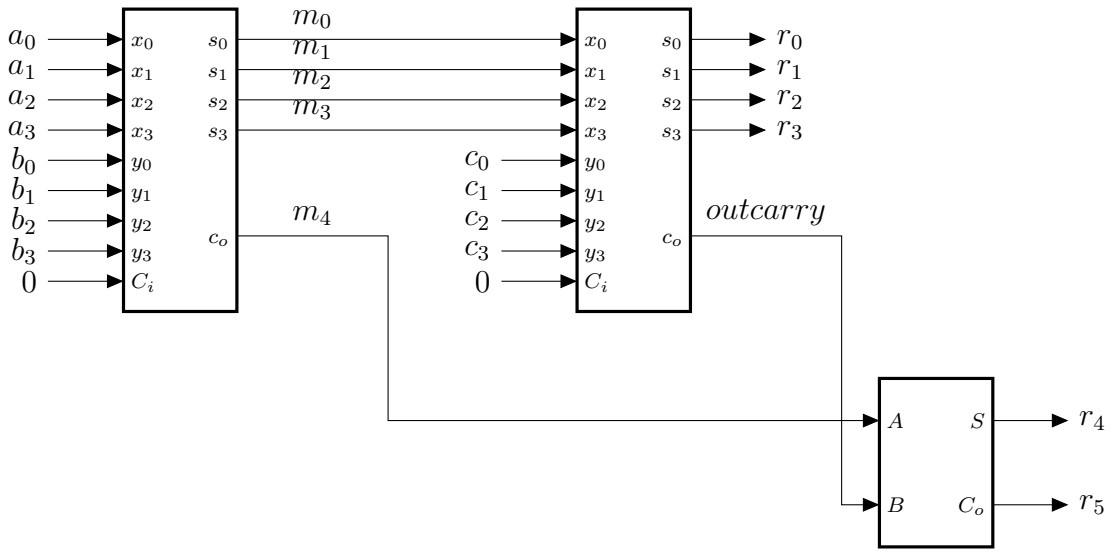
$$\begin{array}{lcllll} A & = & a_3 & a_2 & a_1 & a_0 \\ B & = & b_3 & b_2 & b_1 & b_0 \\ C & = & c_3 & c_2 & c_1 & c_0 \end{array}$$

The summation of 2, 4-bit numbers will be a 5-bit result. Therefore, the summation of 3, 4-bit numbers will be a 6-bit result. We can write down the following:

$$\begin{array}{llllllll} & & & a_3 & a_2 & a_1 & a_0 \\ + & & & b_3 & b_2 & b_1 & b_0 \\ \hline & & m_4 & m_3 & m_2 & m_1 & m_0 \\ + & & 0 & c_3 & c_2 & c_1 & c_0 \\ \hline & r_5 & r_4 & r_3 & r_2 & r_1 & r_0 \end{array}$$

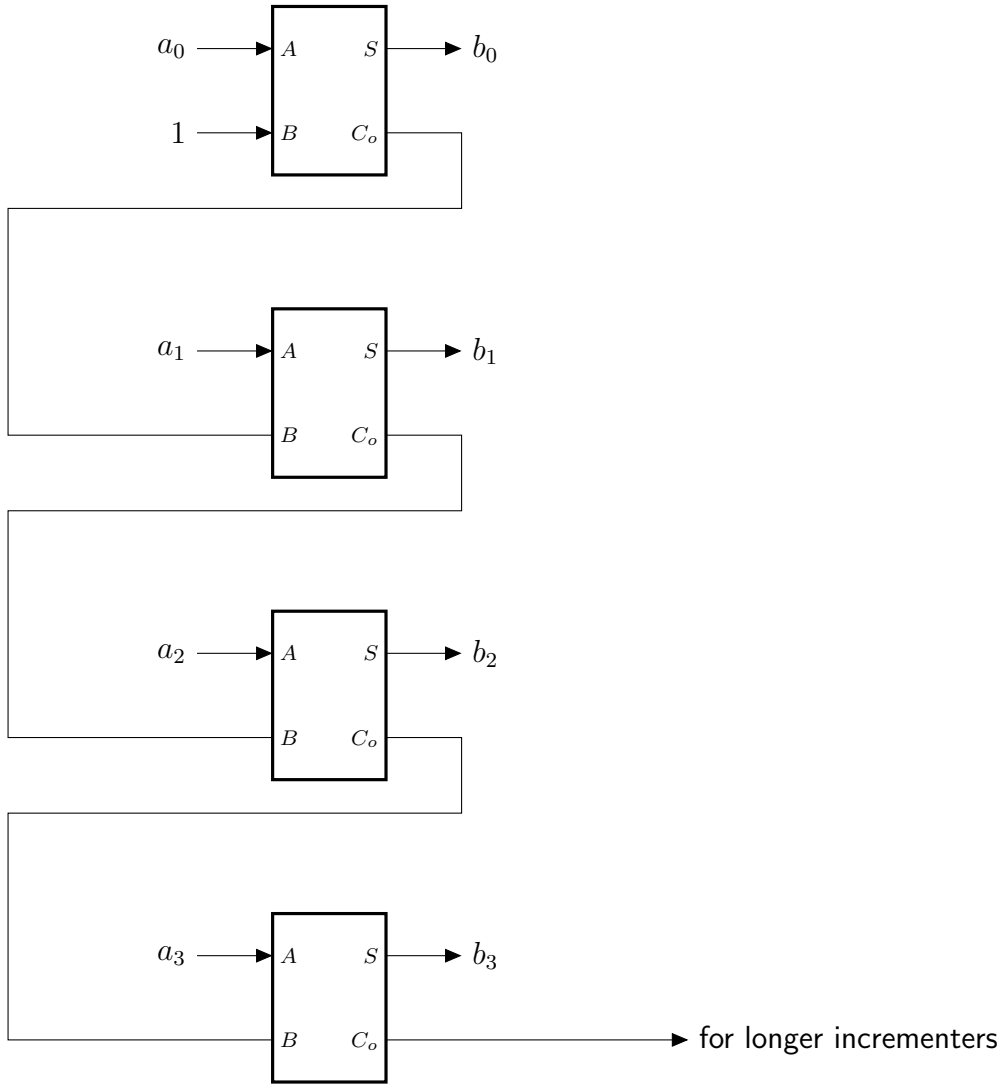
We can use one, 4-bit adder to add $a_3a_2a_1a_0$ and $b_3b_2b_1b_0$ to get $m_3m_2m_1m_0$ and the carry out m_4. Then, we can use another 4-bit adder to add $m_3m_2m_1m_0$ and $c_3c_2c_1c_0$. This addition could generate a carry out, call it *outcarry*. The result of this summation will produce $r_3r_2r_1r_0$. Finally, we need to produce r_4 and r_5 which will be the summation of m_4 and *outcarry*. This can be accomplished with a half-adder.

Finally, we end up with the following circuit:



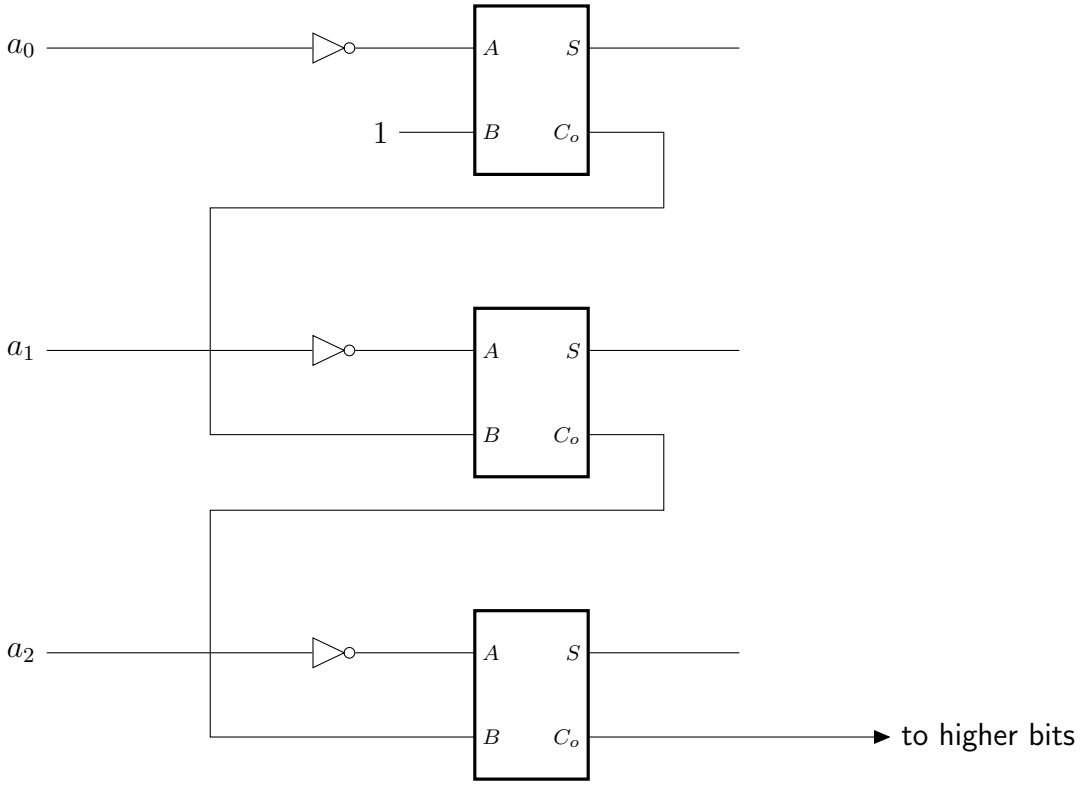
Q14: An incrementer circuit is a circuit that adds one to a binary number. Design a 4-bit incrementer circuit using only half-adder circuits.

This is straightforward. Let the 4-bit number be $a_3a_2a_1a_0$. We simply link the half-adders together and connect a constant 1 to the second input on the first half-adder. Let the incremented result be $b_3b_2b_1b_0$. The result is available at the outputs of the half-adders.



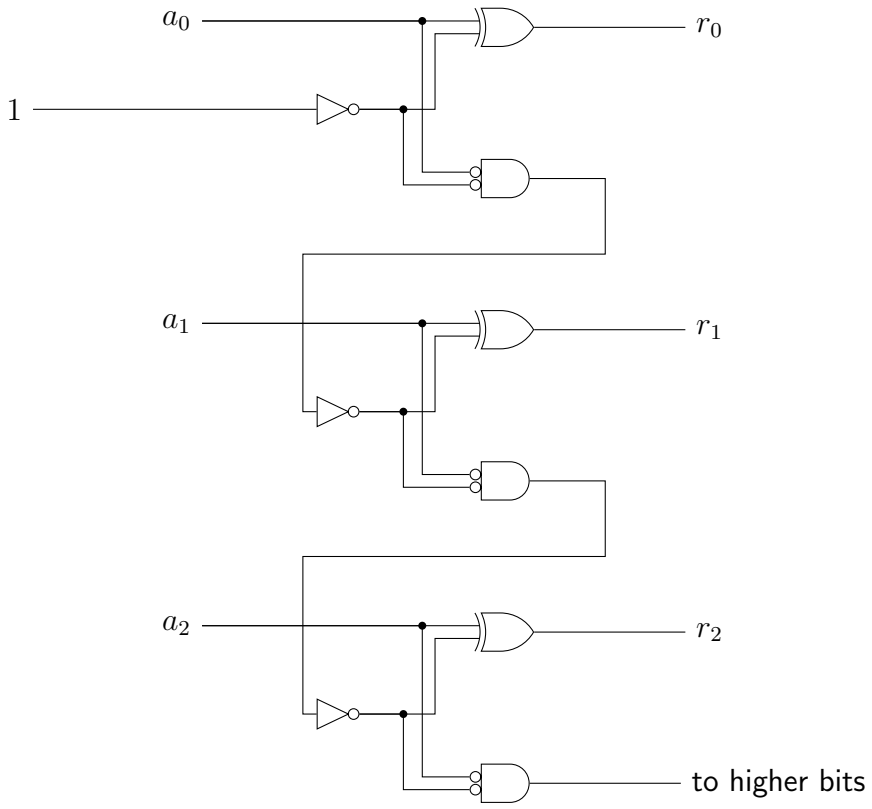
Q15: Design a simple circuit that outputs the 2s complement if an n-input binary number. Show that the circuit can be constructed with only XOR and OR gates.

We could build the circuit much like and incrementer circuit if we realize that the 2s complement requires flipping the bits and adding 1. This would yield the following:

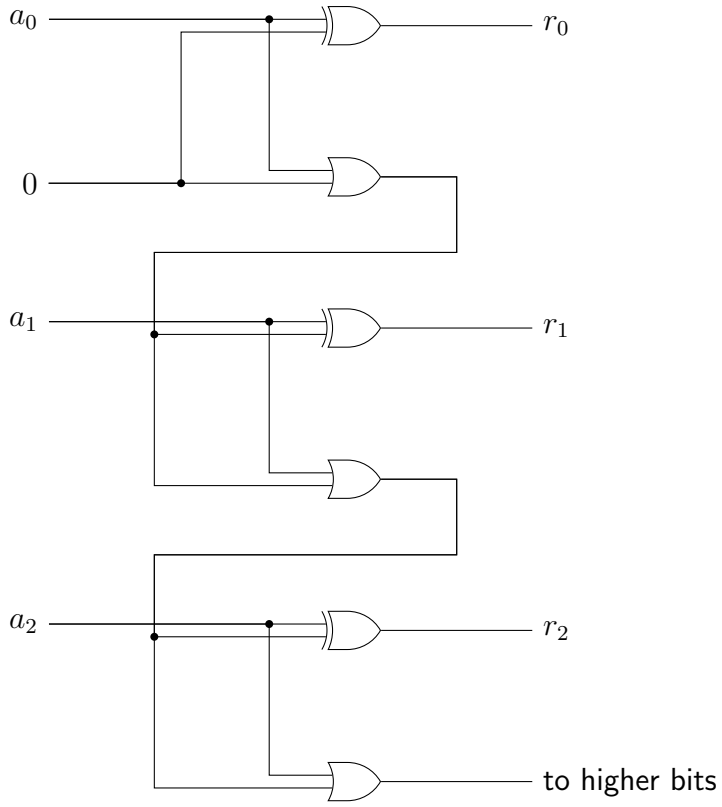


We can do better than this; we do not want any explicit inverters to invert a_i.

We can first observe that $s_i = a_i^{'} \oplus b_i = a_i^{'}b_i^{'} + a_i b_i = a_i \oplus b_i^{'}$. Consequently, we can redraw the circuit as follows (with explict gates). Note that some inversions have been inserted to ensure that the logic is still working correctly.



We can see that the AND gate along the carry chain has inverted inputs — it can be converted to a NOR gate. Then, we can see that the inverter at the output of the NOR gate will cancel the inverter an the input of the next higher bit. Finally, we can replace the initial inverter by connecting 0 rather than a 1. Our final circuit looks like:



This circuit requires only XOR and OR gates and expands to any number of bits.

Q16: Consider the 3-input logic function Y shown in Figure 2 build from CMOS transistors. Complete the circuit and then determine the logic function for Y.

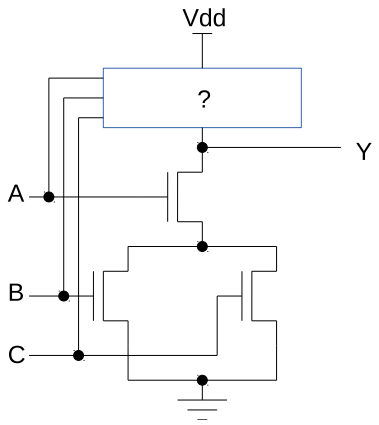


Figure 1: Figure for problem 16.

The missing circuitry is the complement of the shown circuitry. That is, for each NMOS we need a PMOS. For NMOS in series (parallel), the PMOS must be in parallel (series).

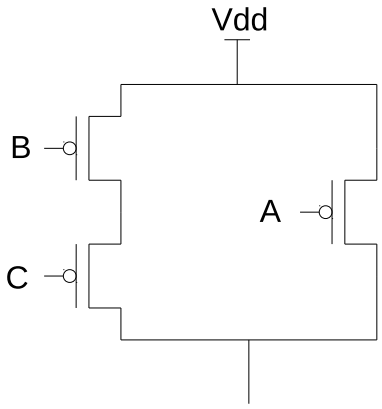


Figure 2: Solution for problem 16.

To determine the logic function, we can simply figure out when the output is pulled to 0 (or, alternatively, when the output is pulled to 1). Due to the complementary nature of the circuit, we know that whenever the output Y is pulled to ground (V_{DD}), it is disconnected from V_{DD} (ground).

We can see that Y is pulled to ground when $A(B + C)$. Equivalently, Y is pulled to V_{DD} when $A^{'} + B^{'}C^{'}$. Hence, $Y = \overline{A(B + C)}$.