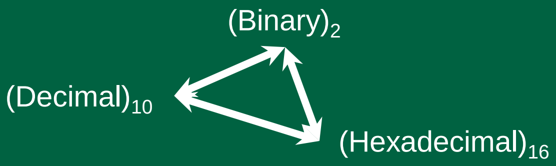
THE UNIVERSITY OF ALABAMA AT BIRMINGHAM

CS330

Binary Conversions, Boolean Algebra

Spring 2022

Lab 4



Overview of Binary

- Only two digits, 0 and 1
- Each binary digit represents exactly a power of two
- 1010 base 2 = (2^3) + (2^1) = 10 base 10

2^3	2^2	2^1	2^0
8	4	2	1
1	0	1	0

8 + 0 + 2 + 0 = 10

Binary addition works exactly like you think:

01 + 01 = 10

0011 + 0010 + 0101

0 and 0 sum to 0

0 and 1 sum to 1

1 and 1 sum to 0 with a carry

$$\begin{array}{r} 01 \\ \underline{+\ 01} \end{array} \qquad \begin{array}{r} 01 \\ 01 \\ \underline{+\ 01} \end{array}$$

Overview of Hexadecimal

16 digits: 0 through 9, then A through F corresponding to 0 through 15

Hexadecimal is commonly used to represent binary, compressed like so:

$$(4C)_{16} = (?)_2$$

$$(4)_{16} = 0100 \quad (C)_{16} = 1100$$

$$(4C)_{16} = (0100\ 1100)_2$$

This allows us to convert freely between hexadecimal and binary, and back.

Decimal	Hex	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Power Notation

All numbers may be conveniently broken down by their digits, multiplied by the base to the power of the position of each digit. This gives you a convenient way to retrieve the base 10 representation of any number.

$(001100)_2 = (1 * 2^3) + (1 * 2^2) = 12$

2^5	2^4	2^3	2^2	2^1	2^0
32	16	8	4	2	1
0	0	1	1	0	0

0 + 0 + 8 + 4 + 0 + 0 = 12

$(837)_{10} = (8 * 10^2) + (3 * 10^1) + (7 * 10^0)$

$(2C)_{16} = (2 * 16^1) + (C * 16^0)$

$= (2 * 16^1) + (12 * 16^0)$

$= 44$

		16^2	16^1	16^0
		256	16	1
			2	C

(2 * 16) +(12 * 1) = 44

So, we can use this tool to convert any other base to base 10.

Decimal to Binary

The common algorithm for converting from decimal to binary works like this:

1. Divide the input by 2, noting the remainder (which will be a 0 or 1)
2. Repeat until the input is 1 or 0 (no division is possible anymore)
3. Reverse all noted remainders in order (including zeroes).

For example: $(10)_{10} = (?)_2$

10 / 2 = 5 r **0**   0101 -> **1010**

5 / 2 = 2 r **1**

2 / 2 = 1 r **0**

1/ 2 = 0 r **1**

If you change the base, this algorithm works for any conversion!

Decimal to Binary (2nd option)

1. Start with a table with one column more than you need (larger than the number)
2. Pick the largest power of two that will still fit in the number (place a one in that column)
3. Subtract for original number
4. Repeat Steps 2,3 until number is zero

For example: $(10)_{10} = (?)$

$$\begin{array}{r} 10 \\ \underline{-\ 8} \\ 2 \\ \underline{-2} \\ 0 \end{array}$$

2^4	2^3	2^2	2^1	2^0
16	8	4	2	1
-	1	-	-	-

2^4	2^3	2^2	2^1	2^0
16	8	4	2	1
-	1	0	1	-

2^5	2^4	2^3	2^2	2^1	2^0
32	16	8	4	2	1
0	0	1	0	1	0

NOTE: On HW, Need to show subtraction

Exercise: Conversions

1) $(1101)_2 = (\qquad)_{10}$
2) $(168)_{10} = (\qquad)_2$
3) $(3B)_{16} = (\qquad)_{10}$
4) $(11001011)_2 = (\qquad)_{10}$
5) $(00110101)_2 = (\qquad)_{10}$
6) $(213)_{10} = (\qquad)_2$
7) $(67)_{10} = (\qquad)_2$
8) $(233)_{10} = (\qquad)_{16}$
9) $(1100\ 1100)_2 = (\qquad)_{16}$

(233)10 = (   )16

(75)10 = (   )16

(11001100)2= (   )16

(11110001)2= (   )16

(5D)16= (   )2

(FA)16= (   )2

(97)16= (   )10

(40)16= (   )10

Exercise: Addition and Subtraction
(convert to binary, then add, subtract)

a) 14 + 7 (001110 + 000111)

b) 8 - 3 (01000 + 11101)
 [add the inverse (flip all bits, add one), make sure # of bits align]
 3 = 00011, to get -3 flip bits (11100), and add 1 (11101)

c) 9 + 21

d) 15 - 6

Boolean Algebra

Boolean Algebra: CS250 Review

Basic Postulates

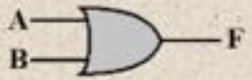
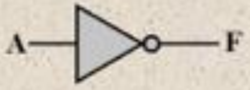
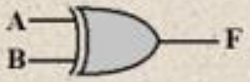
$A \cdot B = B \cdot A$	$A + B = B + A$	Commutative Laws
$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$	$A + (B \cdot C) = (A + B) \cdot (A + C)$	Distributive Laws
$1 \cdot A = A$	$0 + A = A$	Identity Elements
$A \cdot \overline{A} = 0$	$A + \overline{A} = 1$	Inverse Elements

Other Identities

$0 \cdot A = 0$	$1 + A = 1$	
$A \cdot A = A$	$A + A = A$	
$A \cdot (B \cdot C) = (A \cdot B) \cdot C$	$A + (B + C) = (A + B) + C$	Associative Laws
$\overline{A \bullet B} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A} \bullet \overline{B}$	DeMorgan's Theorem

Boolean Algebra: Icons

Everything here follows the same rules as the previous slide, except that these are graphical representations of those various gates.

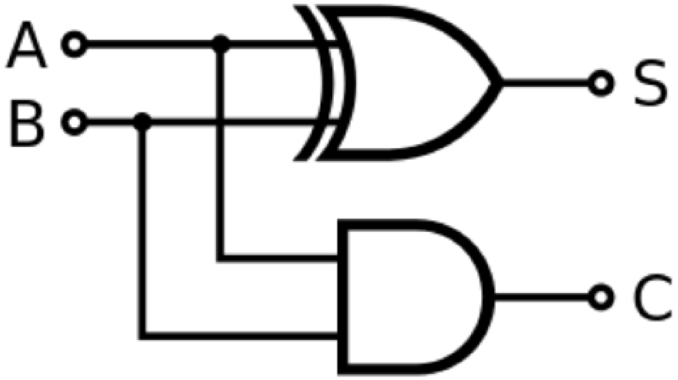
Name	Graphical Symbol	Algebraic Function	Truth Table
AND		$F = A \bullet B$ or $F = AB$	A B | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$F = A + B$	A B | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
NOT		$F = \bar{A}$ or $F = A'$	A | F 0 | 1 1 | 0
NAND		$F = \overline{AB}$	A B | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = \overline{A + B}$	A B | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
XOR		$F = A \oplus B$	A B | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0

Example

- Boolean Algebra Expressions
    - $S = A \oplus B$
    - $C = A \cdot B$
- Truth Table

A	B	S ($A \oplus B$)	C ($A \cdot B$)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Half Adder



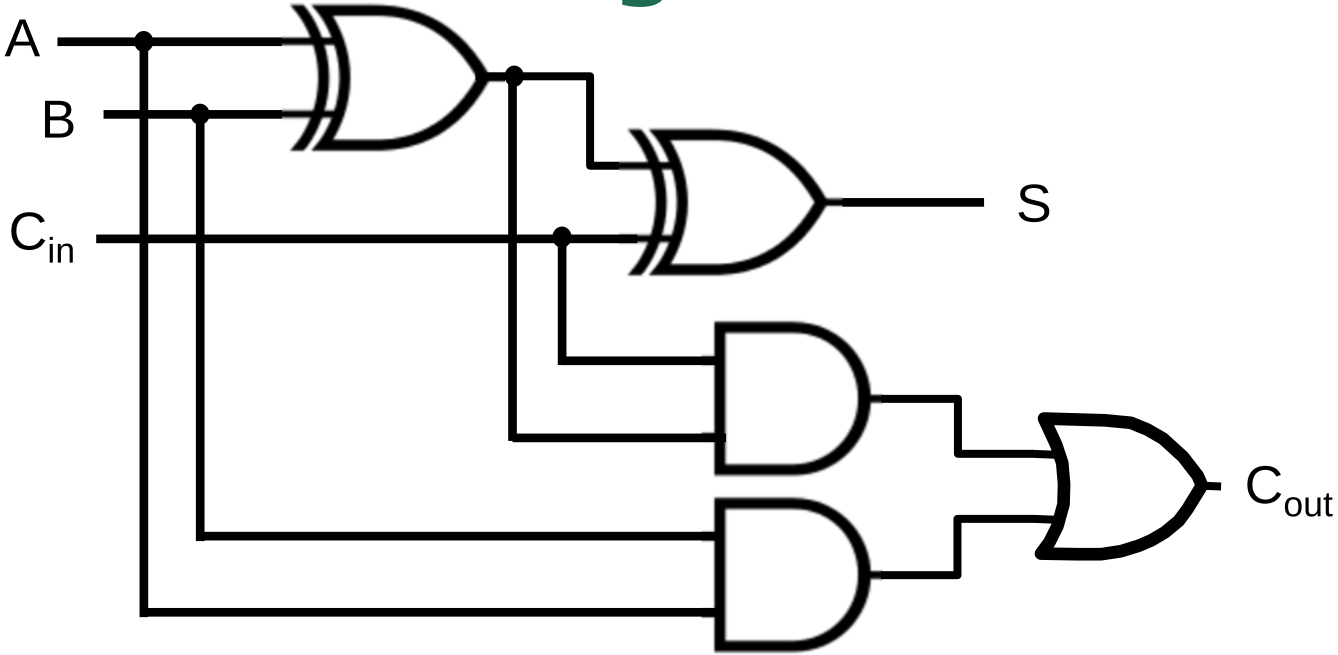
SUM

CARRY

A (input)	B (input)	C (output)	S (output)
0	0	0	0
1	0	0	1
0	1	0	1
1	1	1	0

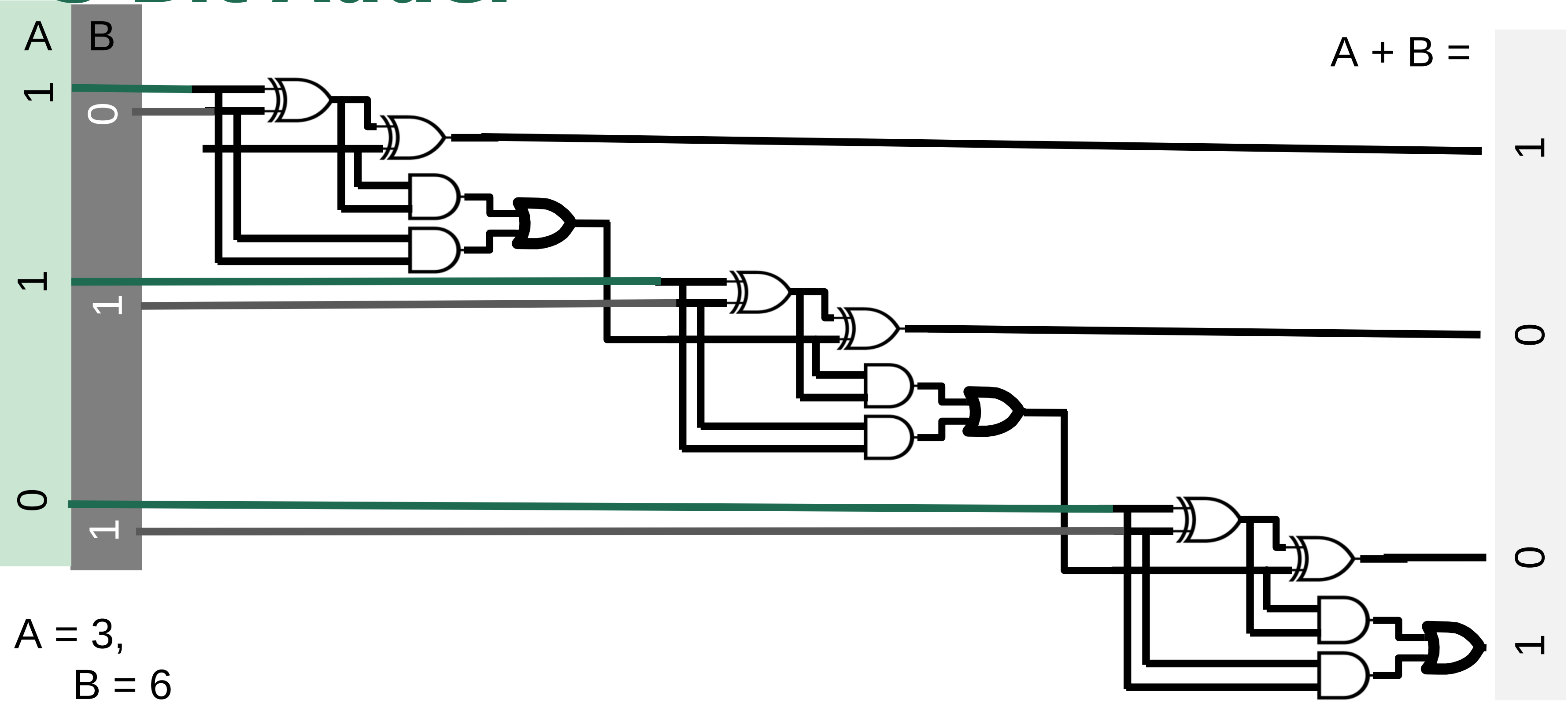
https://www.techopedia.com/definition/7509/half-adder

Exercise: Boolean Algebra



- Write the Boolean Algebra Expression for S and C_{out}
- Create a Truth Table

3 Bit Adder



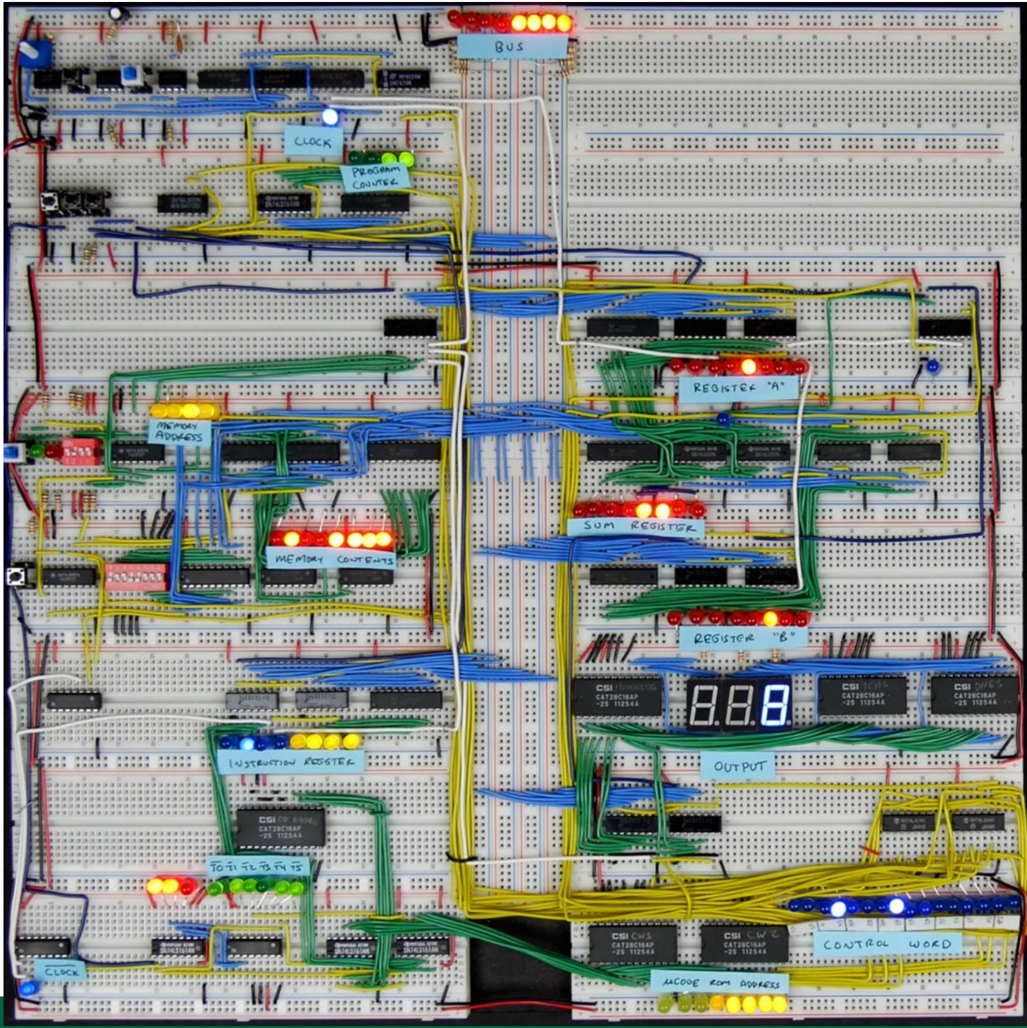
A = 3,
B = 6

- If this interests you, check out Ben Eater's YouTube channel, esp the series where he builds an 8-Bit Computer on a breadboard:

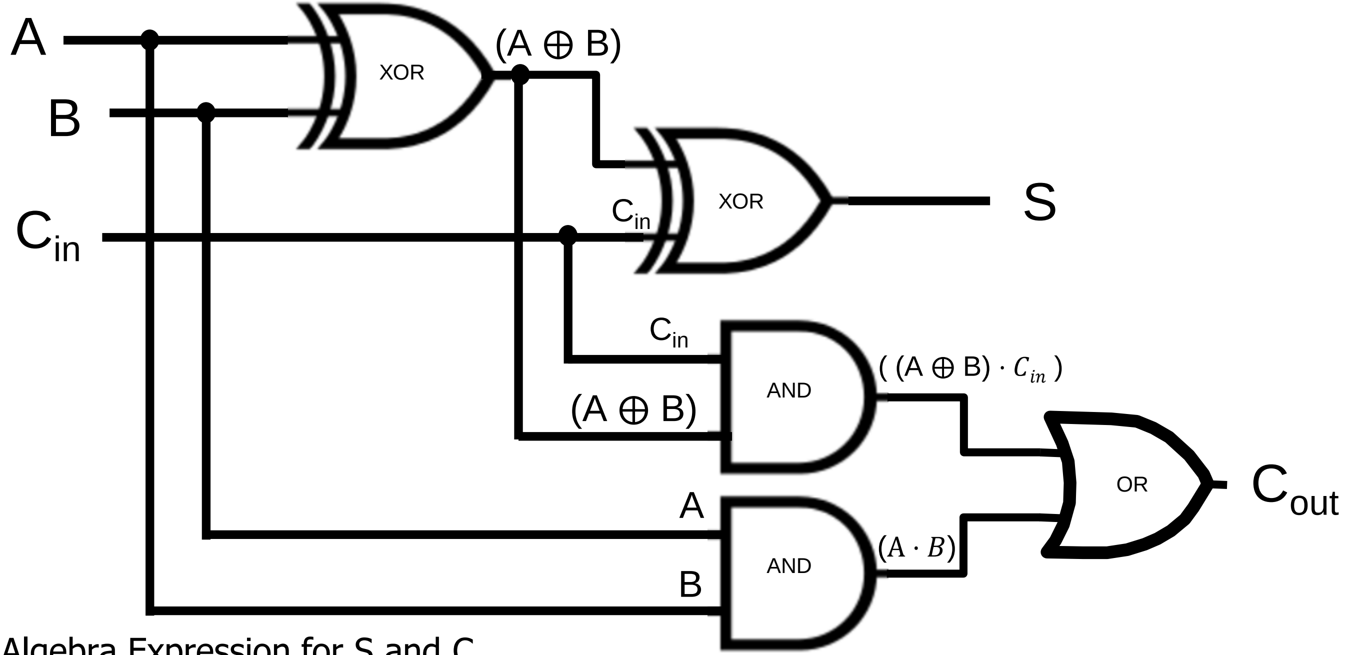
https://youtu.be/HyznrdDSSGM

4-Bit Adder:

https://youtu.be/wvJc9CZcvBc



ANSWER Exercise: Boolean Algebra



- Write the Boolean Algebra Expression for S and C_{out}
  - $S = (A \oplus B) \oplus C_{in}$
  - $C_{out} = (A \cdot B) + ((A \oplus B) \cdot C_{in})$
- Create a Truth Table

- Truth Table

A	B	C_{in}	$(A \oplus B)$	$S = (A \oplus B) \oplus C_{in}$	$(A \oplus B) \cdot C_{in}$	$A \cdot B$	$C_{out} = (A \cdot B) + ((A \oplus B) \cdot C_{in})$
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	1	1	0	0	0
0	1	1	1	0	1	0	0
1	0	0	1	1	0	0	0
1	0	1	1	0	1	0	1
1	1	0	0	0	0	1	1
1	1	1	0	1	0	1	1