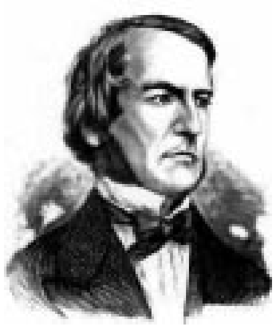
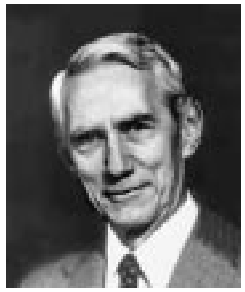
Lecture A3: Boolean Circuits



George Boole
(1815 – 1864)



Claude Shannon
(1916 – present)

Digital Circuits

What is a digital system?

Why digital systems?

Digital circuits and you.

- Computer microprocessors.
- Antilock brakes.
- VCR.
- Cell phone.

Digital Circuits

Logic gates.

- AND, OR, NOT.

Build combinational circuits from gates.

- No memory.
- Adder, multiplexer, decoder.

Lecture A4: build sequential circuits from gates.

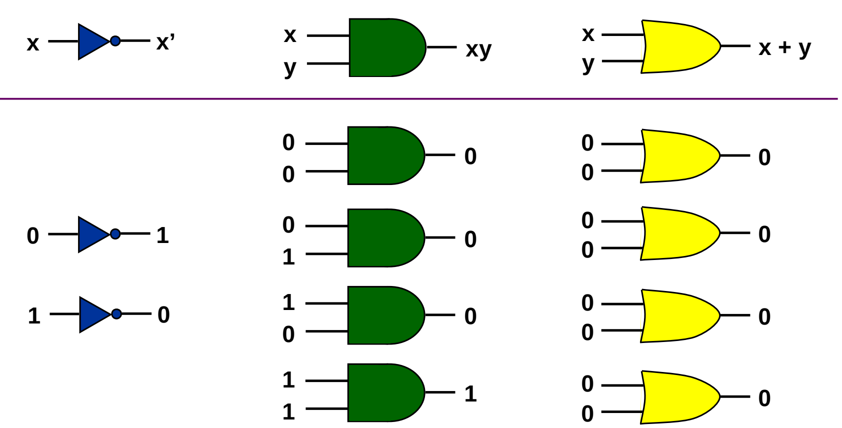
- Memory.
- Flip-flop, register, counter.

Lecture A5: build general-purpose machine from circuits.

Logical Gates

Logical gates.

- A smallest useful circuit.

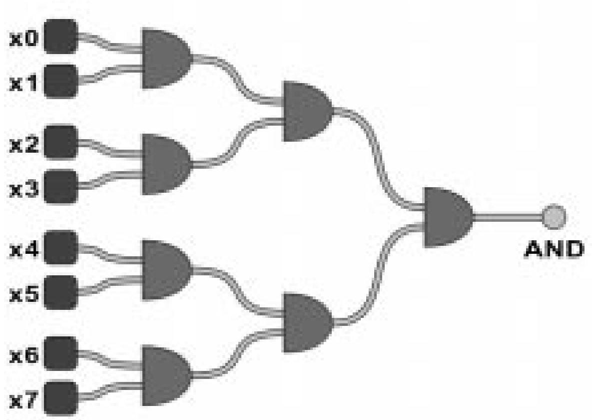


NOT AND OR

Multiway AND Gates

AND(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7).

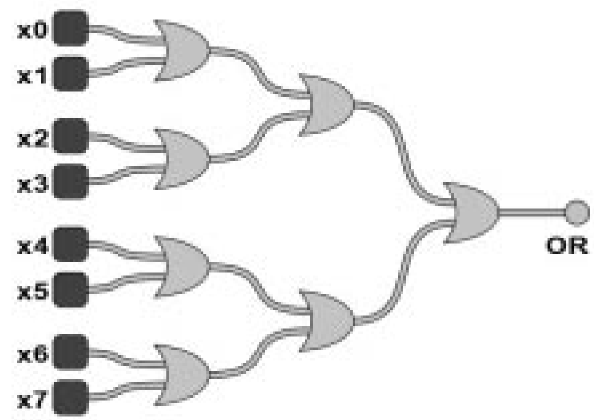
- 1 if all inputs are 1.
- 0 otherwise.



Multiway OR Gates

OR(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7).

- 1 if at least one input is 1.
- 0 otherwise.



Boolean Algebra

History.

- Developed by Boole to solve mathematical logic problems (1847).
- Shannon first applied to digital circuits (1939).

Basics.

- Boolean variable: value is 0 or 1.
- Boolean function: function whose inputs and outputs are Boolean variable.

Relationship to circuits.

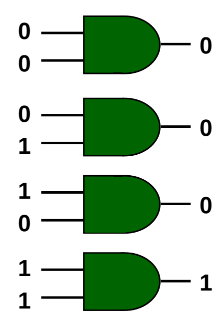
- Boolean variables: signals.
- Boolean functions: circuits.

Truth Table

Truth table.

- Systematic method to describe Boolean function.
- One row for each possible input combination.
- N inputs $\Rightarrow$ 2^N rows.

AND Truth Table		
x	y	AND
0	0	0
0	1	0
1	0	0
1	1	1



AND

Truth Table

Truth table.

- 16 Boolean functions of two variables.
- 2^2^N Boolean functions of N variables!

Truth Table for Some Functions of 2 Variables									
x	y	AND	NAND	OR	NOR	EQ	XOR	1	0
0	0	0	1	0	1	1	0	1	0
0	1	0	1	1	0	0	1	1	0
1	0	0	1	1	0	0	1	1	0
1	1	1	0	1	0	1	0	1	0

Truth Table

Truth table.

- 16 Boolean functions of 2 variables.
- 256 Boolean functions of 3 variables.
- 2^2^N Boolean functions of N variables!

Some Functions of 3 Variables							
x	y	z	AND	OR	MAJ	ODD	MUX
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	0	1	0	1	0
0	1	1	0	1	1	0	1
1	0	0	0	1	0	1	1
1	0	1	0	1	1	0	0
1	1	0	0	1	1	0	1
1	1	1	1	1	1	1	1

Universality of AND, OR, NOT

Any Boolean function can be expressed using AND, OR, NOT.

- "Universal."
- XOR(x,y) = xy' + x'y

Expressing XOR Using AND, OR, NOT							
x	y	x'	y'	x'y	xy'	x'y + xy'	XOR
0	0	1	1	0	0	0	0
0	1	1	0	1	0	1	1
1	0	0	1	0	1	1	1
1	1	0	0	0	0	0	0

Exercise: {AND, NOT}, {OR, NOT}, {NAND} are universal.

Sum-of-Products

Any Boolean function can be expressed using AND, OR, NOT.

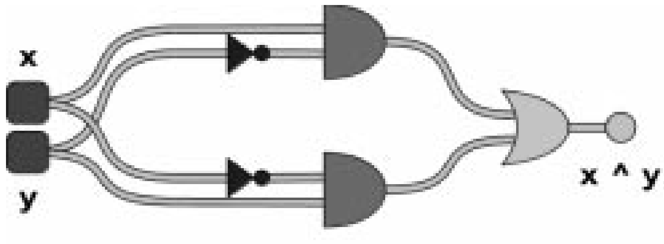
- Sum-of-products is systematic procedure.
  - form AND term for each 1 in Boolean function
  - OR terms together

Expressing MAJ Using Sum-of-Products								
x	y	z	MAJ	x'yz	xy'z	xyz'	xyz	x'yz + xy'z + xyz' + xyz
0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0	1
1	0	0	0	0	0	0	0	0
1	0	1	1	0	1	0	0	1
1	1	0	1	0	0	1	0	1
1	1	1	1	0	0	0	1	1

Translate Boolean Formula to Boolean Circuit

Use sum-of-products form.

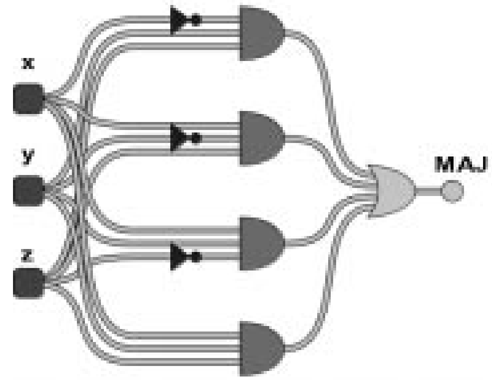
- XOR(x, y) = x'y + xy'.



Translate Boolean Formula to Boolean Circuit

Use sum-of-products form.

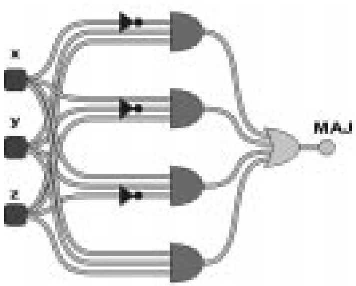
- MAJ(x, y, z) = x'yz + xy'z + xyz' + xyz.



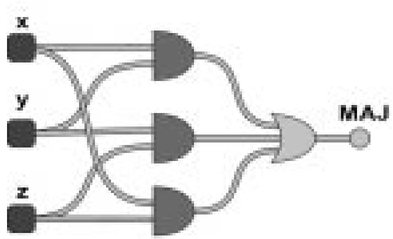
Simplification Using Boolean Algebra

Many possible circuits for each Boolean function.

- Sum-of-products not optimal.
- MAJ(x, y, z) = x'yz + xy'z + xyz' + xyz = xy + yz + xz.



size = 8, depth = 4



size = 4, depth = 3

Recipe for Making Combinational Circuit

Step 1.

- Represent input and output signals with Boolean variables.

Step 2.

- Construct truth table to carry out computation.

Step 3.

- Derive (simplified) Boolean expression using sum-of-products.

Step 4.

- Transform Boolean expression into circuit.

Let's Make an Adder Circuit

Goal: $x + y = z$.

Step 1.

- Represent input and output in binary.
- We build 4-bit adder: 8 inputs, 4 outputs.

	1	1	1	
	2	4	8	7
+	3	5	7	9
	6	1	6	6

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

	c_3	c_2	c_1	
	x_3	x_2	x_1	x_0
+	y_3	y_2	y_1	y_0
	z_3	z_2	z_1	z_0

Let's Make an Adder Circuit

Goal: $x + y = z$.

Step 2.

- Build truth table.

	x_3	x_2	x_1	x_0
+	y_3	y_2	y_1	y_0
	z_3	z_2	z_1	z_0

Adder Truth Table

x_0	x_1	x_2	x_3	y_0	y_1	y_2	y_3	z_0	z_1	z_2	z_3
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	1	1	0	0	1	1
0	0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	1	0	1	0	1
.	.	.	.	.	.	.	.	.	.	.	.
1	1	1	1	1	1	1	1	1	1	1	0

$2^8 = 256$ rows!

Let's Make an Adder Circuit

Goal: $x + y = z$.

Step 2.

- Build truth table for carry bit.
- Build truth table for summand bit.

	c_3	c_2	c_1	$c_0 = 0$
	x_3	x_2	x_1	x_0
+	y_3	y_2	y_1	y_0
	z_3	z_2	z_1	z_0

Carry Bit

x_i	y_i	c_i	c_{i+1}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Summand Bit

x_i	y_i	c_i	z_i
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	1

Let's Make an Adder Circuit

Goal: $x + y = z$.

Step 3.

- Derive (simplified) Boolean expression.

Carry Bit				
x_i	y_i	c_i	c_{i+1}	MAJ
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

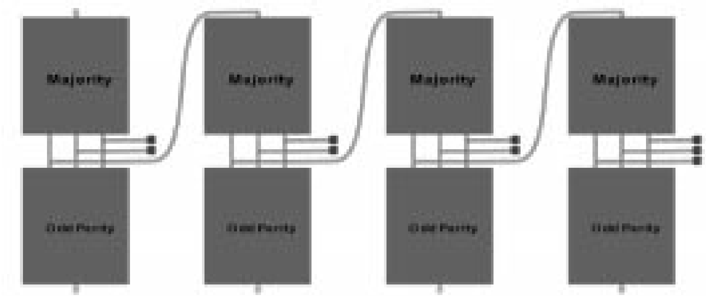
Summand Bit				
x_i	y_i	c_i	z_i	ODD
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Let's Make an Adder Circuit

Goal: x + y = z.

Step 4.

- **Transform Boolean expression into circuit.**

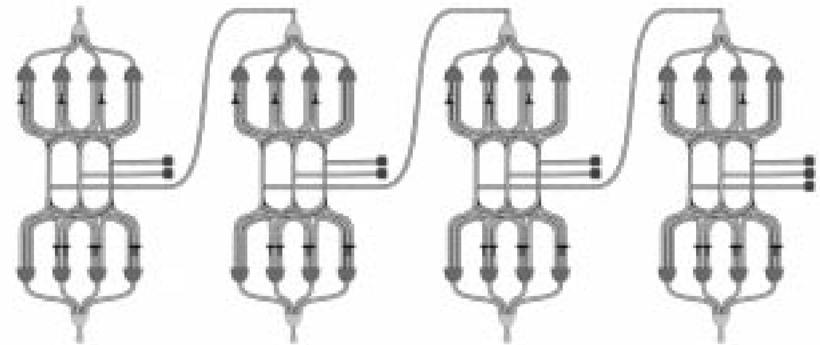


Let's Make an Adder Circuit

Goal: x + y = z.

Step 4.

- **Transform Boolean expression into circuit.**



Lecture A1: Extra Slides

ODD Parity Circuit

ODD(x, y, z).

- **1 if odd number of inputs are 1.**
- **0 otherwise.**

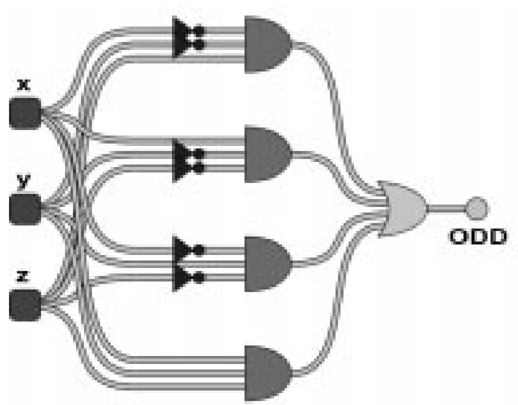
Expressing ODD Using Sum-of-Products

x	y	z	ODD	x'y'z	x'yz'	xy'z'	xyz	x'y'z + x'yz' + xy'z' + xyz
0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	1
0	1	0	1	0	1	0	0	1
0	1	1	0	0	0	0	0	0
1	0	0	1	0	0	1	0	1
1	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0
1	1	1	1	0	0	0	1	1

ODD Parity Circuit

ODD(x, y, z).

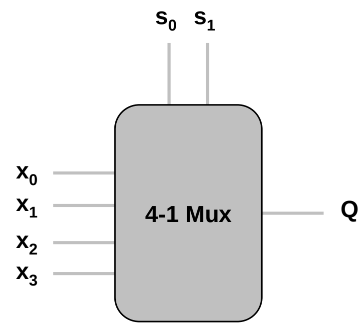
- 1 if odd number of inputs are 1.
- 0 otherwise.

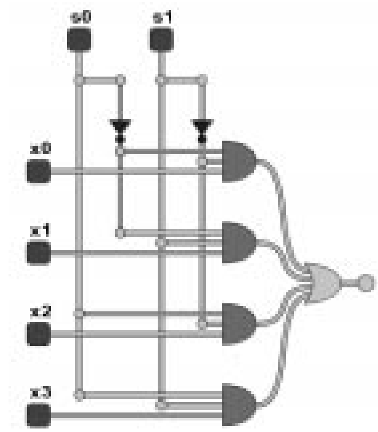


2-Bit Multiplexer Circuit

MUX(x_0, x_1, x_2, x_3, s_0, s_1).

- x_0 if $s_1 = 0$, $s_0 = 0$.
- x_1 if $s_1 = 0$, $s_0 = 1$.
- x_2 if $s_1 = 1$, $s_0 = 0$.
- x_3 if $s_1 = 1$, $s_0 = 1$.





Decoder

N-bit decoder.

- N inputs.
- 2^N outputs.
- Exactly one of outputs is 1; rest are 0.
- Which one?

Ex.

- 3-bit decoder.
- Inputs: x_0, x_1, x_2.
- Outputs: Q_0, Q_1, Q_2, …, Q_7.

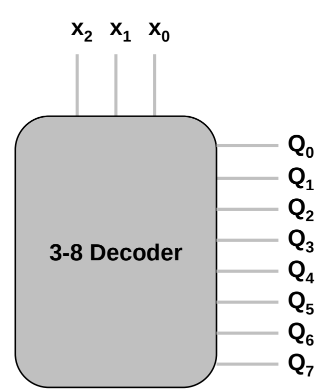
I_2	I_1	I_0
0	1	1

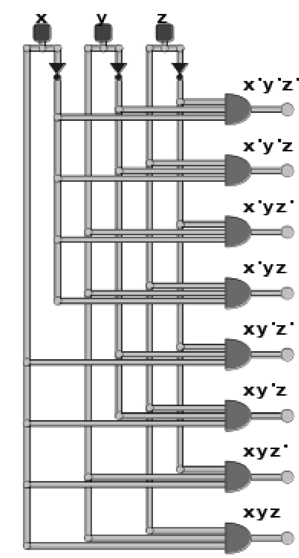
$11_2 = 3_{10}$

Q_7	Q_6	Q_5	Q_4	Q_3	Q_2	Q_1	Q_0
0	0	0	0	1	0	0	0

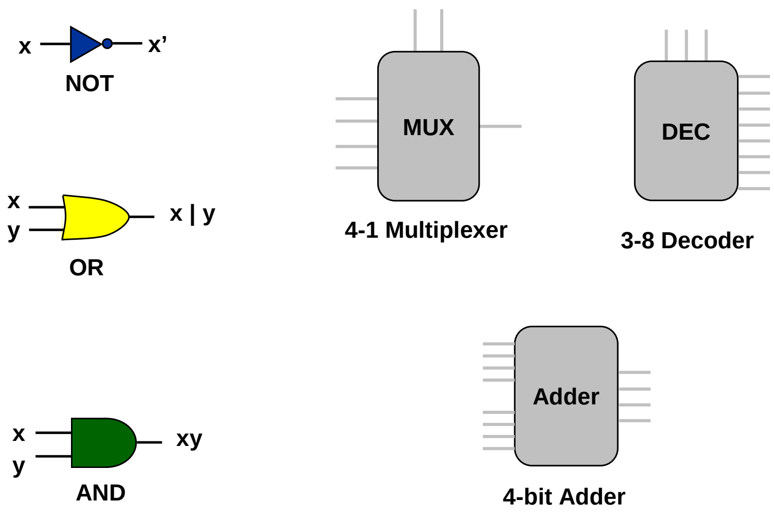
3-Bit Decoder

DECODE(x, y, z).





Cheat Sheet



x
x’
NOT
x
y
x | y
OR
x
y
xy
AND
MUX
4-1 Multiplexer
DEC
3-8 Decoder
Adder
4-bit Adder