Lecture 19

Andrea LaPaugh

April 16, 1996

Boolean Logic

functions of the form

f: $\{0,1\}^n \rightarrow \{0,1\}$

inputs: 0's and 1's

output: 0 or 1

Important in many aspects of computation

- program logic
- bit manipulations
- designing digital hardware

Boolean Logic and Digital Design

Interface between real and abstract machines

Abstract:

everything in machine represented

as binary numbers (sequence of 0-1 values)

Digital:

machines constructed of logic gates, wires

values ``1'' and ``0'' used by gates and carried by wires

(``1'' also referred to as ``on'' or ``high''

``0 also referred to as ``off'' or ``low'')

Real:

logic gates implemented as physical parts

voltages and currents interpreted as ``1''s and ``0''s

Boolean functions of two variables

Sixteen different functions

```
0 0 1 1    x
0 1 0 1    y

0 0 0 0    constant 0
0 0 0 1    AND
0 0 1 0
0 0 1 1    x
0 1 0 0
0 1 0 1    y
0 1 1 0    XOR
0 1 1 1    OR
1 0 0 0    NOR ("not or")
1 0 0 1    equivalence ("not xor")
1 0 1 0    NOT y
1 0 1 1
1 1 0 0    NOT x
1 1 0 1
1 1 1 0    NAND ("not and")
1 1 1 1    constant 1
```

Boolean functions of more than two variables

256 different functions of 3 variables 2^{2^3}

65,536 different functions of four variables 2^{2^4}

Each boolean function can be expressed as

combination of boolean functions on 2 variables

AND(x,y,z) = (x AND y) AND z

Can express all boolean functions with

only AND, OR and NOT (for example)

x XOR y = ((NOT x) AND y) OR (x AND (NOT y))

$$\overline{x}y + x\overline{y}$$

Sum-of-products

Systematic way to express any function
 AND, OR, and NOT

Look at "truth table" for function
 table of values

form AND terms for each 1 in the function
 use v if it corresponds to v = 1
 use v (NOT v) if it corresponds to v = 0

OR the terms together

Example: majority

x:	0	0	0	0	1	1	1	1
y:	0	0	1	1	0	0	1	1
z:	0	1	0	1	0	1	0	1
majority:	0	0	0	1	0	1	1	1

$\overline{x}yz + x\overline{y}z + xy\overline{z} + xyz$

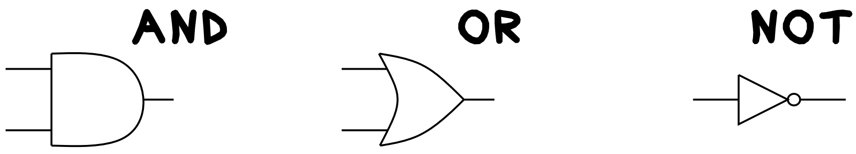
Example: odd parity

x:	0	0	0	0	1	1	1	1
y:	0	0	1	1	0	0	1	1
z:	0	1	0	1	0	1	0	1
odd parity:	0	1	1	0	1	0	0	1

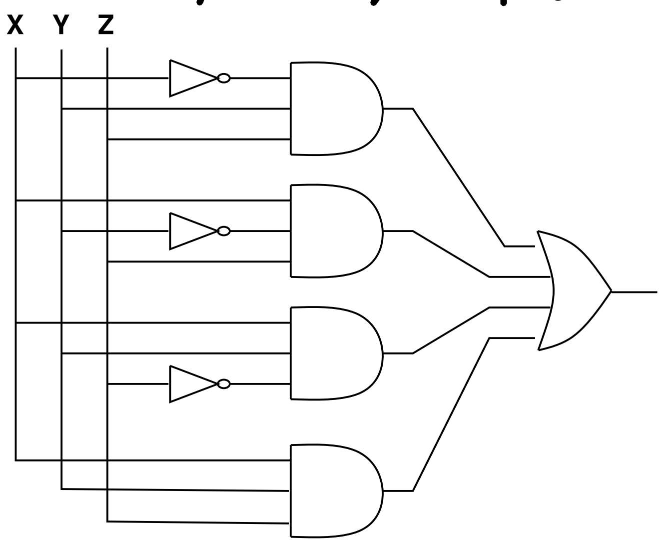
$\overline{x}\,\overline{y}\,z + \overline{x}\,y\,\overline{z} + x\,\overline{y}\,\overline{z} + xyz$

Boolean circuits

made up of interconnected "GATES"
values of variables are on the wires

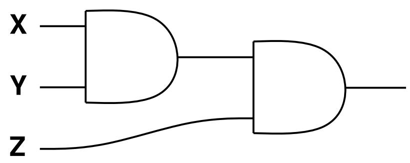


Example: majority $\overline{x}yz$ + $x\overline{y}z$ + $xy\overline{z}$ + xyz



Implementing Boolean circuits

Multi-input (>2) gates,
can do with two-input gates:



ANY boolean function can be implemented

use sum of products expression for function

just position inverters at inputs

position an AND gate for each AND term

position one OR gate - gives function value

Implementation of gate is the basis of

the interface between

real and abstract machines

Gates are built as physical components

Switches, relays, vacuum tubes,

transistors

Example: building an Adder

Goal: add two N-bit numbers x and y

Not a simple bit-by-bit function of the inputs, because of "carry"

$$\begin{array}{ccc} & 1 & 1 \\ & 0 & 1 \\ \hline 1 & 0 & 0 \end{array}$$

For each BIT

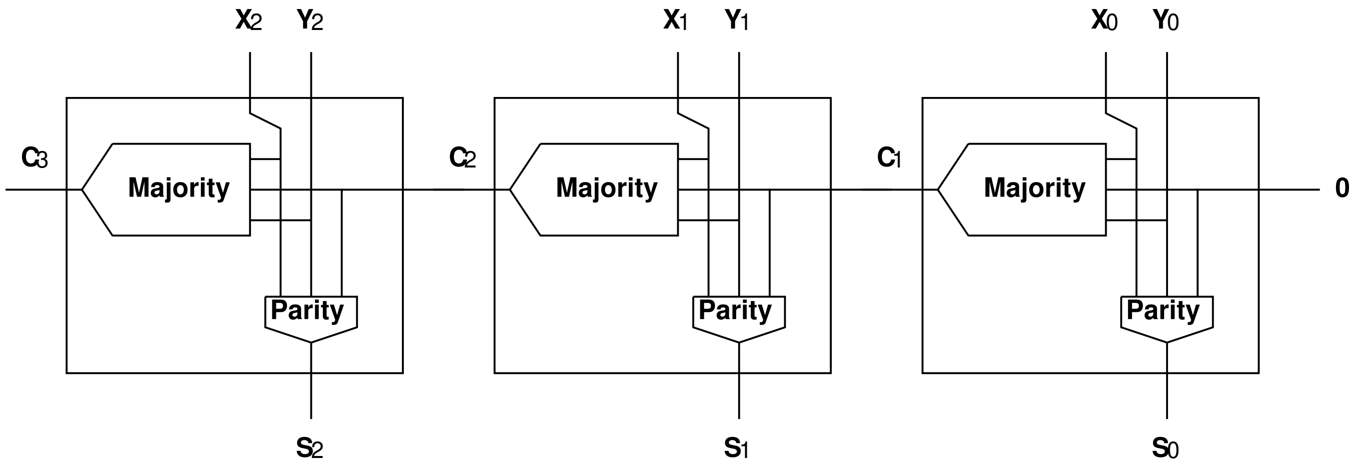
three inputs (x, y, carry)

one circuit computes sum

another circuit computes carry

Sum: odd parity

Carry: majority

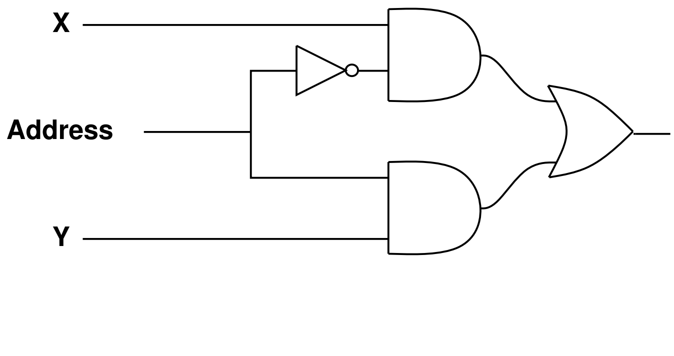


Other useful circuits

MULTIPLEXOR

N "data" inputs
log(N) "address" inputs
1 output
Chooses between N data inputs.

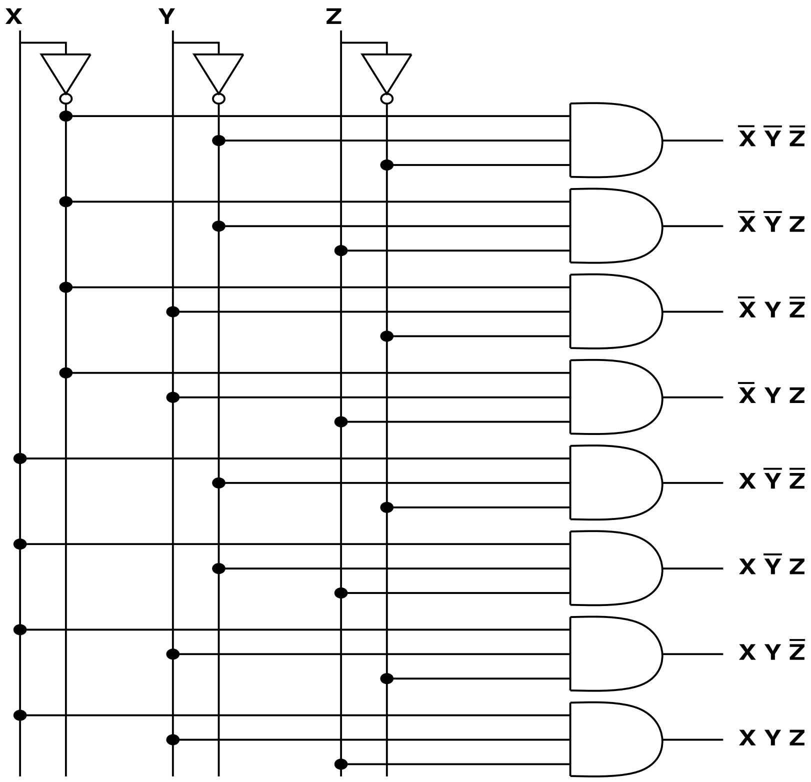
N=2:



Other useful circuits

DECODER

N "inputs"
2^N "outputs"
Precisely one "output" is "1" --
the one whose address is encoded in inputs



Converts binary to "unary"
Used to select wires by binary ID (address)