Networks of gates

how do we go from logic to gates?

eg start with truth table, build equivalent gates

$F(x,y)$ x,y inputs, network F gives the answer

ex:

x	y	F
0	0	1
0	1	0
1	0	0
1	1	1

construct F

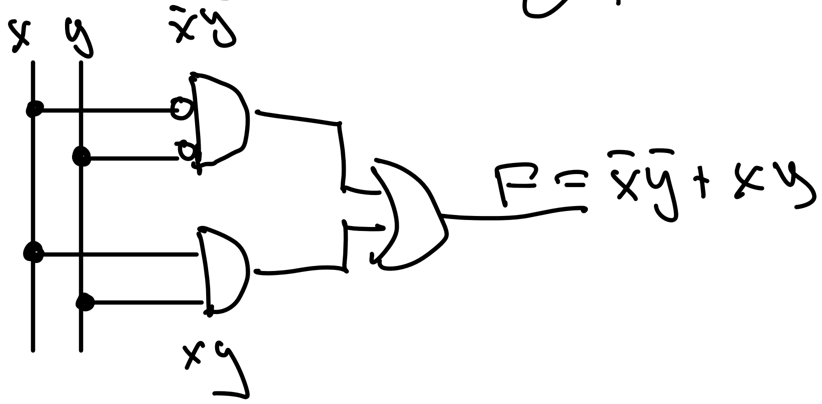
F is "true" (1) when x & y are the same

so $F=1$ when $\bar{x}\bar{y}$ or xy (drop "$\cdot$" so $xy \Rightarrow x\cdot y$)

or $F = \bar{x}\bar{y} + xy$

each "miniterm" ($\bar{x}\bar{y}$ & xy) is made up of products that result in $F=1$

F is the sum of products (SOP)



now apply deMorgan's thm

$$\bar{F} = \overline{\bar{x}\bar{y} + xy} = (\overline{\bar{x}\bar{y}})(\overline{xy})$$
$$= (x+y)(\bar{x}+\bar{y})$$
$$= x\bar{x} + x\bar{y} + y\bar{x} + y\bar{y}$$
$$= x\bar{y} + y\bar{x}$$
$$F = \overline{x\bar{y} + y\bar{x}} = (\overline{x\bar{y}})(\overline{y\bar{x}})$$
$$= (x+\bar{y})(\bar{x}+y)$$

this is a "product of sums" (POS)
where each miniterm is the or for where F=0

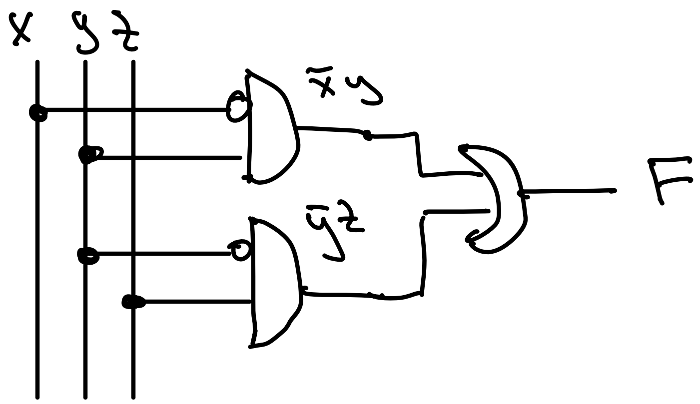
ex

x y z	F	
0 0 0	0	
0 0 1	1	$\bar{x}\bar{y}z$
0 1 0	1	$\bar{x}y\bar{z}$
0 1 1	1	$\bar{x}yz$
1 0 0	0	
1 0 1	1	$x\bar{y}z$
1 1 0	0	
1 1 1	0	

so $F = \overset{1}{\bar{x}\bar{y}z} + \overset{2}{\bar{x}y\bar{z}} + \overset{3}{\bar{x}yz} + \overset{4}{x\bar{y}z}$

simplify!

$$F = \overset{2\ 3}{\bar{x}y(\bar{z}+z)} + \overset{1\ 4}{(\bar{x}+x)\bar{y}z}$$
$$= \bar{x}y + \bar{y}z$$



In general: count how many times $F=0$ & $F=1$

if $\#F=1 < \#F=0$ then use SOP

$>$ POS

Binary, Decimal, Octal, Hex

number 3282 is implied as base 10 ("decimal")

1. base 10 $\Rightarrow$ 10 characters needed to represent each number (0, 1, ... 9)
2. "place" counting from right starting w/ $\emptyset$
3. each digit tells how many powers of 10^{place}

ex: place 3 2 1 0 base

number 3282_{10}

3×10^3 2×10^2 8×10^1 2×10^0

Base 2: Binary so need 2 digits 0, 1

place 4 3 2 1 0

number 1 1 0 0 1

1×2^4 1×2^3 0×2^2 0×2^1 1×2^0

$$11001_2 = (1\times16)+(1\times8)+1 = 25_{10}$$

How to convert from decimal to binary?

ex. 3282_{10}

need to know powers of 2

n	2^n	n	2^n	
0	1	6	64	etc.
1	2	7	128	
2	4	8	256	
3	8	9	512	
4	16	10	1024	
5	32	11	2048	

take 3282_{10} $\Rightarrow$ largest 2^n that fits is $n=11$, $2^n=2048$

$$\begin{array}{r} 3282 \\ 2048 \\ \hline 1234 \\ 1024 \\ \hline 210 \\ 128 \\ \hline 82 \\ 64 \\ \hline 18 \\ 16 \\ \hline 2 \\ 2 \\ \hline 0 \end{array}$$

1234 $\Rightarrow$ largest 2^n is $n=10$, $2^{10}=1024$

210 $\Rightarrow$ " " is $n=7$ $2^7=128$

82 $\Rightarrow$ $2^6=64$

18 $\Rightarrow$ $2^4=16$

2 $\Rightarrow$ 2^1

place	11	10	9	8	7	6	5	4	3	2	1	0
	1	1	0	0	1	1	0	1	0	0	1	0

$110011010010_2 = 3282_{10}$

Another algorithm (equivalent)

1. Start w/ 3282/2
2. remainder is $\emptyset$ so whatever binary rep is of 3282 will have a $\emptyset$ in the

"least significant bit" (LSB) digit

3. 3282 / 2 = 1641 remainder 0

iterate on each divided result

=> 1641 / 2 = 820 remainder 1 so next digit will be 1

3282 / 2 =	1641	r	0	LSB least sig bit
1641 / 2 =	820	r	1	
820 / 2 =	410	r	0	
410 / 2 =	205	r	0	
205 / 2 =	102	r	1	
102 / 2 =	51	r	0	
51 / 2 =	25	r	1	
25 / 2 =	12	r	1	
12 / 2 =	6	r	0	
6 / 2 =	3	r	0	
3 / 2 =	1	r	1	
1 / 2 =	0	r	1	MSB most sig bit

3282 = 110011010010 binary

look at groups of binary digits.

e.g. groups of 3 bits

possible values?

place: 2 1 0

lowest 000 = 0

highest $111 = 1\times2^2 + 1\times2^1 + 1\times2^0$

$= 7$

There are 8 possible values, 0, 1, ... 7

So we can take a binary number &

look at groups of 3 bits and calc value of each group

ex. $\underbrace{110}_{6}\underbrace{101}_{5}\underbrace{001}_{1}\underbrace{000}_{0}\underbrace{101}_{5}$

65105 is the representation in a base that has 8 symbols, 0-7

Base 8 "Octal" 8 symbols

binary ⟺ octal is easy using trick of dividing binary into groups of 3

Can also divide binary into groups of 4

largest value for 4 bits = $1\underbrace{111}_{7} = 8+7 = 15$, $2^3 = 8$

binary	hex
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

need 16 symbols:

0, 1, ... 9, A, B, C, D, E, F

Computers

Integers in binary form - how to represent integers in computers

easy: use binary

ex: 4-bit computer can store 16 possible numbers

what about neg numbers?

need to use 1 bit to specify + or −

sign bit 0 = +, 1 = −

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	−0
1001	−1
1010	−2
1011	−3
1100	−4
1101	−5
1110	−6
1111	−7

redundant

ones complement : if < 0, invert bits

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-7
1001	-6
1010	-5
1011	-4
1100	-3
1101	-2
1110	-1
1111	-0

redundant

Twos complement : 1s complement, add 1 to inverted number (< 0 only)

	1sC	2sC
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	-7	-8
1001	-6	-7
1010	-5	-6
1011	-4	-5
1100	-3	-4
1101	-2	-3
1110	-1	-2
1111	-0	-1

no redundancy

trick: 1110 = MSB(1) * -8

add to ~~bo~~ bottom 3 bits 110 = 6

6 - 8 = -2 ✓

Advantage of 2's comp:

no need to treat pos & neg differently for add or subtract

ex: 69+12 (8-bit computer)

$$\begin{array}{rl} & 11 \\ 69 = & 01000101 \\ 12 = & 00001100 \\ \hline & 01010001 \end{array}$$

64 + 16 + 1 = 81 ✓

how would you subtract 81-12?

same as 81+(-12)

-12 in 1's complement

$$\begin{array}{rl} 12 = & 00001100 \\ -12 = & 11110011 \end{array}$$

$$\begin{array}{rl} & 11111 \\ +81 = & 01010001 \\ -12 = & 11110011 \\ \hline & 101000100 \end{array}$$

not 69!

2's complement

$$\begin{array}{rl} 12 = & 00001100 \\ \text{1's com } -12 = & 11110011 \\ \text{2's com} & 11110100 \end{array}$$

$$\begin{array}{rl} & 111 \\ +81 = & 01010001 \\ +(-12) & 11110100 \\ \hline & 101000101 \end{array}$$

= 69 (drop last bit on left)

It's like "borrowing" in decimal arithmetic

Computer arithmetic: gates for adding

let x, y be binary

construct $S = x + y$ ← arithmetic, not OR

$x\,y$	S
00	0
01	1
10	1
11	2

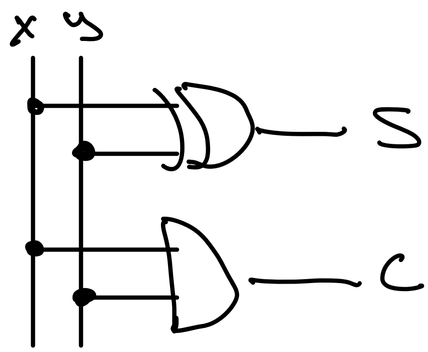
not binary symbol

need another bit for 2

$x\,y$	S	C
00	0	0
01	1	0
10	1	0
11	0	1

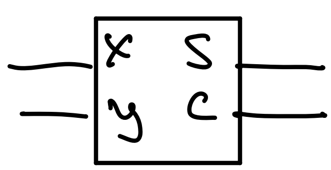
$C \equiv$ "carry" bit

gates: $S = x \oplus y$ & $C = xy$ easy



$\Rightarrow$

"primitive" 1-bit adder



you hook this circuit up as needed