

Logic Gates

22c:19
Hantao Zhang

1

Review of Boolean algebra

- Not is a horizontal bar above the number
 - $\bar{0} = 1$
 - $\bar{1} = 0$
- Or is a plus
 - $0+0 = 0$
 - $0+1 = 1$
 - $1+0 = 1$
 - $1+1 = 1$
- And is multiplication
 - $0*0 = 0$
 - $0*1 = 0$
 - $1*0 = 0$
 - $1*1 = 1$

2

Review of Boolean algebra

- Example: translate $(x+y+z)(\bar{x}\bar{y}\bar{z})$ to a Boolean logic expression
 - $(x \vee y \vee z) \wedge (\neg x \wedge \neg y \wedge \neg z)$
- We can define a Boolean function:
 - $F(x,y) = (x \vee y) \wedge (\neg x \wedge \neg y)$
- And then write a "truth table" for it:

x	y	F(x,y)
1	1	0
1	0	0
0	1	0
0	0	0

3

Basic logic gates

- Not $x \rightarrow \bar{x}$
- And $x, y \rightarrow xy$ $x, y, z \rightarrow xyz$
- Or $x, y \rightarrow x+y$ $x, y, z \rightarrow x+y+z$
- Nand $x, y \rightarrow \overline{xy}$
- Nor $x, y \rightarrow \overline{x+y}$
- Xor $x, y \rightarrow x \oplus y$

4

Converting between circuits and equations

- Find the output of the following circuit

- Answer: $(x+y)\bar{y}$
 – Or $(x \vee y) \wedge \neg y$

5

Converting between circuits and equations

- Find the output of the following circuit

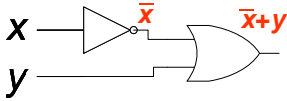
- Answer: $\overline{\bar{x}\bar{y}}$
 – Or $\neg(\neg x \wedge \neg y) \equiv x \vee y$

6

Converting between circuits and equations

- Write the circuits for the following Boolean algebraic expressions

a) $\bar{x}+y$

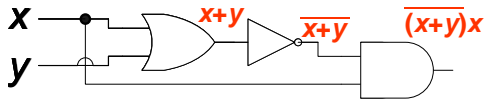


7

Converting between circuits and equations

- Write the circuits for the following Boolean algebraic expressions

b) $(x+y)x$

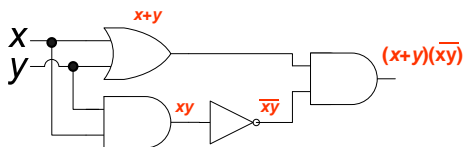


8

Implementing XOR

- $p \oplus q \equiv (p \vee q) \wedge \neg(p \wedge q)$
- $x \oplus y \equiv (x + y)(\overline{xy})$

x	y	xy	$x \oplus y$
1	1	1	0
1	0	0	1
0	1	0	1
0	0	0	0



9

Converting decimal numbers to binary

- $53 = 32 + 16 + 4 + 1$
 $= 2^5 + 2^4 + 2^2 + 2^0$
 $= 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$
 $= 110101$ in binary
 $= 00110101$ as a full byte in binary
- $211 = 128 + 64 + 16 + 2 + 1$
 $= 2^7 + 2^6 + 2^4 + 2^1 + 2^0$
 $= 1 \cdot 2^7 + 1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0$
 $= 11010011$ in binary

10

Converting binary numbers to decimal

- What is 10011010 in decimal?
 $10011010 = 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$
 $= 2^7 + 2^4 + 2^3 + 2^1$
 $= 128 + 16 + 8 + 2$
 $= 154$
- What is 00101001 in decimal?
 $00101001 = 0 \cdot 2^7 + 0 \cdot 2^6 + 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$
 $= 2^5 + 2^3 + 2^0$
 $= 32 + 8 + 1$
 $= 41$

11

How to add binary numbers

- Consider adding two 1-bit binary numbers x and y
 - $0+0=0$
 - $0+1=1$
 - $1+0=1$
 - $1+1=10$

x	y	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

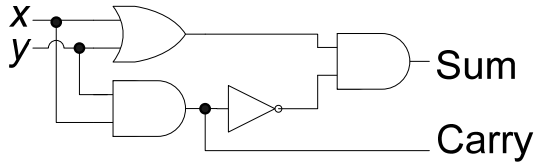
- Carry is x AND y
- Sum is x XOR y
- The circuit to compute this is called a half-adder

12

The half-adder

- Sum = $x \text{ XOR } y$
- Carry = $x \text{ AND } y$

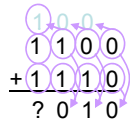
x	y	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



13

Using half adders

- We can then use a half-adder to compute the sum of two Boolean numbers



14

How to fix this

- We need to create an adder that can take a carry bit as an additional input
 - Inputs: $x, y, \text{carry in}$
 - Outputs: $\text{sum}, \text{carry out}$
- This is called a full adder
 - Will add x and y with a half-adder
 - Will add the sum of that to the carry in
- What about the carry out?
 - It's 1 if either (or both):
 - $x+y = 10$
 - $x+y = 01$ and carry in = 1

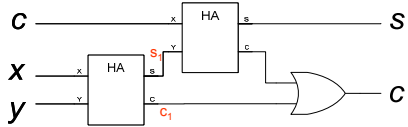
x	y	c	carry	sum
1	1	1	1	1
1	1	0	1	0
1	0	1	1	0
1	0	0	0	1
0	1	1	1	0
0	1	0	0	1
0	0	1	0	1
0	0	0	0	0

15

The full adder

- The "HA" boxes are half-adders

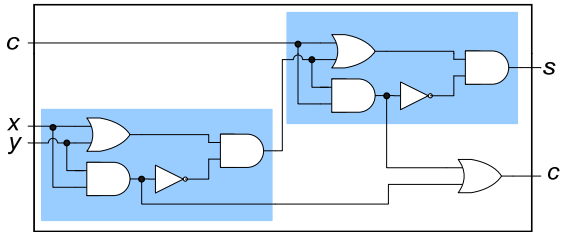
x	y	c	s ₁	c ₁	carry	sum
1	1	1	0	1	1	1
1	1	0	0	1	1	0
1	0	1	1	0	1	0
1	0	0	1	0	0	1
0	1	1	1	0	1	0
0	1	0	1	0	0	1
0	0	1	0	0	0	1
0	0	0	0	0	0	0



16

The full adder

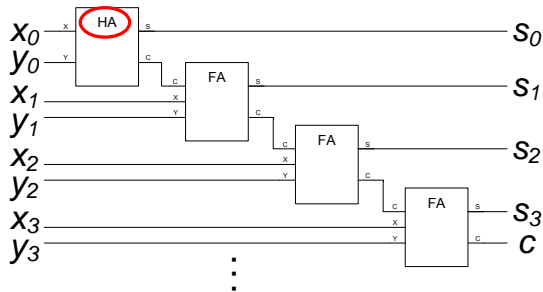
- The full circuitry of the full adder



17

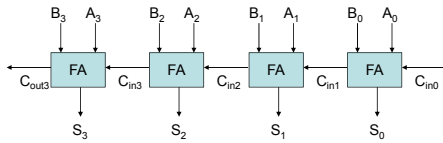
Adding bigger binary numbers

- Just chain full adders together

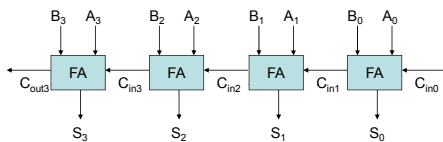


Ripple Carry Adder

- Put full-adders together to add numbers with multiple bits.



Delay of Carry Ripple Adder



- Ripple Adder too slow
 - Longest delay path: $2n$ gates
- Solution: Carry Look-Ahead Adder

Adding bigger binary numbers

- A half adder has 4 logic gates
- A full adder has two half adders plus a OR gate
 - Total of 9 logic gates
- To add n bit binary numbers, you need 1 HA and $n-1$ FAs
- To add 32 bit binary numbers, you need 1 HA and 31 FAs
 - Total of $4+9*31 = 283$ logic gates
- To add 64 bit binary numbers, you need 1 HA and 63 FAs
 - Total of $4+9*63 = 571$ logic gates

More about logic gates

- To implement a logic gate in hardware, you use a transistor
- Transistors are all enclosed in an “IC”, or integrated circuit
- The current Intel Pentium IV processors have 55 million transistors!

22

Representing negative numbers: Two's complement

- Let $c2(x)$ be the 2's complement of x , then $c2(x) = 2^n - x$, where n is the length of bits.
- Compute $c2(x)$:
 - 1. Let \bar{x} be the bit-wise complement of x
 - 2. Then $c2(x) = \bar{x} + 1$
- Example: Let $x = 1101100$.
 $\bar{x} = 0010011$, and $c2(x) = 0010100$.
- We use $c2(x)$ to represent $-x$.
- $c2(c2(x)) = x$ because $2^n - c2(x) = 2^n - (2^n - x) = x$
- $C2(x) = \bar{x} + 1$ because $x + \bar{x} = 11\dots1$ and $x + \bar{x} + 1 = 11\dots1 + 1 = 100\dots0 = 2^n$

Number representation in binary

Example: 8-bit numbers

Unsigned numbers: $10010111 = 151_{10}$

Signed numbers:

Signed-magnitude representation: $\overset{\text{sign}}{\downarrow} \overset{\text{magnitude}}{\uparrow} 10010111 = -23_{10}$

Signed 2's complement representation: $10010111 = ?$
If $c2(x) = -x = 0010111$, then $x = ?$

$$\begin{aligned} 10010111 &= 1 \cdot 2^7 + 0 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 \\ &= 2^7 + 2^4 + 2^2 + 2^1 + 2^0 \\ &= 128 + 16 + 7 = 128 + 23 = 151 \end{aligned}$$

$$\begin{aligned} x = c2(c2(x)) &= c2(0010111) = 01101001 \\ 10010111 &= -01101001 \\ &= -(2^6 + 2^5 + 2^3 + 2^0) \\ &= -(64 + 32 + 8 + 1) = -105 \end{aligned}$$

How to find the decimal representation of a 2's complement number

- Step 1: Look at the most significant bit. If it's 0, then treat it like regular binary number. If 1, then go to Step 2
- Step 2: For a negative number n, find $n' = c2(n)$. Now, $n' = -n$, also n' is positive. Find the decimal representation of n' . Let this be d. Then, the decimal representation of n is -d.
- Example: What is 10010111? (Assume that 10010111 is in 2's complement representation)
- Most significant bit is 1. Therefore, take its 2's complement, 01101001. This number in decimal is 105. Therefore, $10010111_{2c} = -105_{10}$

Subtraction

• $X - Y = X + c2(Y)$

• Example:

Let X = 1010100
Let Y = 1000011
Find X - Y.

Answer:
X = 1010100
-Y = 0111101 (= c2(Y), 2's complement)
X-Y = X+(-Y) = 10010001
Discarding the carry bit:
X-Y = 0010001

• Exercise: Perform 9 - 6 in binary (8 bits)

Overflow

- Overflow in unsigned addition: Last carry-out is 1.
- Can occur when the sum is too big to represent.
- Need to detect overflow and alert user.
- For 2's complement addition and subtraction, last carry-out being 1 doesn't necessarily mean overflow has occurred.
- For 2's complement,
 - Overflow cannot possibly occur if we're adding a negative to a positive number.
 - Overflow has occurred if the sum of two positive numbers is negative, or the sum of two negative numbers is positive

Hexadecimal

- A numerical range from 0-15
 - Where A is 10, B is 11, ... and F is 15
- Often written with a '0x' prefix
- So 0x10 is 10 hex, or 16
 - 0x100 is 100 hex, or 256
- Binary numbers easily translate:

Table 1.5.3

Decimal	Hexadecimal	4-Bit Binary Equivalent
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

© 2002 Thomson Higher Education
