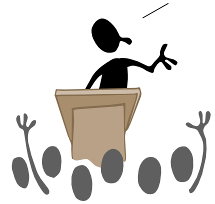
第一章 信息和编码

编码的目的
二进制编码

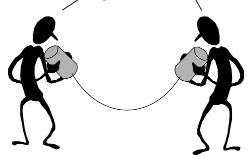
处理信息，
用电路表达

What is “Information 信息”?

information, *n.*
Knowledge, communicated or received concerning a particular fact or circumstance.



男足又输了



减少不确定性，香农信息熵

Information resolves uncertainty.
Information is simply that which cannot be predicted.
The less predictable a message is, the more information it conveys!

Quantifying Information 度量信息
(Claude Shannon, 1948)

Suppose you're faced with N equally probable choices, and I give you a fact that narrows it down to M choices. Then I've given you

$\log_2(N/M)$ bits of information

Information is measured in bits (binary digits) = number of 0/1's required to encode choice(s)

Examples:
- information in one coin flip: $\log_2(2/1) = 1$ bit
- roll of 2 dice: $\log_2(36/1) = 5.2$ bits

Encoding 编码

- Encoding describes the process of *assigning representations to information*
- Choosing an appropriate and efficient encoding is a real engineering challenge
- Impacts design at many levels
  - Mechanism (devices, # of components used)
  - Efficiency (bits used)
  - Reliability (noise)
  - Security (encryption)

100084

- 数制：表示数量的规则
- 码制：表示事物的规则

- 数制：
  ①每一位的构成
  ②从低位向高位的进位规则

我们常用到的：
十进制，二进制，八进制，十六进制

二进制，八进制，十进制，十六进制

- A binary digit has only 2 possibilities　逢二进一

0	1

- An octal digit has 8 possibilities　逢八进一

0	1	2	3	4	5	6	7

- A decimal digit has 10 possibilities　逢十进一

0	1	2	3	4	5	6	7	8	9

- A hexadecimal (hex) digital has 16 possibilities

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

逢十六进一

Encoding numbers

$$v = \sum_{i=0}^{n-1} 2^i b_i$$

2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	1	1	1	1	1	0	1	0	0	0	0

$= 2000_{10}$

7　d　0

03720　0x7d0

Oftentimes we will find it convenient to cluster groups of bits together for a more compact notation. Two popular groupings are clusters of 3 bits and 4 bits.

Octal - base 8

000 - 0
001 - 1
010 - 2
011 - 3
100 - 4
101 - 5
110 - 6
111 - 7

Hexadecimal - base 16

0000 - 0	1000 - 8
0001 - 1	1001 - 9
0010 - 2	1010 - a
0011 - 3	1011 - b
0100 - 4	1100 - c
0101 - 5	1101 - d
0110 - 6	1110 - e
0111 - 7	1111 - f

- 例

$$(101.11)_B = 1\times 2^2 + 0\times 2^1 + 1\times 2^0 + 1\times 2^{-1} + 1\times 2^{-2} = (5.75)_D$$

$$(2A.7F)_H = 2\times 16^1 + 10\times 16^0 + 7\times 16^{-1} + 15\times 16^{-2} = (42.4960937)_D$$

$$D = \sum K_i N^i$$

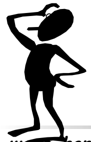
任意进制。。。

二进制数的补码：

- 算术运算
  二进制数的0/1可以表示数量，进行加，减，乘，除。。。等运算

5 + (-5) = 0
0 0101+ 1 0101=?

- 二进制数的正、负号也是用0/1表示的。
  在定点运算中，最高位为符号位（0为正，1为负）
  如 +5 =0 0101
  -5 =1 0101



Signed integers: 2's complement（二进制数的补码）

N bits 补码运算

-2^{N-1}	2^{N-2}	···	···	···	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}

"sign bit" Range: -2^{N-1} to $2^{N-1}-1$ "decimal" point

8-bit 2's complement example:
11010110
$= -2^7 + 2^6 + 2^4 + 2^2 + 2^1$
$= -128 + 64 + 16 + 4 + 2$
$= -42$

By moving the implicit location of "decimal" point, we can represent fractions too:
1101.0110
$= -2^3 + 2^2 + 2^0 + 2^{-2} + 2^{-3}$
$= -8 + 4 + 1 + 0.25 + 0.125$
$= -2.625$

二进制数的补码：

- 最高位为符号位（0为正，1为负）
- 正数的补码和它的原码相同
- 负数的补码 = 数值位逐位求反 + 1

+5 = (0 0101)
-5 = (1 1011)

两个补码表示的二进制数相加时的符号位讨论

例：用二进制补码运算求出

注意编码的取值范围

13 + 10 、13 – 10 、 – 13 + 10 、 – 13 – 10

+ 13	0	01101	+ 13	0	01101
+ 10	0	01010	– 10	1	10110
+ 23	0	10111	+ 3	0	00011

– 13	1	10011	– 13	1	10011
+ 10	0	01010	– 10	1	10110
– 3	1	11101	– 23	1	01001

结论：将两个加数的符号位和来自最高位数字位的进位相加，结果就是和的符号

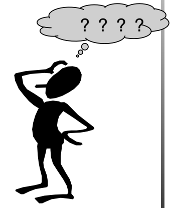
两个补码表示的二进制数相加时的符号位讨论

例：用二进制补码运算求出

注意编码的取值范围

13 + 10 、13 – 10 、 – 13 + 10 、 – 13 – 10

+ 13	0	1101	+ 13	0	1101
+ 10	0	1010	– 10	1	0110
+ 23	1	0111	+ 3	0	0011



– 13	1	0011	– 13	1	0011
+ 10	0	1010	– 10	1	0110
– 3	1	1101	– 23	0	1001

- 码制

用不同数码表示不同事物时遵循的规则
例如：学号，身份证号，车牌号。。。

- 目前，数字电路中都采用二进制
- 表示数量时称二进制
- 表示事物时称二值逻辑

Fixed-length encodings 等长编码

If all choices are equally likely (or we have no reason to expect otherwise), then a fixed-length code is often used. Such a code will use at least enough bits to represent the information content.

ex. Decimal digits 10 = {0,1,2,3,4,5,6,7,8,9}
4-bit BCD (binary code decimal)

$$\log_2(10) = 3.322 < 4\text{bits}$$

ex. ~86 English characters = {A-Z (26), a-z (26), 0-9 (10), punctuation (11), math (9), financial (4)} 7-bit ASCII (American Standard Code for Information Interchange)

$$\log_2(86) = 6.426 < 7\text{bits}$$

几种常用的十进制代码

十进制数	8421码	余3码	2421码	5211码	余3循环码
0	0000	0011	0000	0000	0010
1	0001	0100	0001	0001	0110
2	0010	0101	0010	0100	0111
3	0011	0110	0011	0101	0101
4	0100	0111	0100	0111	0100
5	0101	1000	1011	1000	1100
6	0110	1001	1100	1001	1101
7	0111	1010	1101	1100	1111
8	1000	1011	1110	1101	1110
9	1001	1100	1111	1111	1010

格雷码

每一位的状态变化都按一定的顺序循环。

编码顺序依次变化，按表中顺序变化时，相邻代码只有一位改变状态。

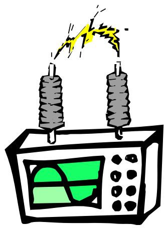
编码顺序	二进制	格雷码	编码顺序	二进制码	格雷码
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

变长编码，哈夫曼编码，解码

Stick with things we know about …

voltages phase
currents frequency

This semester we'll use voltages to encode information. But the best choice depends on the intended application...



Voltage pros:
- easy generation, detection
- lots of engineering knowledge
- potentially ~~low~~ zero power in steady state

Voltage cons:
- easily affected by environment
- DC connectivity required?
- R & C effects slow things down

Representing information with voltage

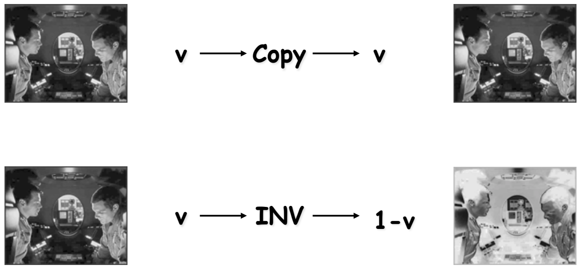
Representation of each point (x, y) on a Picture:

0 volts: BLACK
1 volts: WHITE
0.37 volts: 37% Gray
etc.

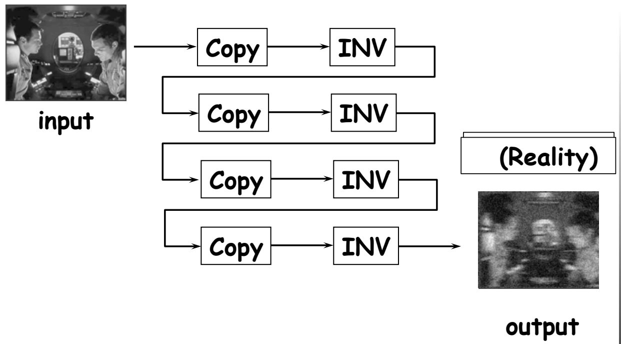
Representation of a picture:
Scan points in some prescribed raster order… generate voltage waveform

Information Processing = Computation

- First let's introduce some processing blocks:



Let's build a system!



Why did our system fail?

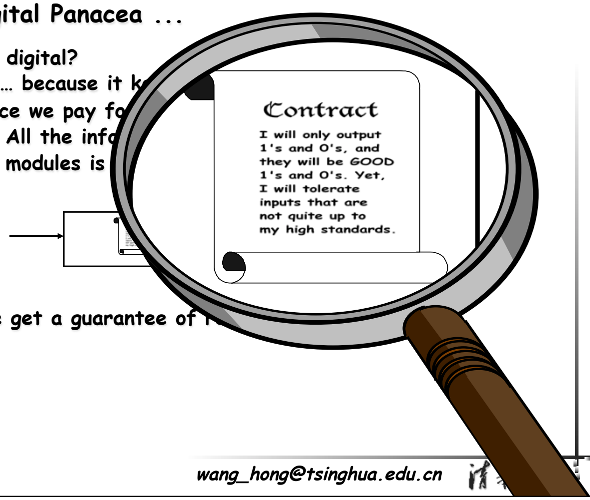
- Why doesn't reality match theory?
  1. COPY Operator doesn't work right
  2. INVERSION Operator doesn't work right
  3. Theory is imperfect
  4. Reality is imperfect
  5. Our system architecture stinks

- ANSWER: **all** of the above!
  Noise and inaccuracy are inevitable; we can't reliably reproduce infinite information-- we must design our system to tolerate some amount of error if it is to process information reliably.

The Digital Panacea ...

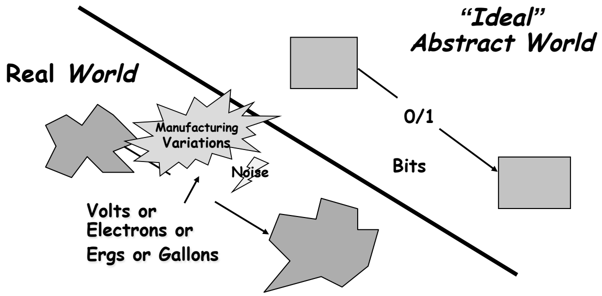
- Why digital?
  … because it k

The price we pay fo
  All the info
  modules is



But, we get a guarantee of

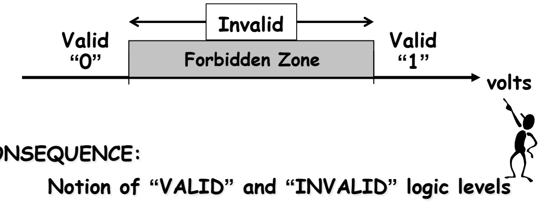
The Digital Abstraction



Keep in mind that the world is not digital, we would simply like to engineer it to behave that way. Furthermore, we must use real physical phenomena to implement digital designs!

Using Voltages "Digitally"

- Key idea: don't allow "0" to be mistaken for a "1" or vice versa
- Use the same "uniform representation convention" for *every* component and wire in our digital system To implement devices with high reliability, we outlaw "close calls" via a representation convention which forbids a range of voltages between "0" and "1".

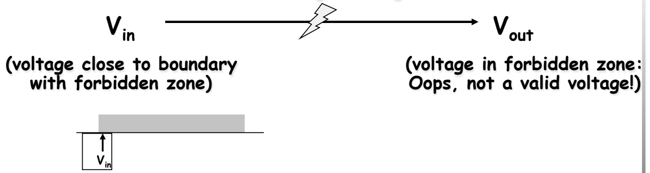


CONSEQUENCE:

Notion of "VALID" and "INVALID" logic levels

Wires: theory vs. practice

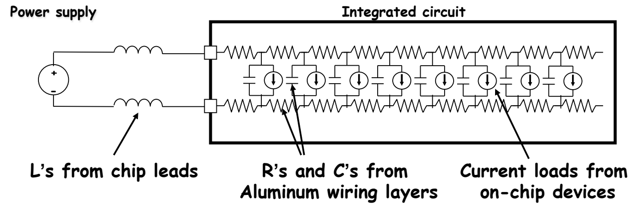
Does a wire obey the static discipline?



Questions to ask ourselves:

In digital systems, where does noise come from?
How big an effect are we talking about?

Power Supply Noise



ΔV from:

- **IR drop**
  (between gates: 30mV, within module: 50mV, across chip: 350mV)
- **L(dI/dt) drop**
  (use extra pins and bypass caps to keep within 250mV)
- **LC ringing triggered by current "steps"**