



中国科学院大学
University of Chinese Academy of Sciences

计算机科学导论

期末复习 学科展望

徐志伟 zxu@ict.ac.cn

提纲

- 期中复习与计算机学科展望
 - 1. 计算机学科的研究对象
 - 2. 计算机学科的基本研究方法
 - 计算思维基本解题思路：PEPS
 - 对计算思维的十个理解：Acu-Exams-CP
 - 3. 重新审视高德纳算法定义
 - 4. 为什么要教“自动机”
 - 5. 计算机学科发展的演化树
 - 6. 格雷12问题
 - 7. 系统思维的“以一耦万”抽象栈思想
 - 8. 信息隐藏程序回顾：自顶向下设计
 - 9. 用时序电路设计4位串行加法器
 - 10. 互联网通信示例

课件中可能包含素材引用，特此致谢！

1. 计算机科学的研究对象

比照《计算机科学基础报告》

- “计算机科学是研究计算机以及它们能干什么的一门学科。它研究**抽象计算机**的能力与局限，**真实计算机**的构造与特征，以及用于求解问题的无数**计算机应用**。”
 - 计算机科学涉及**符号**及其操作；
 - 关注多种**抽象**概念的创造和操作；
 - 创造并研究**算法**；
 - 创造各种**人工结构**，尤其是不受物理定律限制的结构；
 - 利用并应对**指数增长**；
 - 探索计算能力的**基本极限**；
 - 关注与人类**智能**相关的复杂的、分析的、理性的活动。

知道概念，能够举例说明其特点

笔记本电脑

有限状态机、图灵机、
冯诺依曼模型

- “计算机科学是研究计算机以及它们能干什么的一门学科。它研究**抽象计算机**的能力与局限，**真实计算机**的构造与特征，以及用于求解问题的无数**计算机应用**。”

- 计算机科学涉及**符号**及其操作；
- 关注多种**抽象**概念的创造和操作；
- 创造并研究**算法**；

快速排序程序

各种编程抽象

逻辑推理
命题逻辑
谓词逻辑

造各种**人工结构**，尤其是不受物理定律限制的结构；
用并应对**指数增长**；

斐波那契递归程序
P与NP

- 探索计算能力的**基本极限**；
- 关注与人类**智能**相关的复杂的、分析的、理性的活动。

图灵机不可计算问题
哥德尔不完备定理

计算思维概念

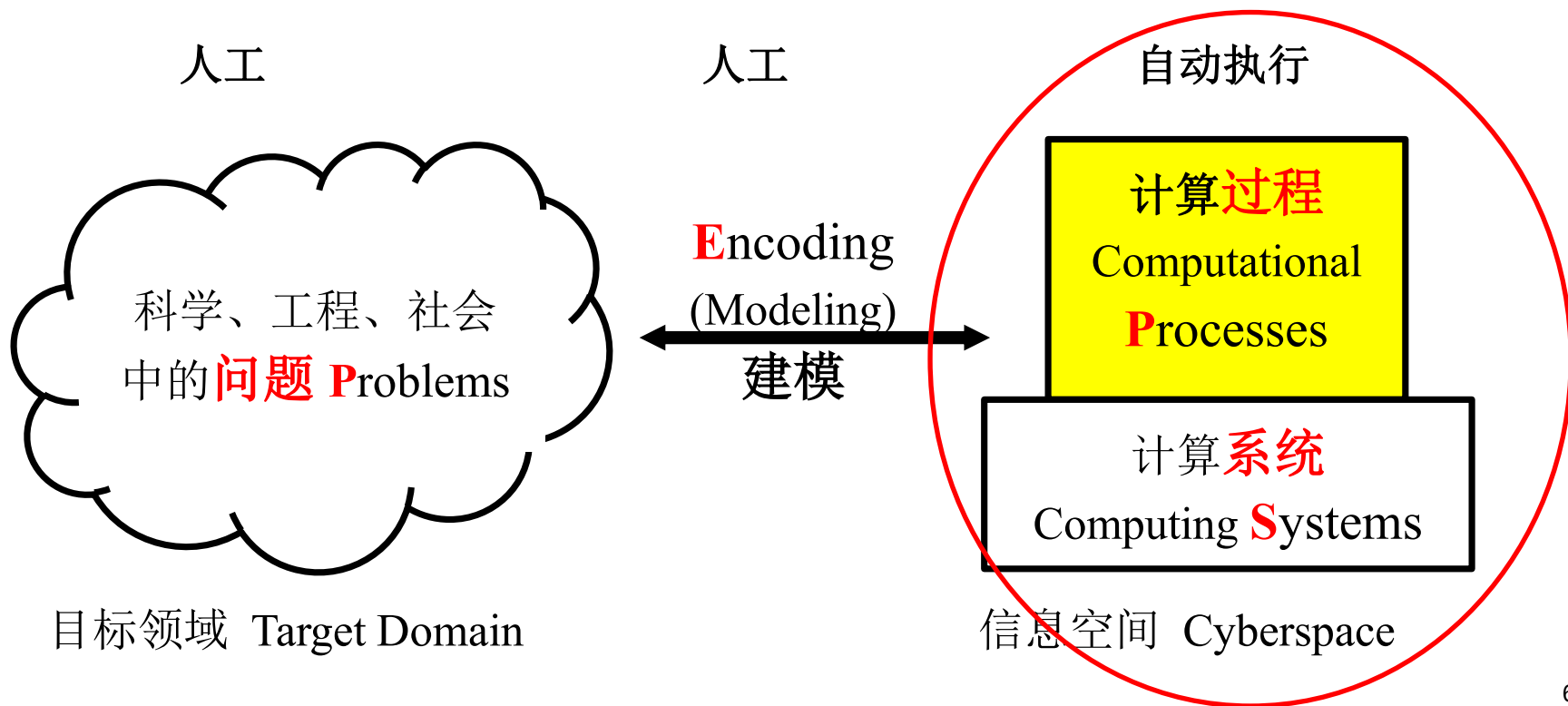
对计算思维的十个理解: **Acu-Exams-CP**

计算机科学是研究计算过程的科学

- | | |
|------------------------------|------------------------------|
| (A): Automatically execution | 自动执行。计算机能够自动执行由离散步骤组成的计算过程。 |
| (C): Correctness | 正确性。计算机求解问题的正确性往往可以精确地定义并分析。 |
| (U): Universality | 通用性。计算机能够求解任意可计算问题。 |
| (E): Effectiveness | 构造性。人们能够构造出聪明的方法让计算机有效地解决问题。 |
| (X): Complexity | 复杂度。这些聪明的方法（算法）具备时间/空间复杂度。 |
| (A): Abstraction | 抽象化。少数精心构造的计算抽象可产生万千应用系统。 |
| (M): Modularity | 模块化。多个模块有规律地组合成为计算系统。 |
| (S): Seamless Transition | 无缝衔接。计算过程在计算系统中流畅地执行。 |
| (C): Connectivity | 连接性。很多问题涉及用户/数据/算法的连接体，而非单体。 |
| (P): Protocol Stack | 协议栈。连接体的节点之间通过协议栈交互。 |

2. 计算思维基本方法：活力解题法（PEPS）

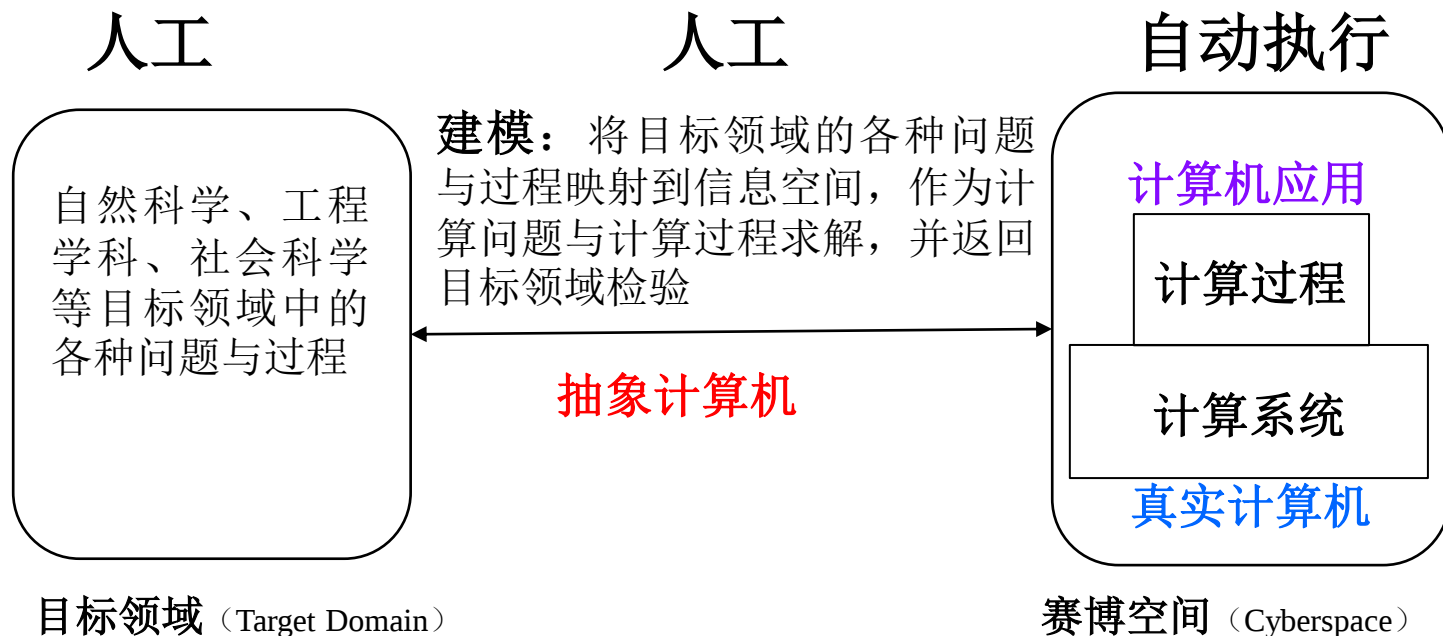
- 定义领域**问题**（**P**roblem in target domain）
- **建模**（**E**ncoding）到信息空间（赛博空间，cyberspace）的计算问题
- 自动执行计算**系统**（**S**ystem）上的计算**过程**（**P**rocess），解决问题
- 映射回到目标领域检验



计算思维基本方法

- 一种认识世界、定义问题、解决问题的科学方法
 - 计算系统 = 计算机
 - 使用算法、程序、状态转移表等刻画计算过程
- 建模是双向映射
- 目标领域也可以是赛博空间

计算机科学研究**抽象计算机**，**真实计算机**，**计算机应用**。



3. 重新审视高德纳的算法定义

● 确定性是什么意思？

一个算法是一组有限条规则，给出求解特定类型问题的操作序列，并具备下列五个特征：

- ① 有限性：算法在有限步骤之后必然要终止。
- ② 确定性：算法的每个步骤都必须精确地（严格地和无歧义地）定义。
- ③ 输入：算法有零个或多个输入，在算法开始或中途动态地给定。
- ④ 输出：算法有一个或多个输出，输出是与输入有特定关系的数值。
- ⑤ 能行性：算法的所有操作必须是充分基本的，原则上一个人能够用笔和纸在有限时间内精确地完成它们。

冒泡排序算法

- 输入：待排序的数组 A ，数组 A 的长度 n 。
- 输出：排好序的数组 A 。
- 算法步骤描述：
for $i = 1$ to $n - 1$
 for $j = 1$ to $n - i$
 if $A[j] > A[j+1]$
 exchange $A[j]$ with $A[j+1]$.

什么不是算法？

- Ada的运算流程图
 - 程序行数应与问题规模无关
- 无穷循环不停机
- 包含下列操作
 - 比较两个物理线段的长度
 - If 线段A长度==线段B长度 {...}

重新审视高德纳的算法定义

● 确定性是什么意思？

一个算法是一组有限条规则，给出求解特定类型问题的操作序列，并具备下列五个特征：

- ① 有限性：算法在有限步骤之后必然要终止。
- ② 确定性：算法的每个步骤都必须精确地（严格地和无歧义地）定义。
- ③ 输入：算法有零个或多个输入，在算法开始或中途动态地给定。
- ④ 输出：算法有一个或多个输出。输出是与输入有特定关系的数值。
- ⑤ 能行性：算法的所有操作必须是充分基本的，原则上一个人能够用笔和纸在有限时间内精确地完成它们。

冒泡排序算法

- 输入：待排序的数组 A ，数组 A 的长度 n 。
- 输出：排好序的数组 A 。
- 算法步骤描述：
for $i = 1$ to $n - 1$
 for $j = 1$ to $n - i$
 if $A[j] > A[j+1]$
 exchange $A[j]$ with $A[j+1]$.

什么不是算法？

- Ada的运算流程图
 - 程序行数应与问题规模无关
- 无穷循环不停机
包含下列操作
 - 比较两个物理线段的长度
 - If 线段A长度==线段B长度 {...}

重新审视高德纳的算法定义

● 确定性是什么意思？

一个算法是一组有限条规则，给出求解特定类型问题的操作序列，并具备下列五个特征：

- ① 有限性：算法在有限步骤之后必然要终止。
- ② 确定性：算法的每个步骤都必须精确地（严格地和无歧义地）定义。

比特精准！

- ① 输入：算法有零个或多个输入，在算法开始或中途动态地给定。
- ② 输出：算法有一个或多个输出，输出是与输入有特定关系的数值。
- ③ 能行性：算法的所有操作必须是充分基本的，原则上一个人能够用笔和纸在有限时间内精确地完成它们。

冒泡排序算法

- 输入：待排序的数组 A ，数组 A 的长度 n 。
- 输出：排好序的数组 A 。
- 算法步骤描述：
for $i = 1$ to $n - 1$
 for $j = 1$ to $n - i$
 if $A[j] > A[j+1]$
 exchange $A[j]$ with $A[j+1]$.

什么不是算法？

- Ada的运算流程图
 - 程序行数应与问题规模无关
- 无穷循环不停机
- 包含下列操作
 - 比较两个物理线段的长度
 - If 线段A长度==线段B长度 {...}

重新审视高德纳的算法定义

● 注意三种有穷性

一个算法是一组**有限**条规则，给出求解特定类型问题的操作序列，并具备下列五个特征：

- ① 有限性：算法在**有限**步骤之后必然要终止。
- ② 确定性：算法的每个步骤都必须精确地（严格地和无歧义地）定义。
- ③ 输入：算法有零个或多个输入，在算法开始或中途动态地给定。
- ④ 输出：算法有一个或多个输出，输出是与输入有特定关系的数值。
- ⑤ 能行性：算法的所有操作必须是充分基本的，原则上一个人能够用笔和纸在**有限**时间内精确地完成它们。

冒泡排序算法

- 输入：待排序的数组 A ，数组 A 的长度 n 。
- 输出：排好序的数组 A 。
- 算法步骤描述：
for $i = 1$ to $n - 1$
 for $j = 1$ to $n - i$
 if $A[j] > A[j+1]$
 exchange $A[j]$ with $A[j+1]$.

什么不是算法？

- ~~Ada的运算流程图~~
 - 程序行数应与问题规模无关
- ~~无穷循环不停机~~
- ~~包含下列操作~~
 - 两个物理线段求余
 - ~~if哥德巴赫猜想为真{...}~~
 有限时间完不成！

重新审视高德纳的算法定义

● 注意三种有穷性

一个算法是一组**有限**条规则，给出求解特定类型问题的操作序列，并具备下列五个特征：

- ① 有限性：算法在**有限**步骤之后必然要终止。
- ② 确定性：算法的每个步骤都必须精确地（严格地和无歧义地）定义。
- ③ 输入：算法有零个或多个输入，在算法开始或中途动态地给定。
- ④ 输出：算法有一个或多个输出，输出是与输入有特定关系的数值。
- ⑤ 能行性：算法的所有操作必须是充分基本的，原则上一个人能够用笔和纸在**有限**时间内精确地完成它们。

冒泡排序算法

- 输入：待排序的数组 A ，数组 A 的长度 n 。
- 输出：排好序的数组 A 。
- 算法步骤描述：

```
for  $i = 1$  to  $n - 1$ 
  for  $j = 1$  to  $n - i$ 
    if  $A[j] > A[j+1]$ 
      exchange  $A[j]$  with  $A[j+1]$ .
```

规则有穷性

伪代码行数是 $O(1)$ ，与 n 无关

步骤有穷性

即算法的时间复杂度

操作能行性

任一操作是充分基本的，
所需时间肯定是 $O(1)$ ，很短

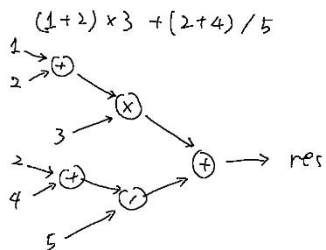
4. 为什么要教自动机（状态机）？

- 程序处理的数据通过程序这个结构“流”进来, 然后经过处理之后得到一个输出。然后只要将这样的in和out连在一起就能够组合生成更大的程序。然后中间不像是图灵机那样的通过改变自己状态来处理数据的思路, 更像是一种数据进来, 然后流出的映射关系。

```
# Ruby
[1, 2, 3].map {|i| i.to_s(16)}
  .join("0x").split("0x")
  .map {|i| i.to_i(16)}    # => [1, 2, 3]
```

Diagram illustrating the data flow for the Ruby code:

$[1, 2, 3] \xrightarrow{\text{block}} \text{map} \xrightarrow{"0x"} \text{join} \xrightarrow{"0x"} \text{split} \xrightarrow{\text{block}} \text{map} \rightarrow [1, 2, 3]$

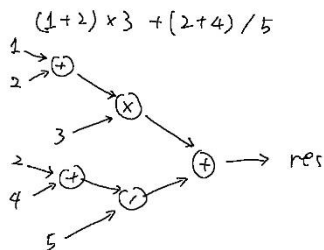


为什么要教自动机（状态机）？

- 程序处理的数据通过程序这个结构“流”进来, 然后经过处理之后得到一个输出。然后只要将这样的in和out连在一起就能够组合生成更大的程序。然后中间不像是图灵机那样的通过改变自己状态来处理数据的思路, 更像是一种数据进来, 然后流出的映射关系。
- 很好。但是, 这种模式似乎做不了快排。你看能画出快排的图吗? 或者书上的求30亿个数之和。
 - 事实上, 较难做转账操作: “中关村转账50给凉凉, 然后转账75给快快, 。。。”

```
# Ruby
[1, 2, 3].map {|i| i.to_s(16)}
  .join("0x").split("0x")
  .map {|i| i.to_i(16)} # => [1, 2, 3]
```

$[1, 2, 3] \xrightarrow{\text{block}} \text{map} \xrightarrow{"0x"} \text{join} \xrightarrow{"0x"} \text{split} \xrightarrow{\text{block}} \text{map} \rightarrow [1, 2, 3]$

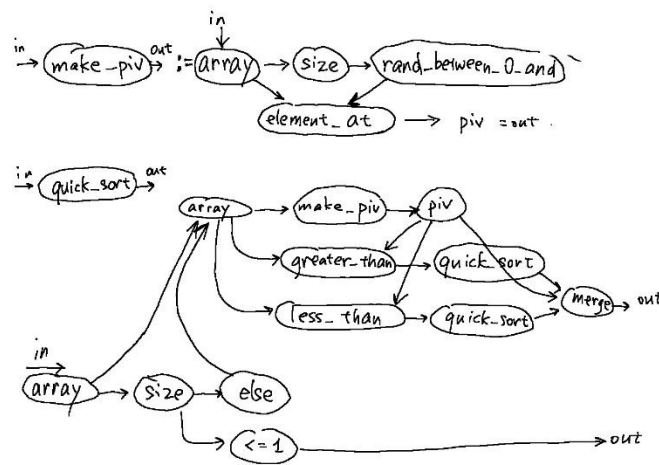
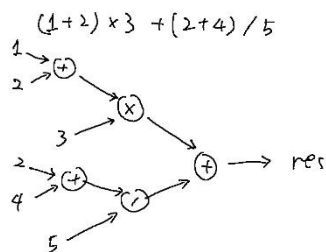


为什么要教自动机（状态机）？

- 程序处理的数据通过程序这个结构“流”进来, 然后经过处理之后得到一个输出。然后只要将这样的in和out连在一起就能够组合生成更大的程序。然后中间不像是图灵机那样的通过改变自己状态来处理数据的思路, 更像是一种数据进来, 然后流出的映射关系。
- 很好。但是, 这种模式似乎做不了快排。你看能画出快排的图吗? 或者书上的求30亿个数之和。
- 事实上, 很难做转账操作: “中关村转账50给凉凉”

```
# Ruby
[1, 2, 3].map {|i| i.to_s(16)}
  .join("0x").split("0x")
  .map {|i| i.to_i(16)} # => [1, 2, 3]
```

block "0x" "0x" block
[1, 2, 3] → map → join → split → map → [1, 2, 3]

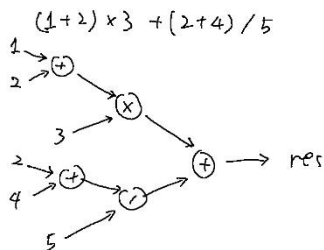


为什么要教自动机（状态机）？

- 70年的CS经验展示，它是最基础最通用的计算机理论模型
 - 其他还有很多模型
 - λ 演算，函数式编程，电路模型，...
- 不要学孔乙己的“回字有24种写法”
- 本课程的自动机思路贯穿整个抽象栈从Go语言一直到指令、甚至数字电路

程序
进程
指令
冯氏结构
指令流水线
时序电路
组合电路

```
#Ruby
[1, 2, 3].map {|i| i.to_s(16)}
  .join("0x").split("0x")
  .map {|i| i.to_i(16)} # => [1, 2, 3]
```



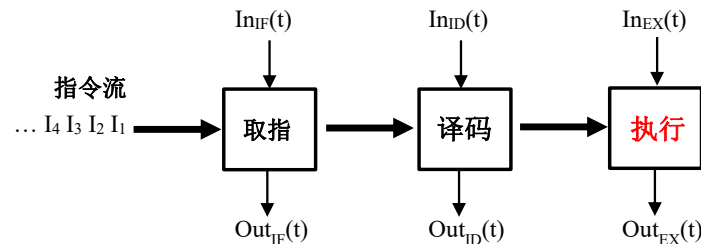
比特精准地将抽象映射到最底层硬件

万千应用

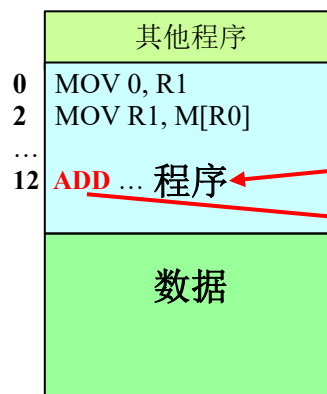
程序
进程
指令
冯氏结构
指令流水线
时序电路
组合电路

```
MOV 0, R1
MOV R1, M[R0]
MOV 1, R1
MOV R1, M[R0+8]
MOV 2, R2 // i:=2
MOV 0, R1 // label Loop
ADD M[R0+R2*8-16], R1
ADD M[R0+R2*8-8], R1
MOV R1, M[R0+R2*8-0]
INC R2 // i++
CMP 51, R2 // i < 51?
JL Loop // if '<' goto Loop
```

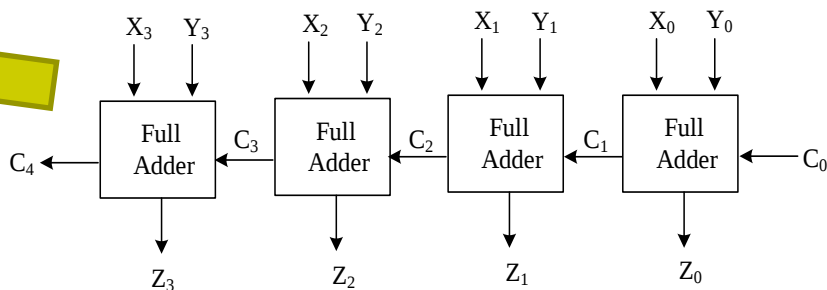
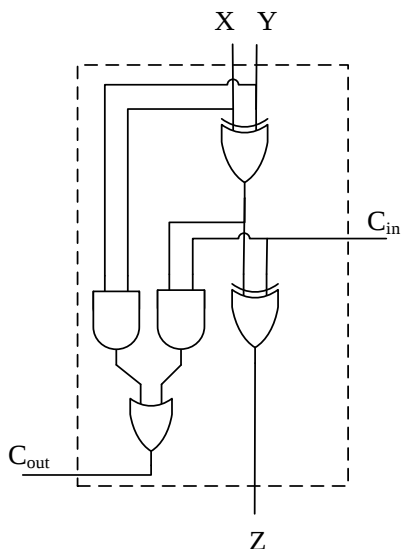
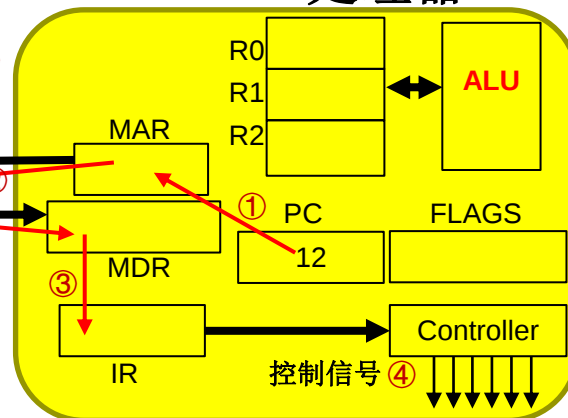
```
fib[0] = 0
fib[1] = 1
for i := 2; i < 51; i++ {
    fib[i] = fib[i-1] + fib[i-2]
}
```



内存



处理器

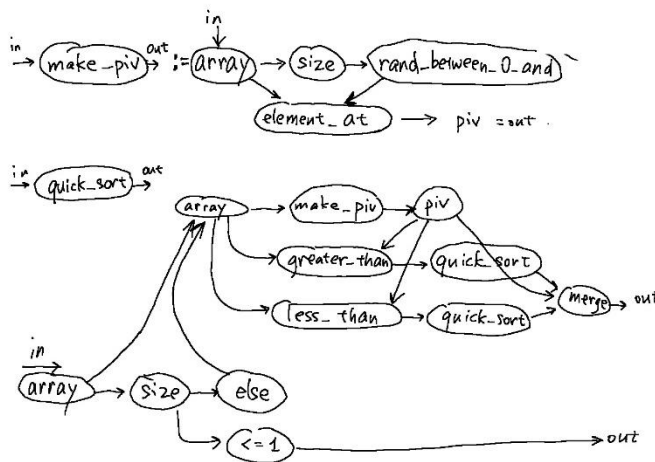
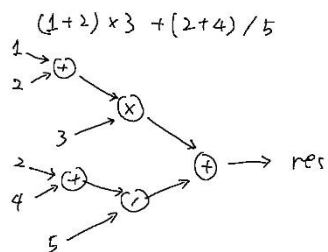


为什么要教自动机（状态机）？

- 70年的CS经验展示，它是最基础最通用的计算机理论模型
 - 其他还有很多模型
 - λ 演算，函数式编程，电路模型，...
- 不要学孔乙己的“回字有24种写法”
- 本课程的自动机思路贯穿整个抽象栈
从Go语言一直到指令、甚至数字电路
- 除了fmt.Printf，没有使用同学们不能做的操作

```
# Ruby
[1, 2, 3].map {|i| i.to_s(16)}
  .join("0x").split("0x")
  .map {|i| i.to_i(16)} # => [1, 2, 3]

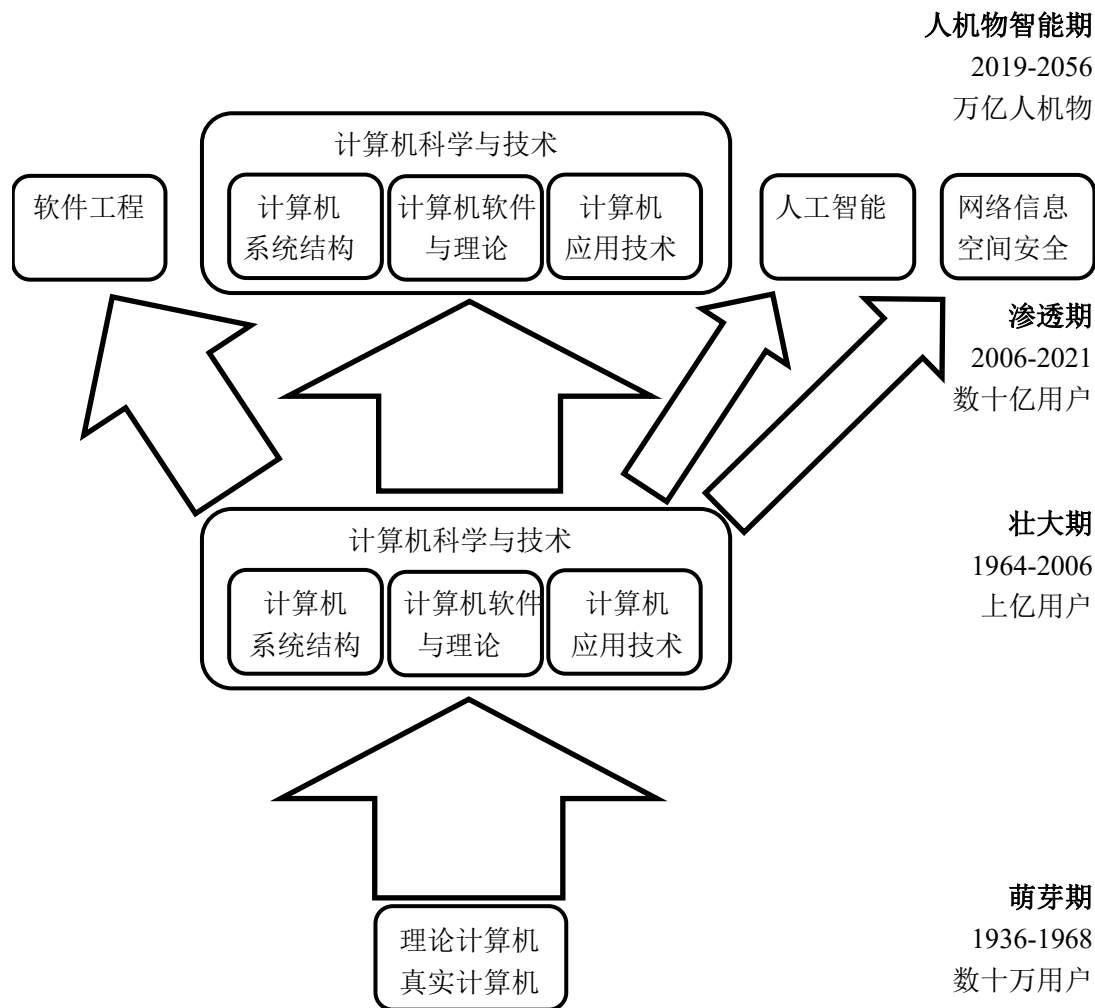
[1, 2, 3] → map → join → split → map → [1, 2, 3]
```



程序
进程
指令
冯氏结构
指令流水线
时序电路
组合电路

5. 计算机学科发展的演化树

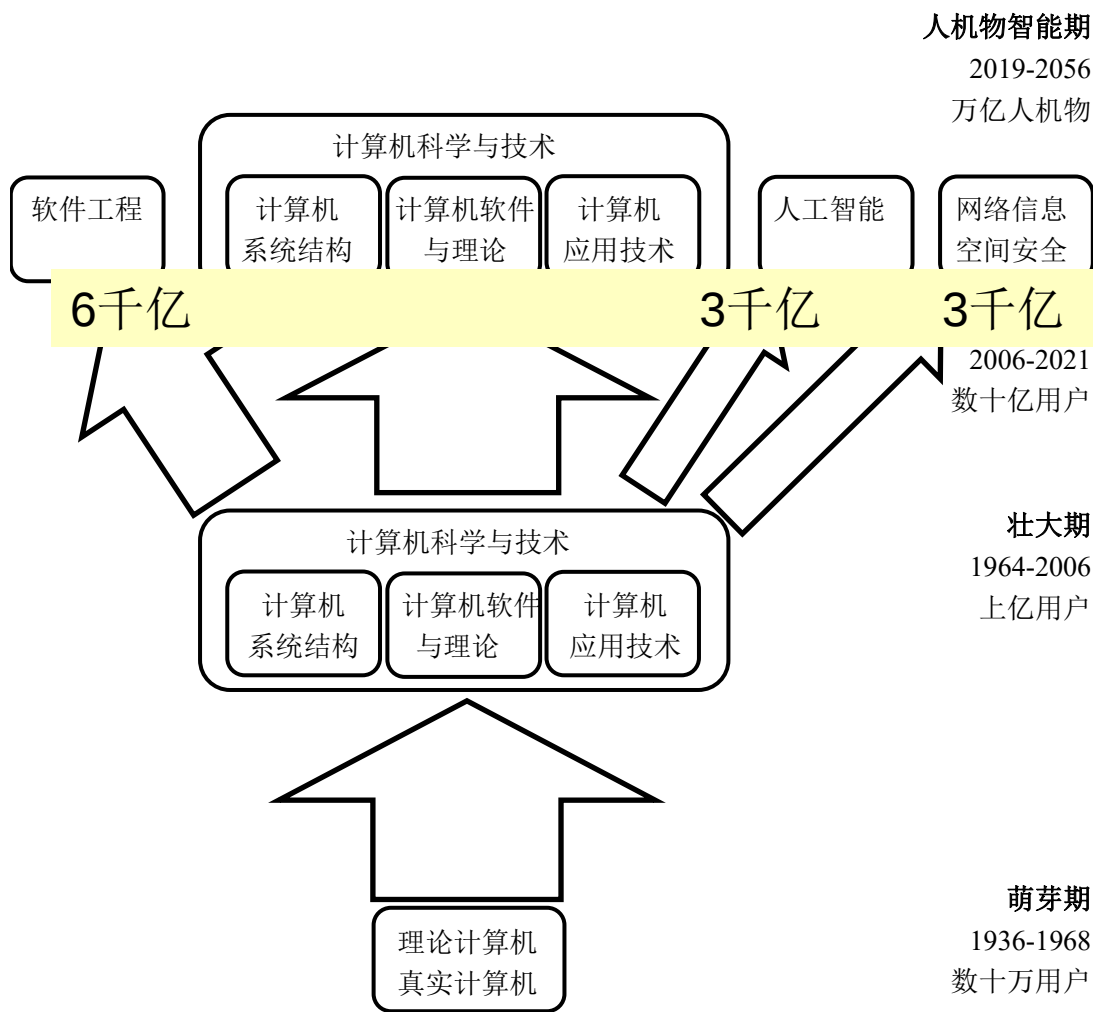
- 1936-2056，120年，大体上可分为四个时期



计算机学科发展演化树

- 1936-2022，前三个时期

- 全球市场

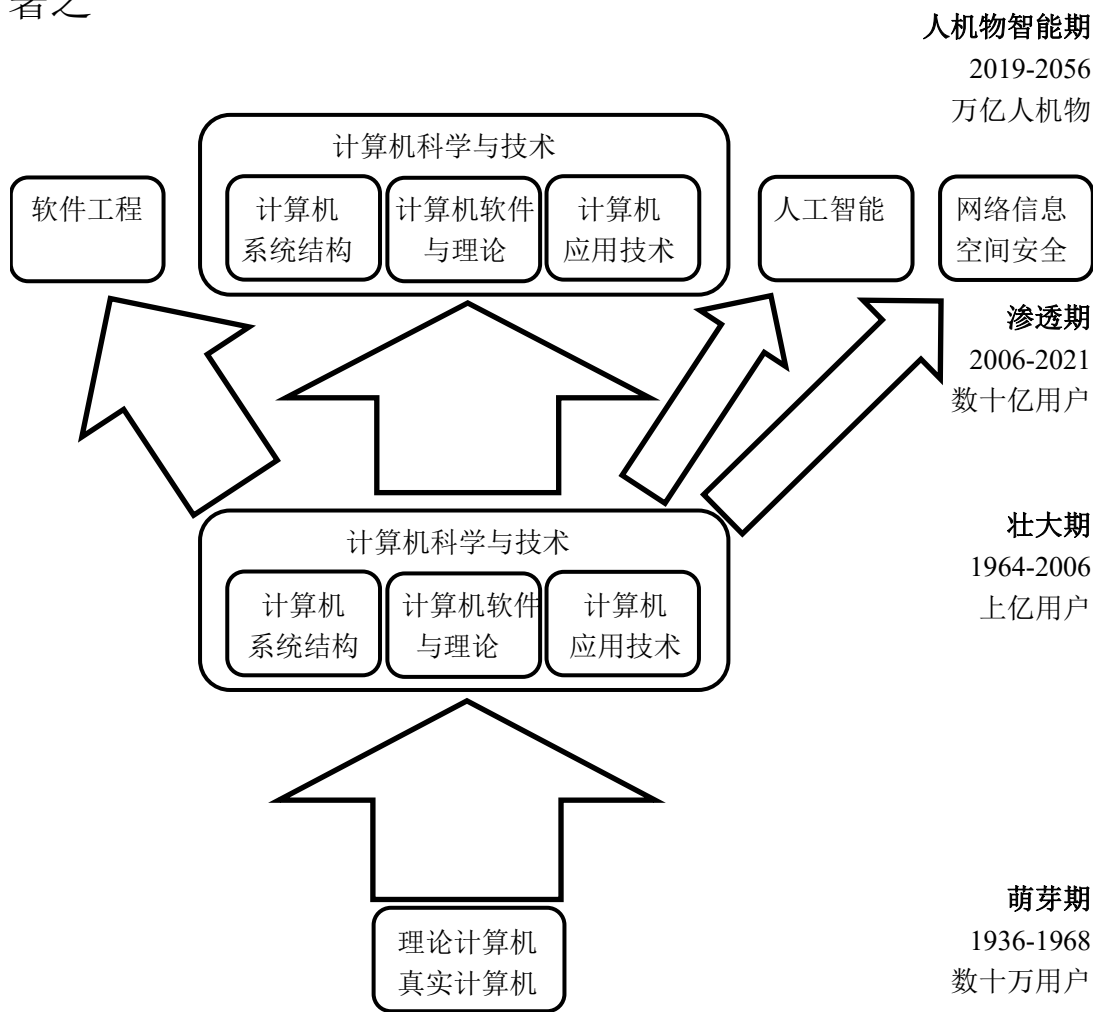


计算机学科发展演化树：萌芽期

- 1936：图灵机论文发表
- 1968：高德纳的《计算机程序设计艺术》教科书出版
 - 20世纪最有影响的12部科学专著之一

- 重要进展

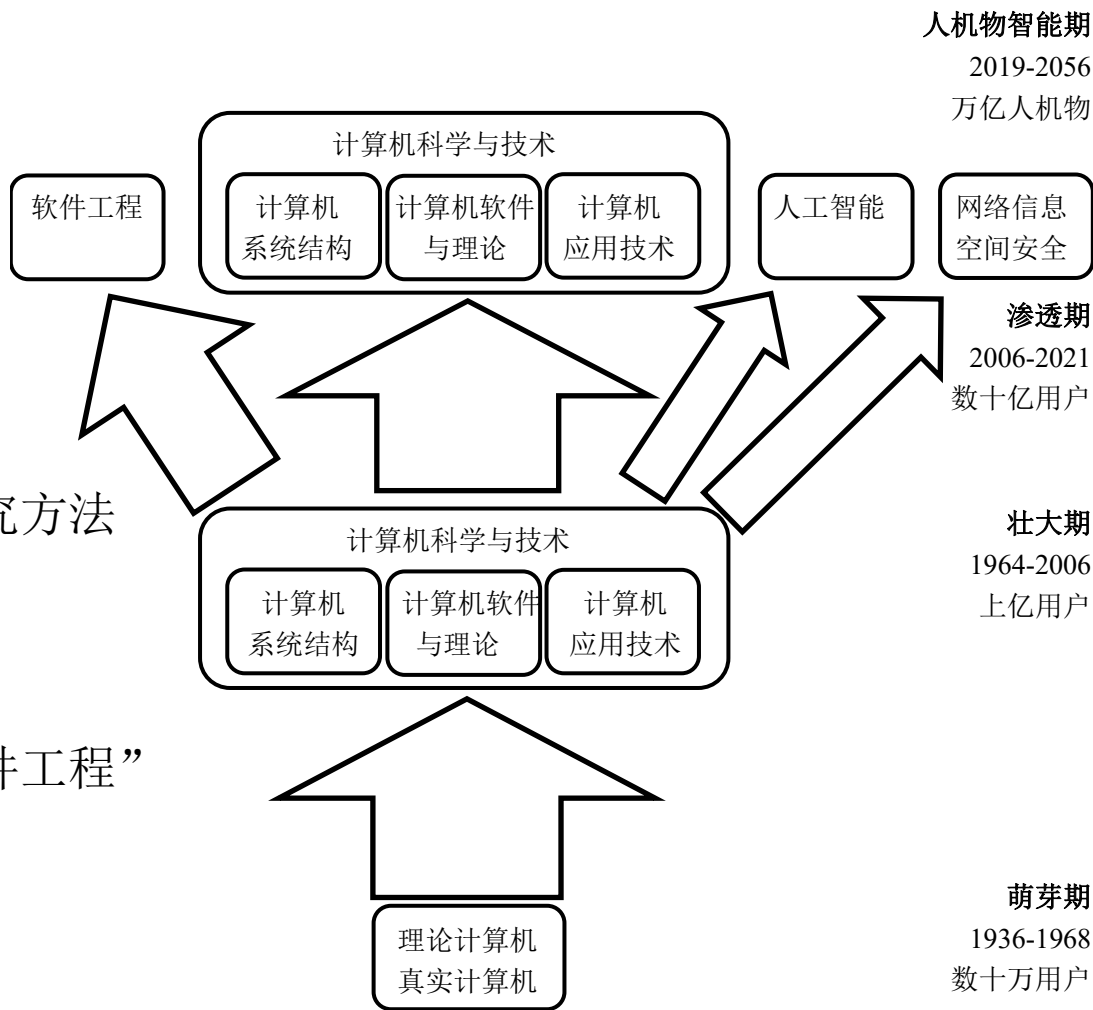
- 冯诺依曼体系结构
- 通用的真实计算机
- 操作系统
- 高级程序设计语言
- 科学计算、计算机模拟仿真、计算机图形学、计算机过程控制、计算机信息管理系统等计算机应用
- 普度大学创立了全球第一个计算机专业
- ACM、IEEE-CS、中国计算机学会成立



计算机学科发展演化树：壮大期

- 1964年：IBM发布S/360通用计算机系列
- 2006年：周以真发表“计算思维”论文
- 重要进展

- 计算局部性原理
- P vs. NP, NP完备性
- 分布式计算理论
- 计算机系统结构概念
 - 真实计算机定性研究
- 计算机体系结构：量化研究方法
 - 真实计算机定量研究
- 基础软件与应用软件
- 应对“软件危机”的“软件工程”
- 计算机网络
- 万维网

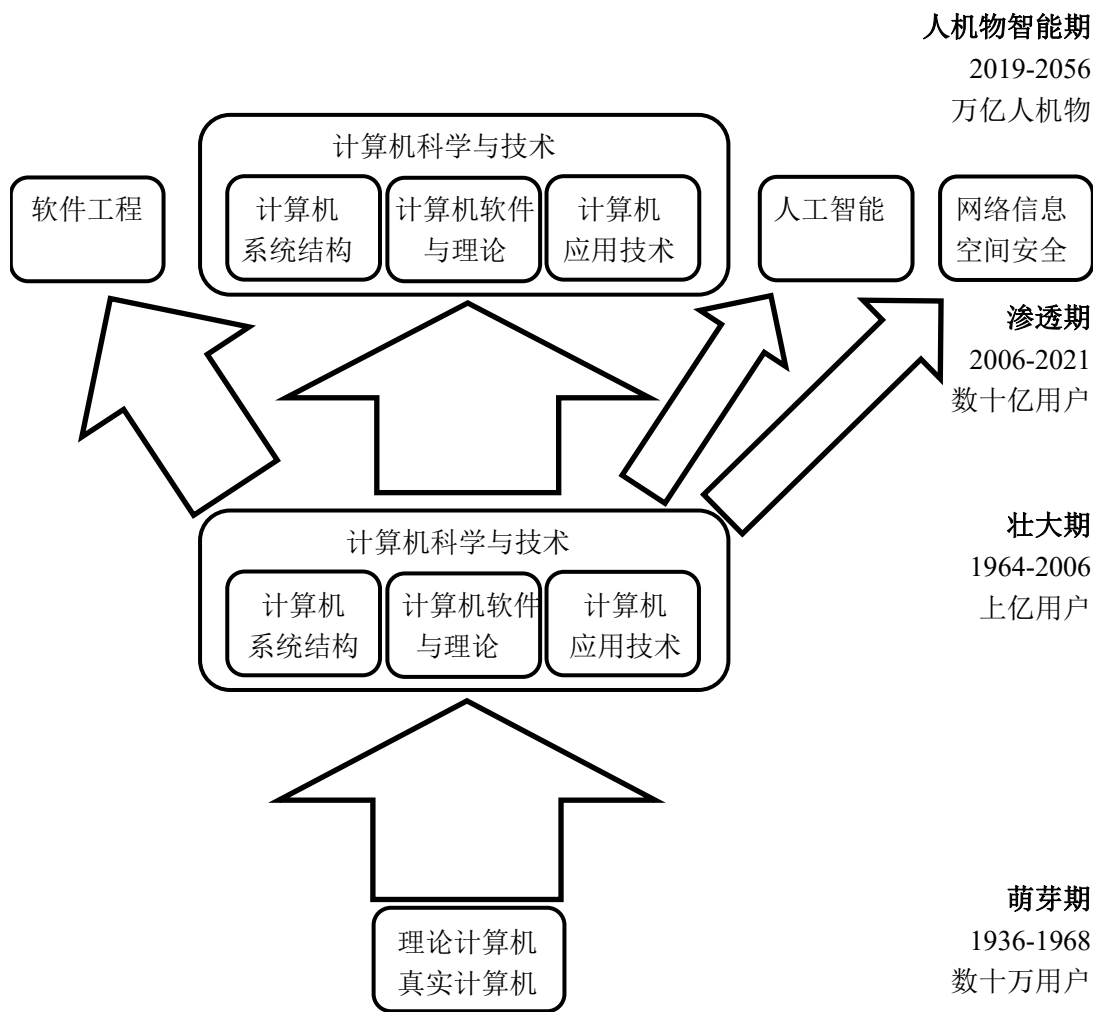


计算机学科发展演化树：渗透期

- 2006年：周以真发表“计算思维”论文
- 2021年：抖音成为全球访问量最大的网站，超过谷歌

- 重要进展

- 计算思维愿景
- 移动互联网、云计算、大数据、人工智能等计算机技术迅猛发展，得到了大规模应用
- 渗透到了发展中国家
- 互联网全球渗透率
2000: 6%
2021: 59.5%
- 抖音（TikTok）成为2021年全球访问量最大的互联网网站，超过2020年冠军谷歌

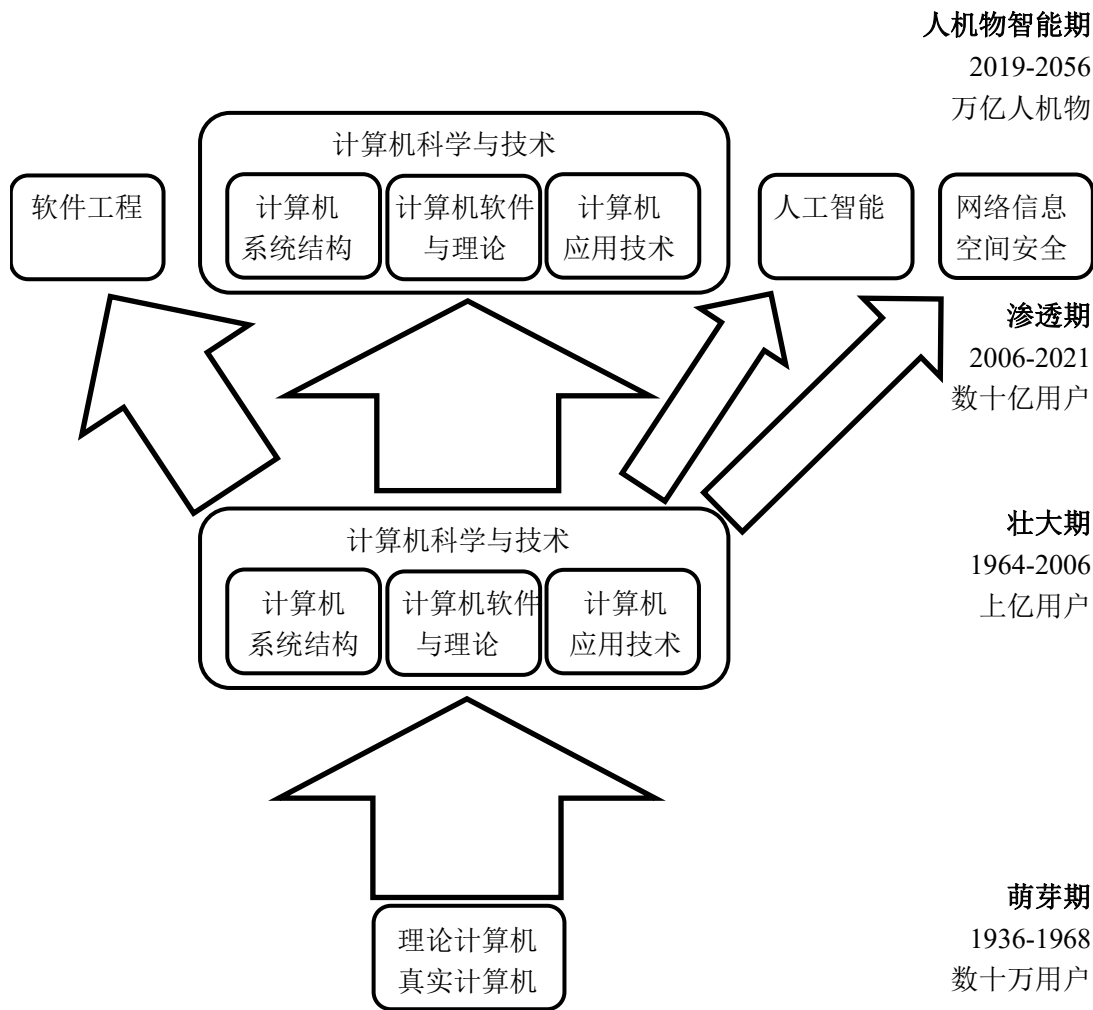


计算机学科发展演化树：人机物智能期

- 2019年：万亿设备新世界（the Trillion-Device World）
- 2056年：图灵机论文发表120周年

- 主要特征

- 计算从赛博空间拓展到人机物三元计算
- “人机物”融合的智能万物互联时代
- 计算机不只是电子计算机，它的部件还包括人和物



6. 格雷12问题，对标数学的希尔伯特23问题

2050年前做到

- ① 可扩展系统（**Scalability**）：设计出算力可扩展1百万倍的系统结构。
- ② 图灵测试（**Turing Test**）：设计出可通过图灵测试的计算机系统。
- ③ 母语听（**Speech to text**）：构建计算机系统，能够像说母语的人一样听懂人讲话。
- ④ 母语说（**Text to speech**）：构建计算机系统，能够像说母语的人一样讲话。
- ⑤ 真人看（**See as well as a person**）。构建计算机系统，能够像人一样识别事物行为。
- ⑥ 个人数字资产库系统（**Personal Memex**）。记录个人一生所读、所看、所听，并能快速检索，但不做任何分析。消费者个人能够负担其购买成本与使用成本。
- ⑦ 全球数字资产库系统（**World Memex**）。所有文本、声音、图像、视频上网；专家级的分析和摘要能力；快速检索能力。
- ⑧ 远程呈现（**TelePresence**）。模拟人出现在另一个地方，能与当地环境和其他人交互，好像真实出现一样。
- ⑨ 无故障系统（**Trouble-Free Systems**）。构建计算机系统，能够供百万人日常使用，但只需一个兼职人员管理维护。
- ⑩ 安全系统（**Secure System**）。保障上述无故障系统只对授权用户开放、非法用户不能阻碍合法用户使用、信息不可能被窃取。证明这三种安全性。
- ⑪ 系统可用性（**AlwaysUp**）。保障上述无故障系统的可用性高达99.9999999%（9个9），即每100年才出错1秒。证明这种可用性。
- ⑫ 自动程序员（**Automatic Programmer**）。设计出一种规约语言或用户界面，具备5个性质：
（1）通用，能够表达任意应用的设计；（2）高效，表达效率提升一千倍；（3）自动，计算机能够自动编译该设计表达；（4）易用，系统较为易用；（5）智能，可针对应用设计中存在的异常与缺失自动做推理、询问用户。

格雷12问题：性能

- ① **可扩展系统（Scalability）**：设计出算力可扩展**1百万倍**的系统结构。
- ② 图灵测试（Turing Test）：设计出可通过图灵测试的计算机系统。
- ③ 母语听（Speech to text）：构建计算机系统，能够像说母语的人一样听懂人讲话。
- ④ 母语说（Text to speech）：构建计算机系统，能够像说母语的人一样讲话。
- ⑤ 真人看（See as well as a person）。构建计算机系统，能够像人一样识别事物行为。
- ⑥ 个人数字资产库系统（Personal Memex）。记录个人一生所读、所看、所听，并能快速检索，但不做任何分析。消费者个人能够负担其购买成本与使用成本。
- ⑦ 全球数字资产库系统（World Memex）。所有文本、声音、图像、视频上网；专家级的分析和摘要能力；快速检索能力。
- ⑧ 远程呈现（TelePresence）。模拟人出现在另一个地方，能与当地环境和其他人交互，好像真实出现一样。
- ⑨ 无故障系统（Trouble-Free Systems）。构建计算机系统，能够供百万人日常使用，但只需一个兼职人员管理维护。
- ⑩ 安全系统（Secure System）。保障上述无故障系统只对授权用户开放、非法用户不能阻碍合法用户使用、信息不可能被窃取。证明这三种安全性。
- ⑪ 系统可用性（AlwaysUp）。保障上述无故障系统的可用性高达**99.9999999%**（**9个9**），即每**100年**才出错**1秒**。证明这种可用性。
- ⑫ 自动程序员（Automatic Programmer）。设计出一种规约语言或用户界面，具备**5个**性质：
（1）通用，能够表达任意应用的设计；（2）高效，表达效率提升一千倍；（3）自动，计算机能够自动编译该设计表达；（4）易用，系统较为易用；（5）智能，可针对应用设计中存在的异常与缺失自动做推理、询问用户。

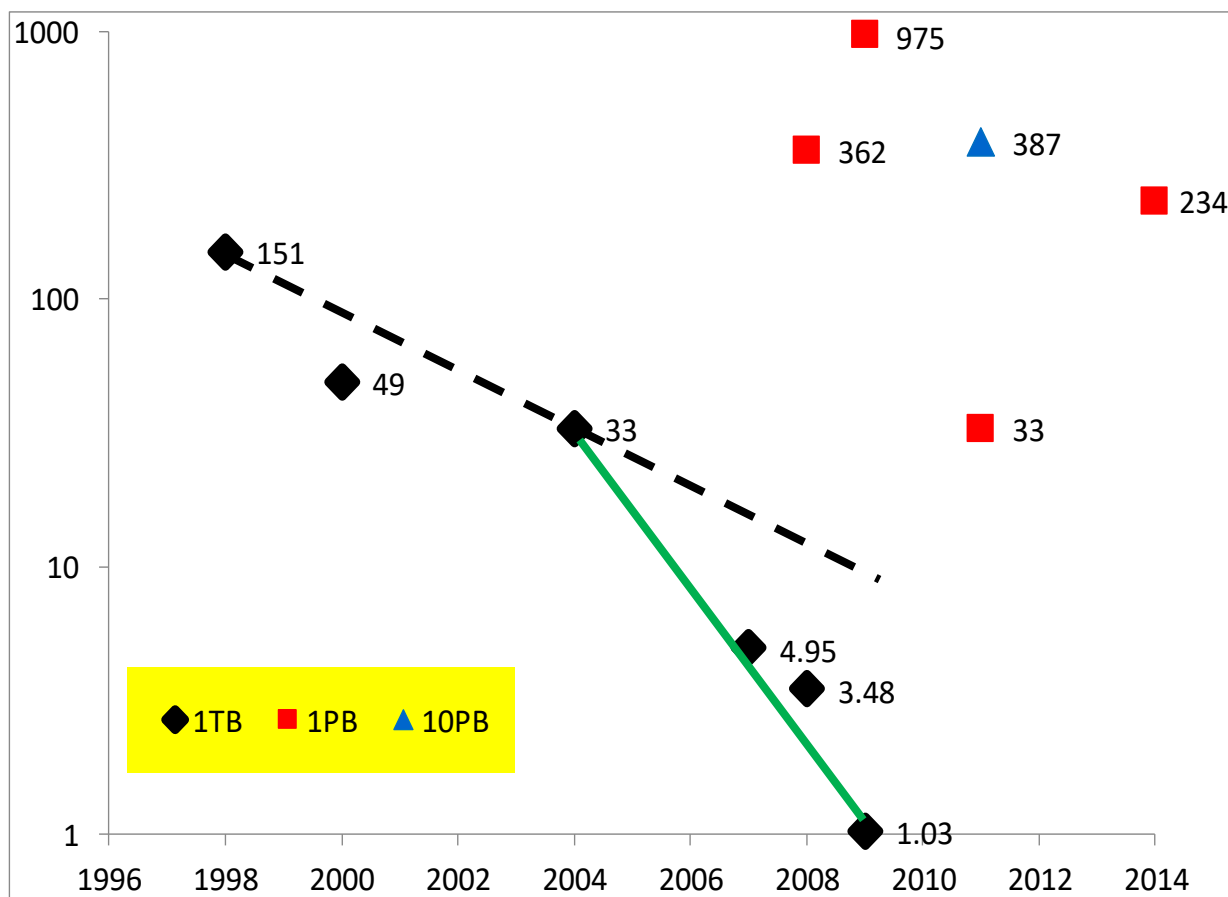
科学计算系统（超级计算机）已可扩展**10万倍**

- Linpack基准程序取得的实际计算速度最快，计算速度的单位是每秒执行的64比特浮点运算次数
- Fugaku采用了由多核并行计算节点连接而成的机群体系结构，整机系统结构可扩展，最多可添加**15万个**计算节点，将计算速度提升到单节点计算速度的**12万倍**

发布时间	1993年	2020年	1993-2020年增长倍数
冠军名称	Thinking Machine CM-5	Fujitsu Fugaku	N/A
问题规模	N = 52,224	N = 20,459,520	392
计算速度	59.7 GFlop/s	415,530 TFlop/s	6,960,302
主频	32 MHz	2.2 GHz	69
并发度	1,024 cores	7,299,072 cores	7,128
内存容量	32 GB	4,866,048 GB	152,064
功耗	96.5 KW	28,334.5 KW	294
成本	US \$30 million	US \$1 billion	33

大数据计算系统已可扩展上千倍

- 使用TeraSort基准程序，衡量进步的标准是排序1 TB需要多少时间
 - 前期进展很慢，采用“**机群+大数据应用框架**”后进展迅速
 - 腾讯系统拥有当前世界纪录，每分钟排序60.7 TB

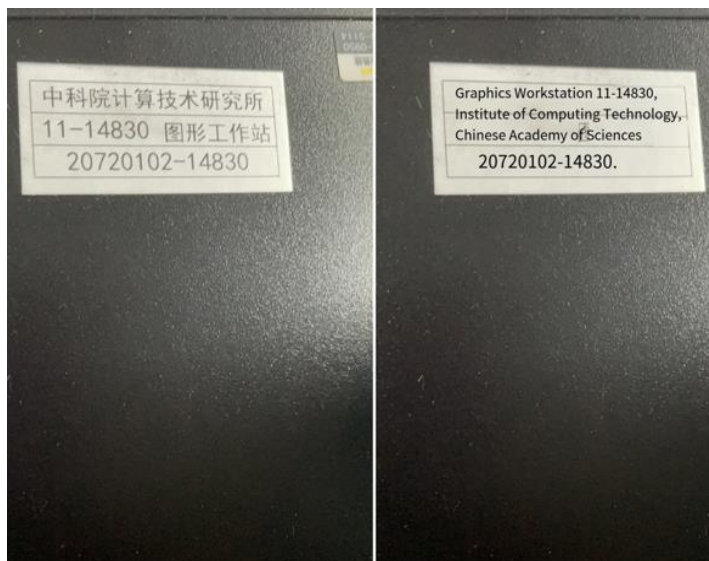


格雷12问题：智能

- ① 可扩展系统（**Scalability**）：设计出算力可扩展1百万倍的系统结构。
- ② 图灵测试（**Turing Test**）：设计出可通过图灵测试的计算机系统。
- ③ 母语听（**Speech to text**）：构建计算机系统，能够像说母语的人一样听懂人讲话。
- ④ 母语说（**Text to speech**）：构建计算机系统，能够像说母语的人一样讲话。
- ⑤ 真人看（**See as well as a person**）。构建计算机系统，能够像人一样识别事物行为。
- ⑥ 个人数字资产库系统（**Personal Memex**）。记录个人一生所读、所看、所听，并能快速检索，但不做任何分析。消费者个人能够负担其购买成本与使用成本。
- ⑦ 全球数字资产库系统（**World Memex**）。所有文本、声音、图像、视频上网；专家级的分析和摘要能力；快速检索能力。
- ⑧ 远程呈现（**TelePresence**）。模拟人出现在另一个地方，能与当地环境和其他人交互，好像真实出现一样。
- ⑨ 无故障系统（**Trouble-Free Systems**）。构建计算机系统，能够供百万人日常使用，但只需一个兼职人员管理维护。
- ⑩ 安全系统（**Secure System**）。保障上述无故障系统只对授权用户开放、非法用户不能阻碍合法用户使用、信息不可能被窃取。证明这三种安全性。
- ⑪ 系统可用性（**AlwaysUp**）。保障上述无故障系统的可用性高达99.9999999%（9个9），即每100年才出错1秒。证明这种可用性。
- ⑫ 自动程序员（**Automatic Programmer**）。设计出一种规约语言或用户界面，具备5个性质：
（1）通用，能够表达任意应用的设计；（2）高效，表达效率提升一千倍；（3）自动，计算机能够自动编译该设计表达；（4）易用，系统较为易用；（5）智能，可针对应用设计中存在的异常与缺失自动做推理、询问用户。

智能应用已有明显进展

- 但离格雷的目标还差得远
- 进步主要得益于深度学习技术的深入发展和应用
 - 算法 + 算力 + 数据



网易拍照翻译（左）、讯飞翻译机（中）与百度智能音箱（右）

格雷12问题：品质

- ① 可扩展系统（Scalability）：设计出算力可扩展1百万倍的系统结构。
- ② 图灵测试（Turing Test）：设计出可通过图灵测试的计算机系统。
- ③ 母语听（Speech to text）：构建计算机系统，能够像说母语的人一样听懂人讲话。
- ④ 母语说（Text to speech）：构建计算机系统，能够像说母语的人一样讲话。
- ⑤ 真人看（See as well as a person）。构建计算机系统，能够像人一样识别事物行为。
- ⑥ 个人数字资产库系统（Personal Memex）。记录个人一生所读、所看、所听，并能快速检索，但不做任何分析。消费者个人能够负担其购买成本与使用成本。
- ⑦ 全球数字资产库系统（World Memex）。所有文本、声音、图像、视频上网；专家级的分析和摘要能力；快速检索能力。
- ⑧ 远程呈现（TelePresence）。模拟人出现在另一个地方，能与当地环境和其他人交互，好像真实出现一样。
- ⑨ 无故障系统（Trouble-Free Systems）。构建计算机系统，能够供百万人日常使用，但只需一个兼职人员管理维护。
- ⑩ 安全系统（Secure System）。保障上述无故障系统只对授权用户开放、非法用户不能阻碍合法用户使用、信息不可能被窃取。证明这三种安全性。
- ⑪ 系统可用性（AlwaysUp）。保障上述无故障系统的可用性高达99.9999999%（9个9），即每100年才出错1秒。证明这种可用性。
- ⑫ 自动程序员（Automatic Programmer）。设计出一种规约语言或用户界面，具备5个性质：
（1）通用，能够表达任意应用的设计；（2）高效，表达效率提升一千倍；（3）自动，计算机能够自动编译该设计表达；（4）易用，系统较为易用；（5）智能，可针对应用设计中存在的异常与缺失自动做推理、询问用户。

云账户互联网服务系统

- 为6600万自由职业者提供秒批办照、收入结算、税款代缴、保险保障服务
 - 2016年成立，2021年实现收入500多亿元、纳税30多亿元；
 - 云账户董事长获“全国脱贫攻坚先进个人”荣誉
 - 仅有200多名技术人员
- 得益于分布式系统进展，使用了30余种开源应用框架
- 格雷问题进步
 - 第9问题：无故障系统，供百万人日常使用，只需一个兼职人员管理维护
 - 云账户系统支持6600多万活跃用户，需要三十余名全职工程师做运行维护



云账户互联网服务系统

- 为6600万自由职业者提供秒批办照、收入结算、税款代缴、保险保障服务
 - 2016年成立，2021年实现收入500多亿元、纳税30多亿元；
 - 云账户董事长获“全国脱贫攻坚先进个人”荣誉
 - 仅有200多名技术人员
- 得益于分布式系统进展，使用了30余种开源应用框架



- 格雷问题进步
 - 第10问题：安全系统。保障上述无故障系统只对授权用户开放、非法用户不能阻碍合法用户使用、信息不可能被窃取。证明这三种安全性。
 - 云账户系统实际运营体现了上述三种安全性，但不能数学严密地证明

云账户互联网服务系统

- 为6600万自由职业者提供秒批办照、收入结算、税款代缴、保险保障服务
 - 2016年成立，2021年实现收入500多亿元、纳税30多亿元；
 - 云账户董事长获“全国脱贫攻坚先进个人”荣誉
 - 仅有200多名技术人员
- 得益于分布式系统进展，使用了30余种开源应用框架



- 格雷问题进步
 - 第11问题：可用性高达99.9999999%（9个9），每100年才出错1秒。证明这种可用性。
 - 云账户系统已经在五年多时间内无中断地为用户服务
 - 可用性还不能证明

格雷12问题：易用

- ① 可扩展系统（Scalability）：设计出算力可扩展1百万倍的系统结构。
- ② 图灵测试（Turing Test）：设计出可通过图灵测试的计算机系统。
- ③ 母语听（Speech to text）：构建计算机系统，能够像说母语的人一样听懂人讲话。
- ④ 母语说（Text to speech）：构建计算机系统，能够像说母语的人一样讲话。
- ⑤ 真人看（See as well as a person）。构建计算机系统，能够像人一样识别事物行为。
- ⑥ 个人数字资产库系统（Personal Memex）。记录个人一生所读、所看、所听，并能快速检索，但不做任何分析。消费者个人能够负担其购买成本与使用成本。
- ⑦ 全球数字资产库系统（World Memex）。所有文本、声音、图像、视频上网；专家级的分析和摘要能力；快速检索能力。
- ⑧ 远程呈现（TelePresence）。模拟人出现在另一个地方，能与当地环境和其他人交互，好像真实出现一样。
- ⑨ 无故障系统（Trouble-Free Systems）。构建计算机系统，能够供百万人日常使用，但只需一个兼职人员管理维护。
- ⑩ 安全系统（Secure System）。保障上述无故障系统只对授权用户开放、非法用户不能阻碍合法用户使用、信息不可能被窃取。证明这三种安全性。
- ⑪ 系统可用性（AlwaysUp）。保障上述无故障系统的可用性高达99.9999999%（9个9），即每100年才出错1秒。证明这种可用性。
- ⑫ **自动程序员（Automatic Programmer）。设计出一种规约语言或用户界面，具备5个性质：（1）通用，能够表达任意应用的设计；（2）高效，表达效率提升一千倍；（3）自动，计算机能够自动编译该设计表达；（4）易用，系统较为易用；（5）智能，可针对应用设计中存在的异常与缺失自动做推理、询问用户。**

云账户互联网服务系统

- 为6600万自由职业者提供秒批办照、收入结算、税款代缴、保险保障服务
 - 2016年成立，2021年实现收入500多亿元、纳税30多亿元；
 - 云账户董事长获“全国脱贫攻坚先进个人”荣誉
 - 仅有200多名技术人员
- 得益于分布式系统进展，使用了30余种开源应用框架



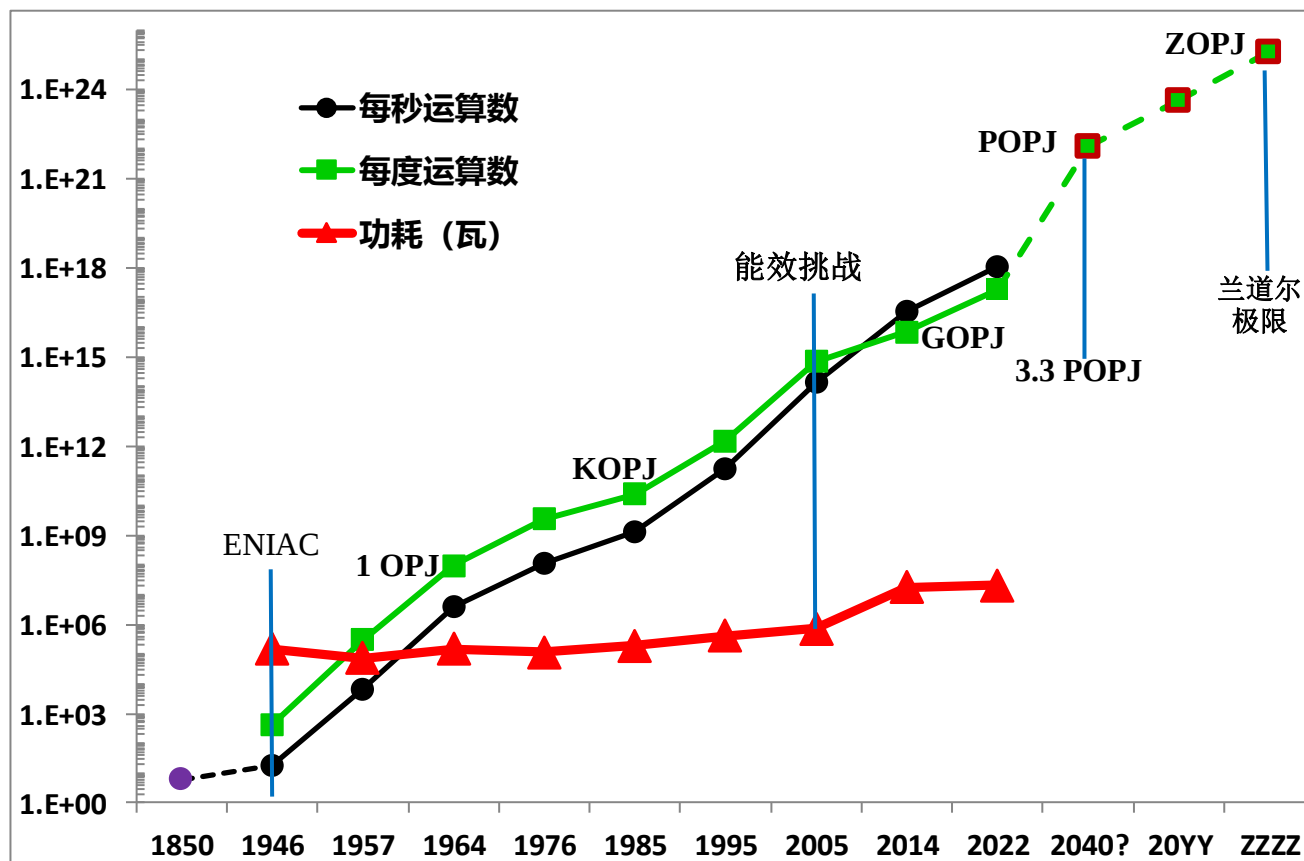
- 格雷问题进步
 - 第12问题：自动程序员。设计出一种规约语言或用户界面，使得表达效率提升一千倍。
 - 云账户仍然主要依赖于产品经理和工程师的合作来设计、开发和测试系统
 - 虽然利用基础组件库、微服务、开发工具链、敏捷开发流程等技术可减少重复工作，但云账户团队主要基于高级编程语言开发分布式应用系统
 - 只在几个限定的应用场景中，才支持最终用户在可视化环境中拖拽完成系统设计
 - 尚缺乏可自动执行的系统抽象

解决格雷问题受制于能效挑战

- OPJ: 每焦耳运算数
- 物理学的兰道尔极限前的三个里程碑
 - ENIAC: 终结了95年缓慢进展, 开启了60年的指数改善
 - 2005年开始, 能效挑战变得严峻
 - DARPA提出3.3 POPJ的目标, 试图大幅提升能效

Z: Zetta, 泽, 10^{21}
P: Peta, 拍, 10^{15}
G: Giga, 吉, 10^9
K: Kilo, 千, 10^3

单套计算机系统的
速度、能效、
功耗发展历史与
展望



7. 计算系统思维

- 通过**抽象**，将**模块**组合成为系统，**无缝**执行计算过程
 - 抽象化：一个通用抽象代表多个具体需求
 - 模块化：系统由多个模块组合而成
 - 计算机 = 硬件 + 系统软件 + 应用软件
 - 全系统一致性；信息隐藏原理，接口概念
 - 无缝衔接
 - 避免缝隙：
 - 扬雄周期原理、波斯特尔鲁棒性原理、冯诺依曼穷举原理
 - 重视瓶颈：阿姆达尔定律
 - 系统性能改进受限于系统瓶颈（针对某任务）
 - $\text{加速比} = 1 / ((1-f)/p + f) \rightarrow 1/f$ 当 $p \rightarrow \infty$

以一耦万

- 向编程者提供抽象，比特精准地将抽象映射到硬件

万千应用

程序
进程
指令
冯氏结构
指令流水线
时序电路
组合电路

本课程学到的一些基本抽象

- 是什么、通过实例解释、能够举一反三

Data Type 数据类型	bit (1 bit), hexadecimal number (4 bits), byte (8 bits), uint8 (8-bit unsigned integer), integer (64 bits); array (n elements of the same type), slice (a descriptor pointing to an array); text file, BMP image file; hypertext and hyperlink 比特、字节、整数、数组、切片、文本文件、图像文件、超链接	
Software 软件	Algorithm 算法	Smart method of information transformation, such as quicksort, hiding text in a BMP file, etc. 快排算法
	Program 程序	Code realizing algorithms in computer language, such as hide.go in the Text Hider project 信息隐藏程序 hide.go
	Process 进程	Program in execution, such as the “hide” process running in a Linux environment > ./WebServer & 系统显示 PID=79
	Instruction 指令	The smallest unit of software, directly executable by computer hardware 班级快排指令集
von Neumann Architecture: a computer model bridging software and hardware 冯诺依曼计算机模型		
Hardware 硬件	Instruction Pipeline 指令流水线	The basic hardware mechanism to automatically execute any instruction “取指-译码-执行” 三级流水线
	Sequential Circuit 时序电路	More precisely, only consider Synchronous Sequential Circuit comprised of combinational circuits and state circuits and driven by a clock signal; equivalent to the automata concept 串行加法器
	Combinational Circuit 组合电路	Also known as Boolean circuit, realizing a Boolean function 使用全加器的波纹进位加法器

8. 信息隐藏程序回顾

- Write a program `hide-0.go`
 - to hide the text of Shakespeare's *Hamlet*
 - in a picture file `Autumn.bmp`
 - such that the doctored file shows no visible difference from the original picture
- and another program `show-0.go` to recover the text

HAMLET

DRAMATIS PERSONAE

CLAUDIUS king of Denmark. (KING CLAUDIUS:)

HAMLET son to the late, and nephew to the present king.

.....

ACT ISCENE I Elsinore. A platform before the castle.

.....

PRINCE FORTINBRAS Let four captains
Bear Hamlet, like a soldier, to the stage;
For he was likely, had he been put on,
To have proved most royally: and, for his passage,
The soldiers' music and the rites of war
Speak loudly for him.
Take up the bodies: such a sight as this
Becomes the field, but here shows much amiss.
Go, bid the soldiers shoot.
[A dead march. Exeunt, bearing off the dead
bodies; after which a peal of ordnance is shot off]

.....

] 换行键0x0A



Original picture

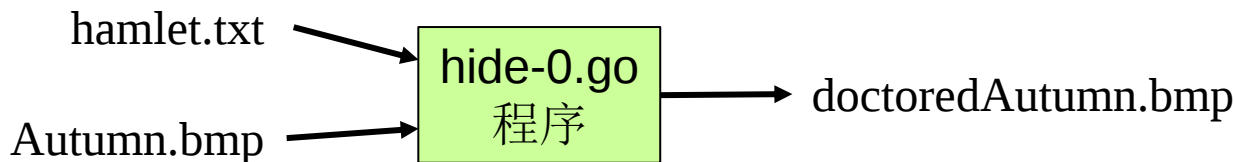


After careless hiding



After careful hiding

程序hide-0.go的设计思路：自顶向下



信息隐藏算法

- 输入：文本文件hamlet.txt 与图像文件 Autumn.bmp。
- 输出：新图像文件doctoredAutumn.bmp，它隐藏了hamlet.txt的全部内容，且与Autumn.bmp无可见差别。
- 计算过程的步骤：
 1. 将文本文件hamlet.txt读进变量t // t 代表文本（text）
 2. 将图像文件Autumn.bmp读进变量p // p 代表图像（picture）
 3. 将hamlet.txt的文件长度隐藏到变量p中Pixel Array的头32个字节
 4. 将hamlet.txt的文件内容隐藏到变量p中Pixel Array的剩余字节
 5. 将p写到新文件doctoredAutumn.bmp

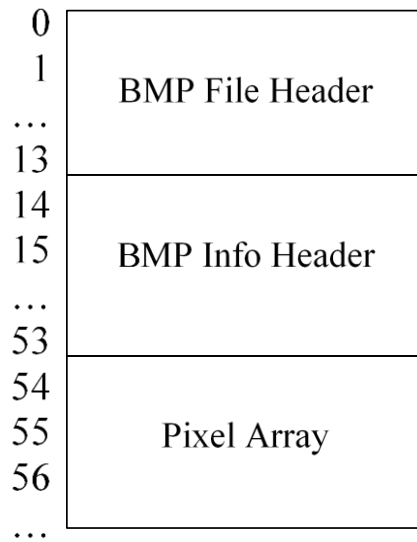
读文件
读文件
隐藏文件长度
隐藏文件内容
写文件

要点1：将文件以字节切片的方式读进内存，以便定位和操作

要点2：使用“基址+索引+偏移量”寻址模式，便于循环

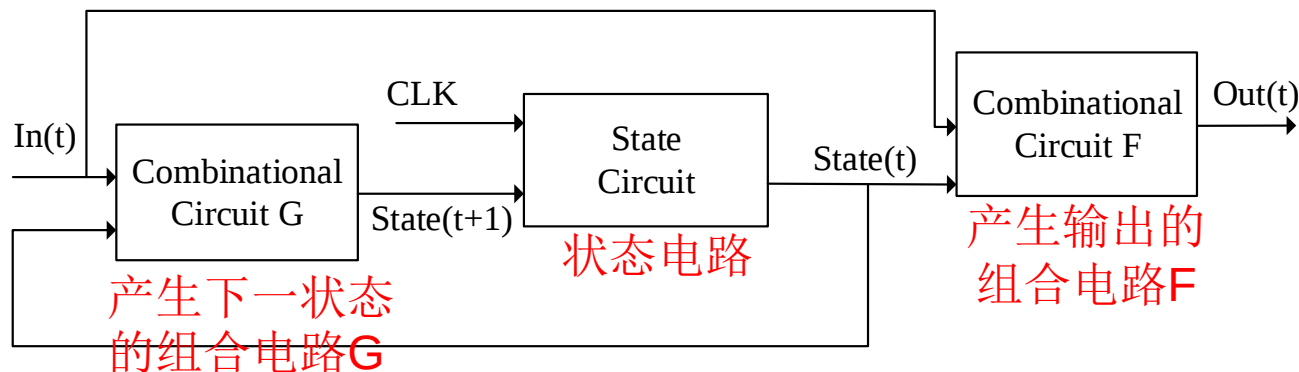


Autumn.bmp



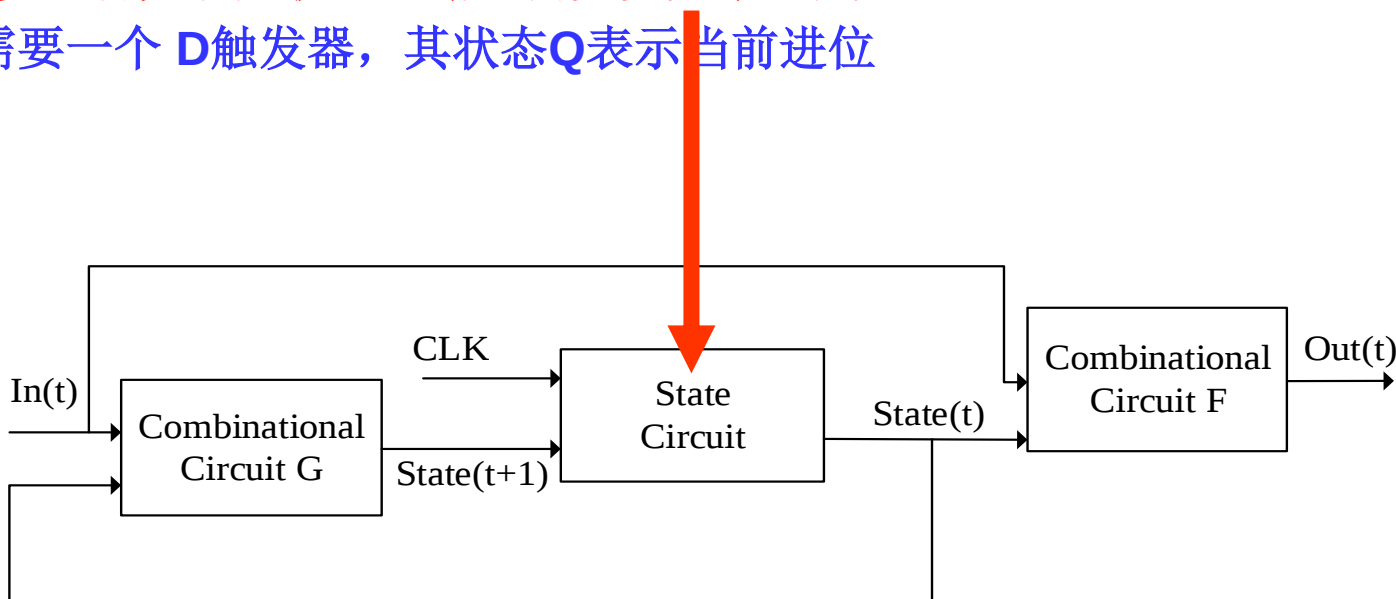
9. 时序电路的一般组成

- 两个组合电路，一个状态电路；时钟信号驱动
- 状态电路由一个或多个D触发器组成
- 两个组合电路是
 - 输出电路F: $\text{Out}(t) = F(\text{In}(t), \text{State}(t))$
 - 下一状态电路G: $\text{State}(t+1) = G(\text{In}(t), \text{State}(t))$
- 时序电路的逻辑线路图
 - 也称为时钟同步的时序电路（synchronous sequential circuit）



示例：设计4位串行加法器

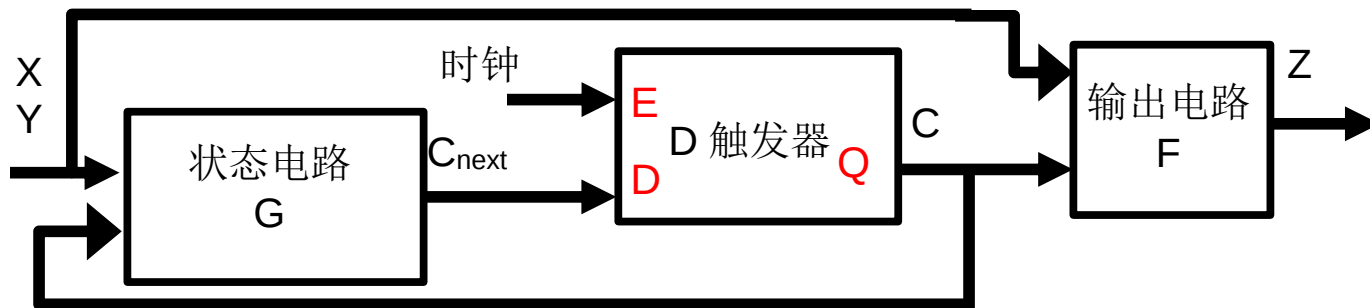
- 在四个时钟周期完成加法 $Z_3Z_2Z_1Z_0 = X_3X_2X_1X_0 + Y_3Y_2Y_1Y_0$
 - 每一周期完成一个全加操作
- 再用一个实例验证
 - $11_{10} + 9_{10} = 1011_2 + 1001_2 = 10100_2 = 20_{10} = 4_{10}$ and overflow
输入 $X=1011$, $Y=1001$; 则输出 $Z=0100$ 且产生进位溢出
- 设计过程
 - 第1步：首先确定状态电路，需要多少个D触发器？
 - 只需要一个D触发器，其状态 Q 表示当前进位



设计4位串行加法器

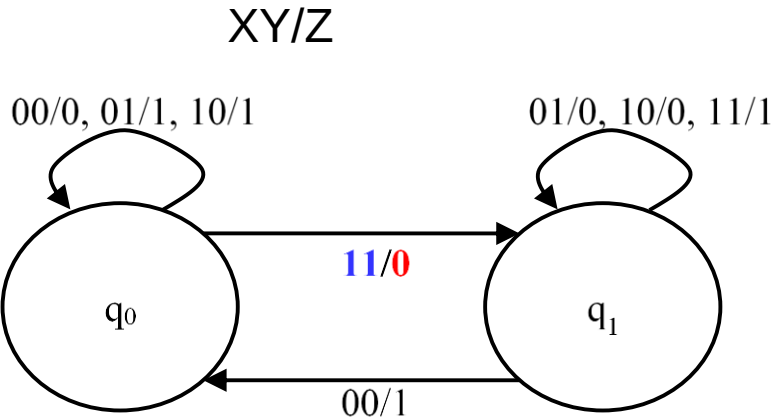
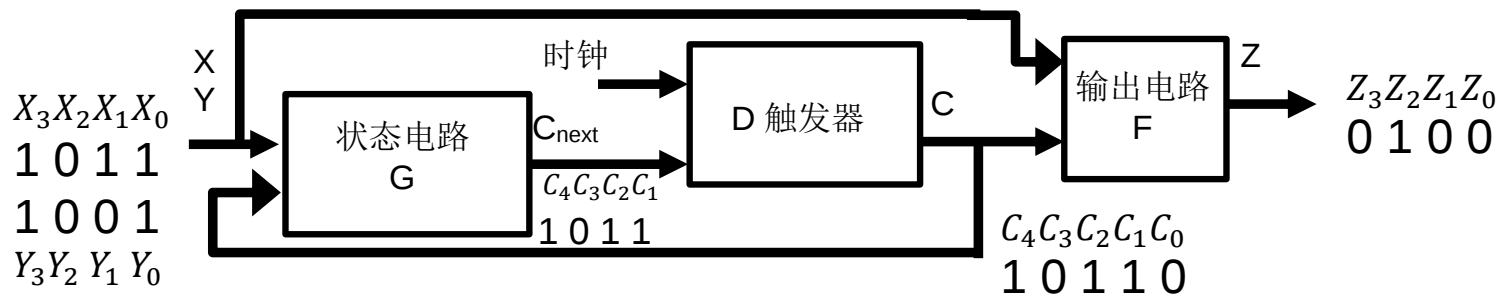
- 设计过程

- 第1步：首先确定状态电路，需要多少个D触发器？
- 只需要一个 D触发器，其状态Q表示当前进位C
- 第2步：套用时序电路一般结构
- In是输入X，Y； Out是输出Z； Q是进位C； Enable=时钟信号CLK



设计过程

- 第3步：求输出电路F与状态电路G的布尔表达式
 - 画出时序电路的状态转换图



$$\begin{array}{r} 101\textcolor{blue}{1} = X \\ 100\textcolor{blue}{1} = Y \\ + \\ \hline 101\textcolor{red}{1}0 = C \\ 010\textcolor{red}{0} = Z \end{array}$$

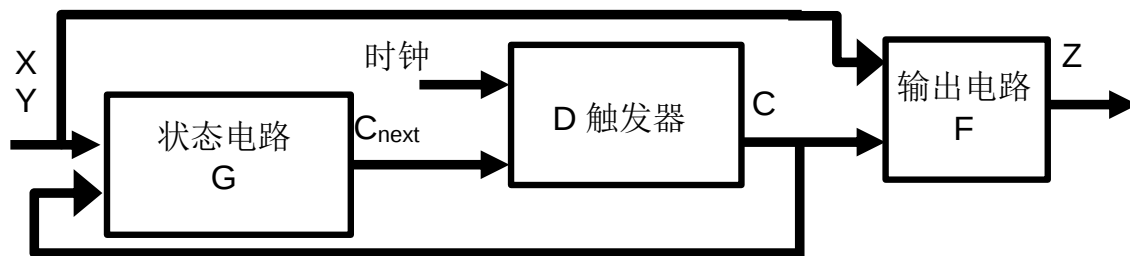
画出时序电路的状态转换图

q₀ denotes C=0
q₁ denotes C=1

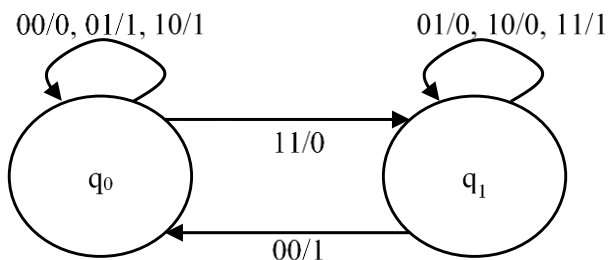
设计过程

● 第3步：求输出电路F与状态电路G的布尔表达式

- 画出时序电路的状态转换图；**进而导出真值表**



XY/Z



q_0 denotes $C=0$

q_1 denotes $C=1$

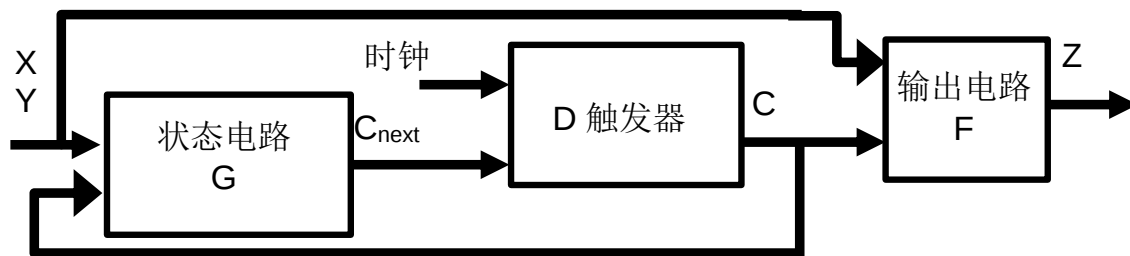
C	X	Y	Z	C_{next}
q_0	0	0	0	q_0
q_0	0	1	1	q_0
q_0	1	0	1	q_0
q_0	1	1	0	q_1
q_1	0	0	1	q_0
q_1	0	1	0	q_1
q_1	1	0	0	q_1
q_1	1	1	1	q_1

C	X	Y	Z	C_{next}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

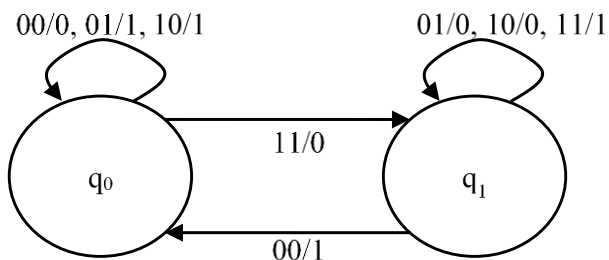
设计过程

● 第3步：求输出电路F与状态电路G的布尔表达式

- 画出时序电路的状态转换图；进而导出真值表；进而求出F和G的布尔表达式



XY/Z



q_0 denotes $C=0$
 q_1 denotes $C=1$

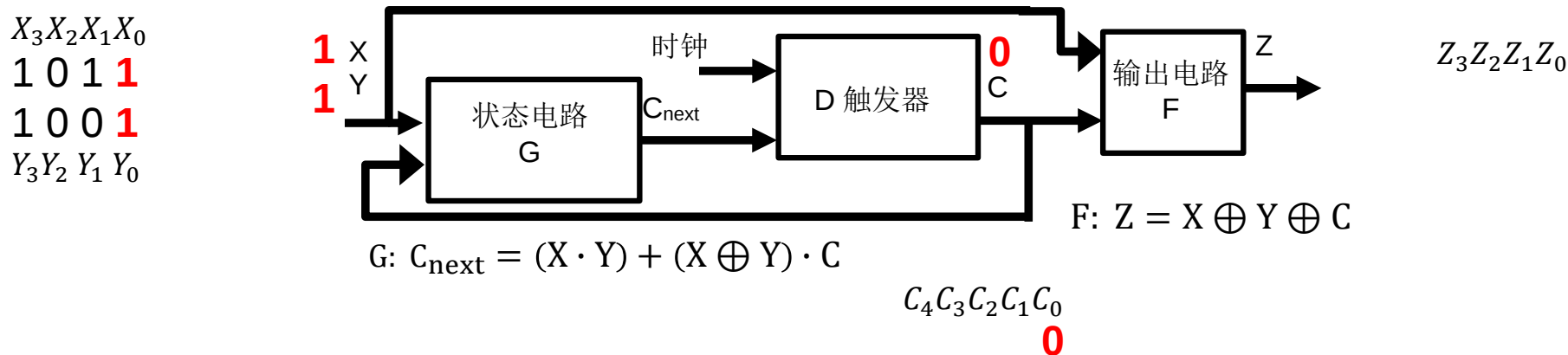
C	X	Y	Z	C_{next}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$Z = F(X, Y, C) = X \oplus Y \oplus C$$

$$C_{next} = G(X, Y, C) = (X \cdot Y) + (X \oplus Y) \cdot C$$

设计过程第4步：验算

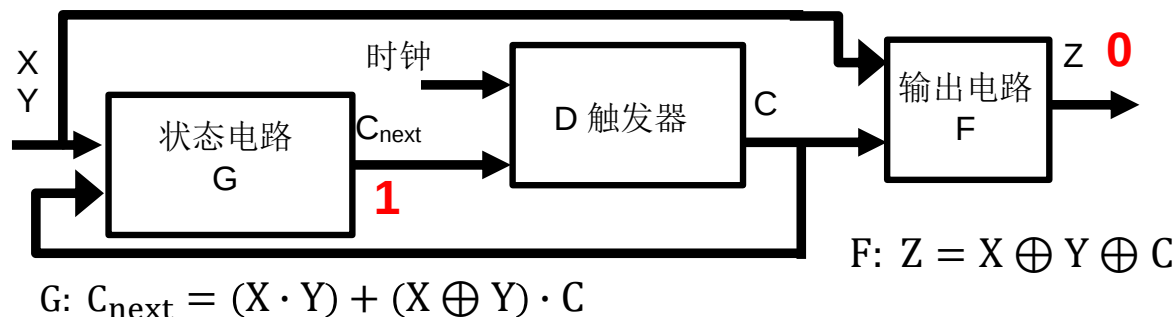
- 给定
 - (1) 设计好的下图时序电路，
 - (2) 输入 $X_3X_2X_1X_0 = 1011$, $Y_3Y_2Y_1Y_0 = 1001$, 与初始进位 $C_0 = 0$
 - 正确结果应为 $Z_3Z_2Z_1Z_0 = 0100$ ，且有 $C_4 = 1$ 表示溢出
- 验算步骤
 - 时钟周期0: $Z_0 = X_0 \oplus Y_0 \oplus C_0 = 1 \oplus 1 \oplus 0$;
 - $C_1 = (X_0 \cdot Y_0) + (X_0 \oplus Y_0) \cdot C_0 = (1 \cdot 1) + (1 \oplus 1) \cdot 0$



设计过程第4步：验算

- 给定
 - (1) 设计好的下图时序电路，
 - (2) 输入 $X_3X_2X_1X_0 = 1011$, $Y_3Y_2Y_1Y_0 = 1001$, 与初始进位 $C_0 = 0$
 - 正确结果应为 $Z_3Z_2Z_1Z_0 = 0100$, 且有 $C_4 = 1$ 表示溢出
- 验算步骤
 - 时钟周期0: $Z_0 = X_0 \oplus Y_0 \oplus C_0 = 1 \oplus 1 \oplus 0 = 0$;
 - $C_1 = (X_0 \cdot Y_0) + (X_0 \oplus Y_0) \cdot C_0 = (1 \cdot 1) + (1 \oplus 1) \cdot 0 = 1$

$X_3X_2X_1X_0$
 1 0 1 **1**
 $Y_3Y_2Y_1Y_0$
 1 0 0 **1**



$Z_3Z_2Z_1Z_0$
 0 1 0 0
0

$C_4C_3C_2C_1C_0$
 1 0

设计过程第4步：验算

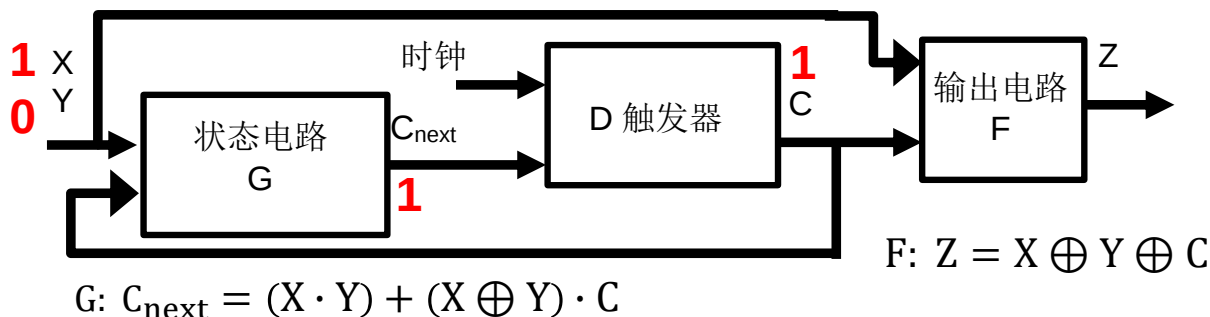
● 给定

- (1) 设计好的下图时序电路，
- (2) 输入 $X_3X_2X_1X_0 = 1011$, $Y_3Y_2Y_1Y_0 = 1001$, 与初始进位 $C_0 = 0$
- 正确结果应为 $Z_3Z_2Z_1Z_0 = 0100$, 且有 $C_4 = 1$ 表示溢出

● 验算步骤

- 时钟周期0: $Z_0 = X_0 \oplus Y_0 \oplus C_0 = 1 \oplus 1 \oplus 0 = 0$; $C_1 = (X_0 \cdot Y_0) + (X_0 \oplus Y_0) \cdot C_0 = (1 \cdot 1) + (1 \oplus 1) \cdot 0 = 1$
- 时钟周期1: $Z_1 = X_1 \oplus Y_1 \oplus C_1 = 0 \oplus 0 \oplus 1 = 1$; $C_2 = (X_1 \cdot Y_1) + (X_1 \oplus Y_1) \cdot C_1 = (0 \cdot 0) + (0 \oplus 0) \cdot 1 = 0$

$X_3X_2X_1X_0$
 1 0 1 1
 $Y_3Y_2Y_1Y_0$
 1 0 0 1



$Z_3Z_2Z_1Z_0$
 0

$C_4C_3C_2C_1C_0$
 1 0

设计过程第4步：验算

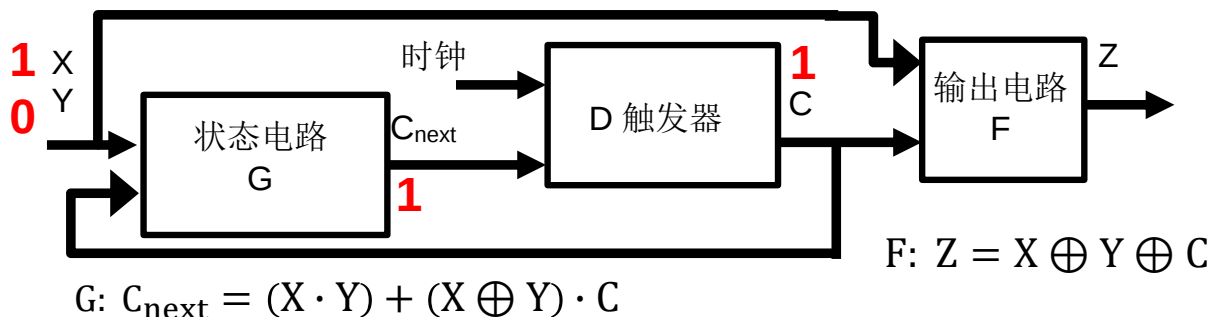
● 给定

- (1) 设计好的下图时序电路，
- (2) 输入 $X_3X_2X_1X_0 = 1011$, $Y_3Y_2Y_1Y_0 = 1001$, 与初始进位 $C_0 = 0$
- 正确结果应为 $Z_3Z_2Z_1Z_0 = 0100$, 且有 $C_4 = 1$ 表示溢出

● 验算步骤

- 时钟周期0: $Z_0 = X_0 \oplus Y_0 \oplus C_0 = 1 \oplus 1 \oplus 0 = 0$; $C_1 = (X_0 \cdot Y_0) + (X_0 \oplus Y_0) \cdot C_0 = (1 \cdot 1) + (1 \oplus 1) \cdot 0 = 1$
- 时钟周期1: $Z_1 = X_1 \oplus Y_1 \oplus C_1 = 1 \oplus 0 \oplus 1 = 0$; $C_2 = (X_1 \cdot Y_1) + (X_1 \oplus Y_1) \cdot C_1 = (1 \cdot 0) + (1 \oplus 0) \cdot 1 = 1$

$X_3X_2X_1X_0$
 1 0 1 1
 $Y_3Y_2Y_1Y_0$
 1 0 0 1



$Z_3Z_2Z_1Z_0$
 0 0

$C_4C_3C_2C_1C_0$
 1 1 0

设计过程第4步：验算

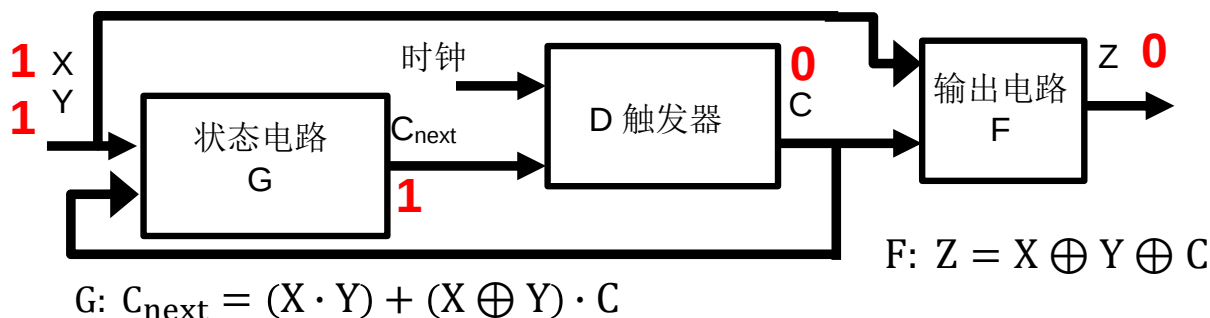
● 给定

- (1) 设计好的下图时序电路，
- (2) 输入 $X_3X_2X_1X_0 = 1011$, $Y_3Y_2Y_1Y_0 = 1001$, 与初始进位 $C_0 = 0$
- 正确结果应为 $Z_3Z_2Z_1Z_0 = 0100$ ，且有 $C_4 = 1$ 表示溢出

● 验算步骤

- 时钟周期0: $Z_0 = X_0 \oplus Y_0 \oplus C_0 = 1 \oplus 1 \oplus 0 = 0$; $C_1 = (X_0 \cdot Y_0) + (X_0 \oplus Y_0) \cdot C_0 = (1 \cdot 1) + (1 \oplus 1) \cdot 0 = 1$
- 时钟周期1: $Z_1 = X_1 \oplus Y_1 \oplus C_1 = 0 \oplus 0 \oplus 1 = 1$; $C_2 = (X_1 \cdot Y_1) + (X_1 \oplus Y_1) \cdot C_1 = (0 \cdot 0) + (0 \oplus 0) \cdot 1 = 0$
- 时钟周期2: $Z_2 = X_2 \oplus Y_2 \oplus C_2 = 1 \oplus 0 \oplus 0 = 1$; $C_3 = (X_2 \cdot Y_2) + (X_2 \oplus Y_2) \cdot C_2 = (1 \cdot 0) + (1 \oplus 0) \cdot 0 = 0$
- 时钟周期3: $Z_3 = X_3 \oplus Y_3 \oplus C_3 = 1 \oplus 1 \oplus 0 = 0$; $C_4 = (X_3 \cdot Y_3) + (X_3 \oplus Y_3) \cdot C_3 = (1 \cdot 1) + (1 \oplus 1) \cdot 0 = 1$

$X_3X_2X_1X_0$
1 0 1 1
1 0 0 1
 $Y_3Y_2Y_1Y_0$



$Z_3Z_2Z_1Z_0$
0 1 0 0

$C_4C_3C_2C_1C_0$
1 0 1 1 0

设计过程第4步：验算

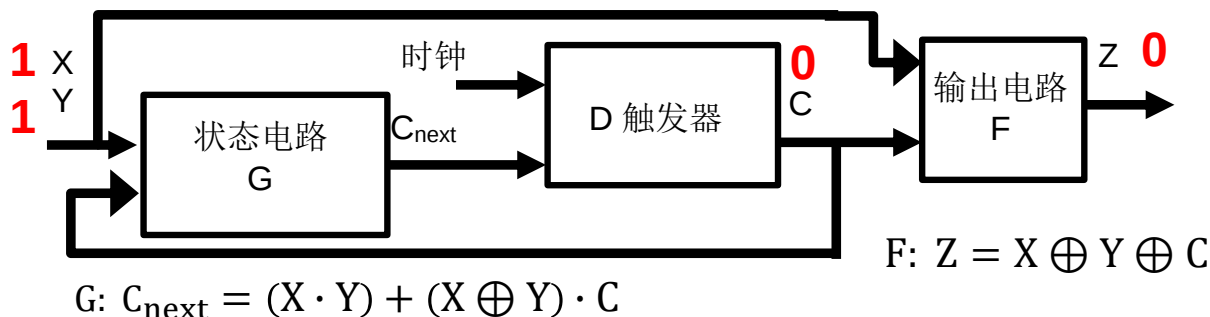
● 给定

- (1) 设计好的下图时序电路，
- (2) 输入 $X_3X_2X_1X_0 = 1011$, $Y_3Y_2Y_1Y_0 = 1001$, 与初始进位 $C_0 = 0$
- 正确结果应为 $Z_3Z_2Z_1Z_0 = 0100$ ，且有 $C_4 = 1$ 表示溢出

● 验算步骤

- 时钟周期0: $Z_0 = X_0 \oplus Y_0 \oplus C_0 = 1 \oplus 1 \oplus 0 = 0$; $C_1 = (X_0 \cdot Y_0) + (X_0 \oplus Y_0) \cdot C_0 = (1 \cdot 1) + (1 \oplus 1) \cdot 0 = 1$
- 时钟周期1: $Z_1 = X_1 \oplus Y_1 \oplus C_1 = 0 \oplus 0 \oplus 1 = 1$; $C_2 = (X_1 \cdot Y_1) + (X_1 \oplus Y_1) \cdot C_1 = (0 \cdot 0) + (0 \oplus 0) \cdot 1 = 0$
- 时钟周期2: $Z_2 = X_2 \oplus Y_2 \oplus C_2 = 1 \oplus 0 \oplus 0 = 1$; $C_3 = (X_2 \cdot Y_2) + (X_2 \oplus Y_2) \cdot C_2 = (1 \cdot 0) + (1 \oplus 0) \cdot 0 = 0$
- 时钟周期3: $Z_3 = X_3 \oplus Y_3 \oplus C_3 = 1 \oplus 1 \oplus 0 = 0$; $C_4 = (X_3 \cdot Y_3) + (X_3 \oplus Y_3) \cdot C_3 = (1 \cdot 1) + (1 \oplus 1) \cdot 0 = 1$

$X_3X_2X_1X_0$
 1 0 1 1
 $Y_3Y_2Y_1Y_0$
 1 0 0 1



$Z_3Z_2Z_1Z_0$
 0 1 0 0

$C_4C_3C_2C_1C_0$
 1 0 1 1 0

举一反三：用时序电路实现4位串行减法器

- 在四个时钟周期完成减法 $Z_3Z_2Z_1Z_0 = X_3X_2X_1X_0 - Y_3Y_2Y_1Y_0$
 - 假设数据表示是二进制补码
 - 四个时钟周期后得到正确结果
- 再用一个实例验证
 - $(-5) - 1 = -6$
- 课上已经讨论过“加减器”

加减器

代表了设计方法4：使用多路复用器和控制信号（本例中的选择输入S）

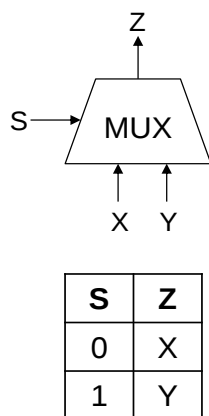
- 采用多路复用器（multiplexer, MUX）的加减器

- 控制信号S选择做加法(S=0)还是做减法(S=1)
- 采用二进制补码，减法等价于加负数， $5-5 = 5+(-5)$
 - 因为-X刚好是X的二进制补码
- 假设 $X = Y = 5$ 。即， $X_3X_2X_1X_0 = Y_3Y_2Y_1Y_0 = 0101$
 - Then, $X - Y = 5 + (-5) = 5 + 5$ 的补码

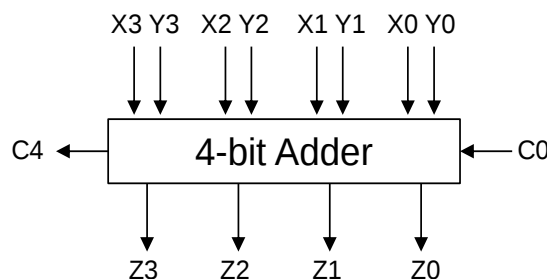
$$Z = \begin{cases} X & \text{if } S = 0 \\ Y & \text{if } S = 1 \end{cases}$$

$$\begin{aligned} &\rightarrow 0101 + (\overline{0101} + 0001) \\ &= 0101 + 1011 \\ &= 10000 = C_4 Z_3 Z_2 Z_1 Z_0 \end{aligned}$$

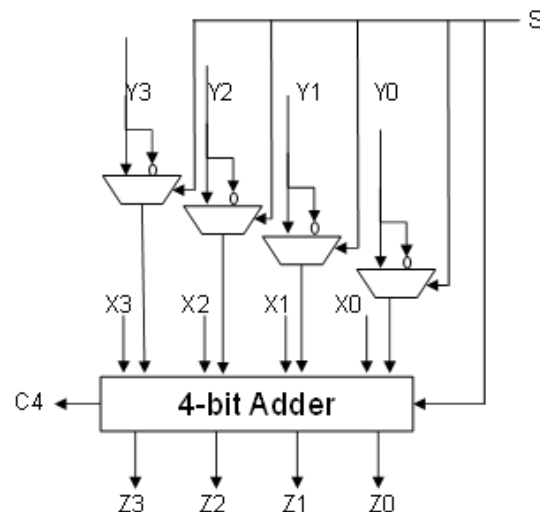
S	X	Y	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



A multiplexer 多路复用器



A two's complement 4-bit adder



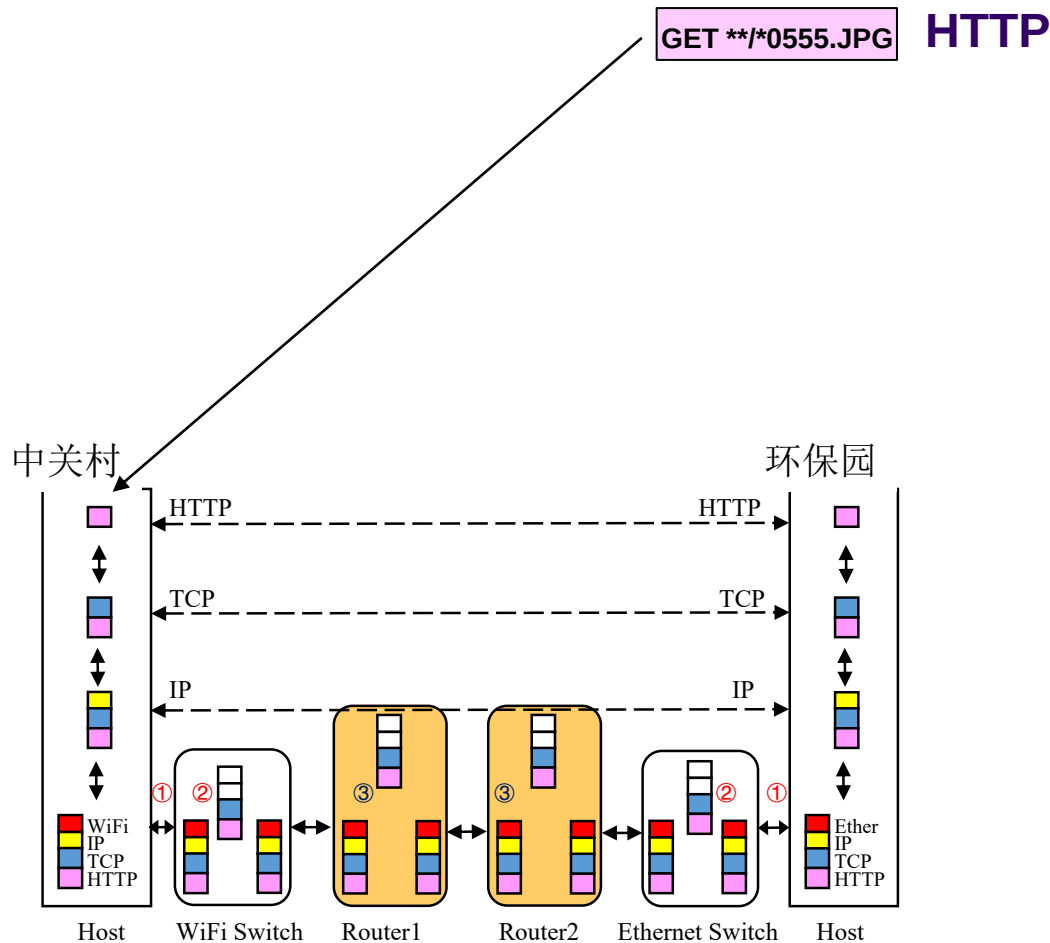
and an adder-subtractor

10. 互联网通信示例



- 从中关村计算所的笔记本电脑通过WiFi访问位于环保园服务器的“猫猫指挥家”图片；
- 笔记本电脑IP为192.168.1.101，服务器IP为10.208.104.3，图片地址为：
http://seafile.solid.things.ac.cn:7019/kitty_band/*0555.JPG

请求消息

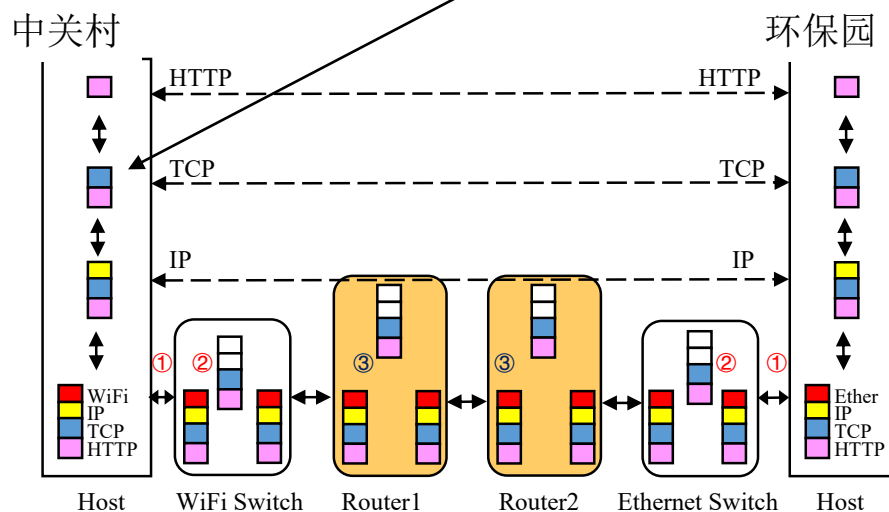
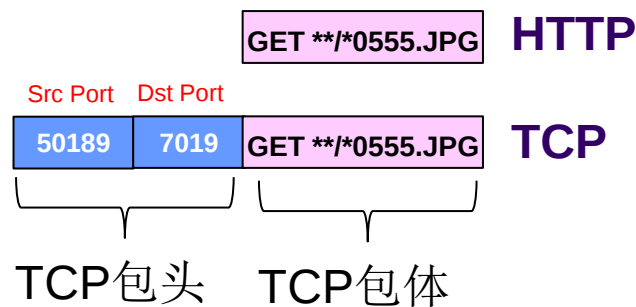


请求消息



- 从中关村计算所的笔记本电脑通过WiFi访问位于环保园服务器的“猫猫指挥家”图片；
- 笔记本电脑IP为192.168.1.101，服务器IP为10.208.104.3，图片地址为：
http://seafile.solid.things.ac.cn:7019/kitty_band/*0555.JPG

- 包头包含三类信息：地址、查错、控制信息
- 忽略了TCP包头中的查错信息与控制信息



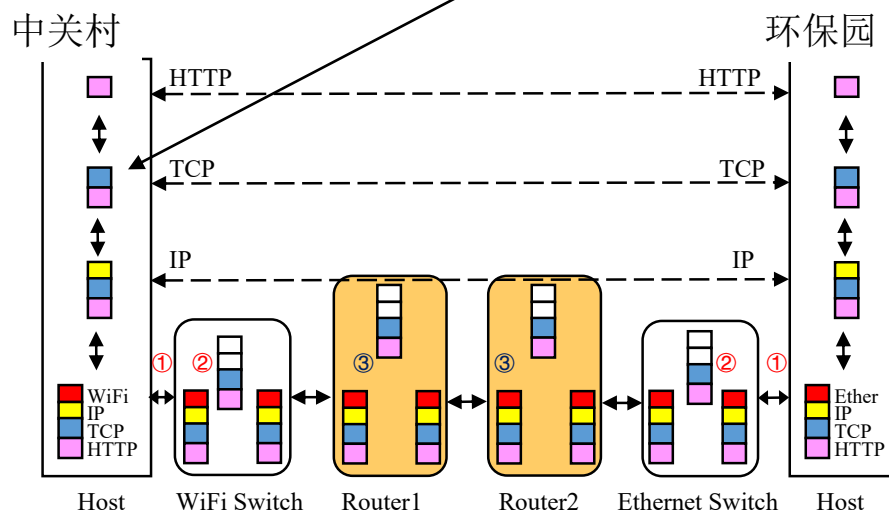
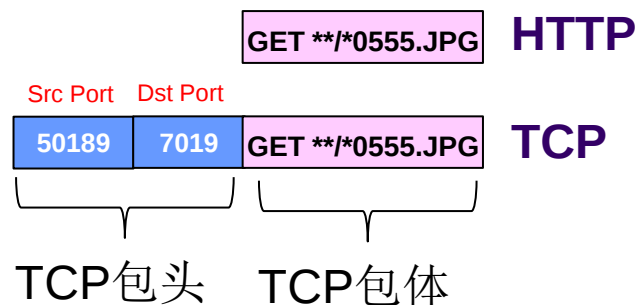
请求消息



- 从中关村计算所的笔记本电脑通过WiFi访问位于环保园服务器的“猫猫指挥家”图片；
- 笔记本电脑IP为192.168.1.101，服务器IP为10.208.104.3，图片地址为：
http://seafle.solid.things.ac.cn:7019/kitty_band/*0555.JPG

- 包头包含三类信息：地址、查错、控制信息
- 忽略了TCP包头中的查错信息与控制信息

- Src: Source 源
- Dst: Destination 目的地

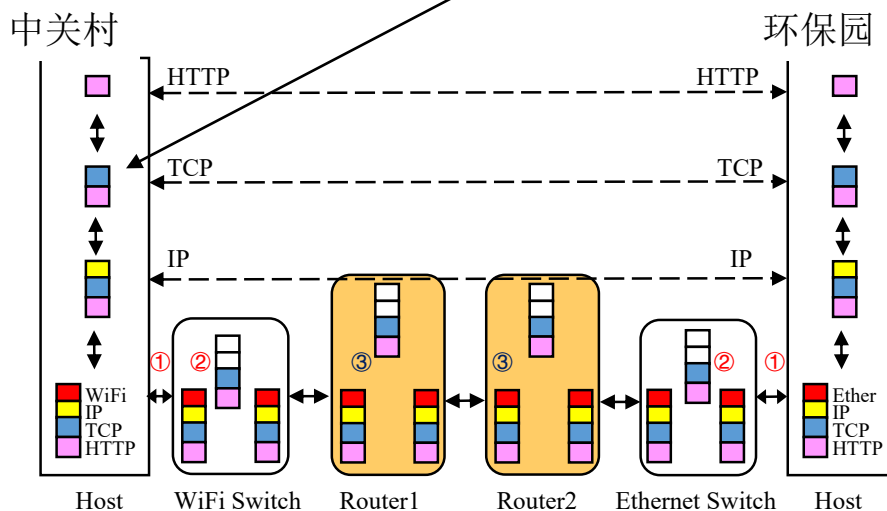
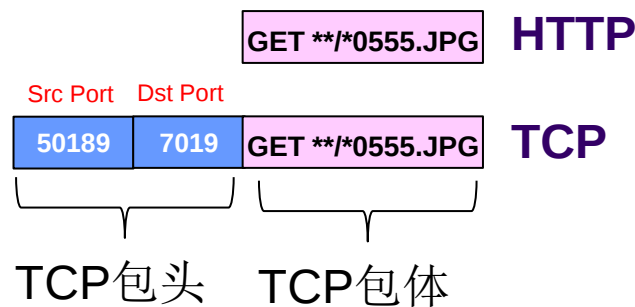


请求消息



- 从中关村计算所的笔记本电脑通过WiFi访问位于环保园服务器的“猫猫指挥家”图片；
- 笔记本电脑IP为192.168.1.101，服务器IP为10.208.104.3，图片地址为：
http://seafile.solid.things.ac.cn:7019/kitty_band/*0555.JPG

- 包头包含三类信息：地址、查错、控制信息
- 忽略了TCP包头中的查错信息与控制信息
- 目的地端口7019指代环保园服务器计算机上的Web服务器进程
- 源端口号50189指代笔记本电脑上的浏览器进程

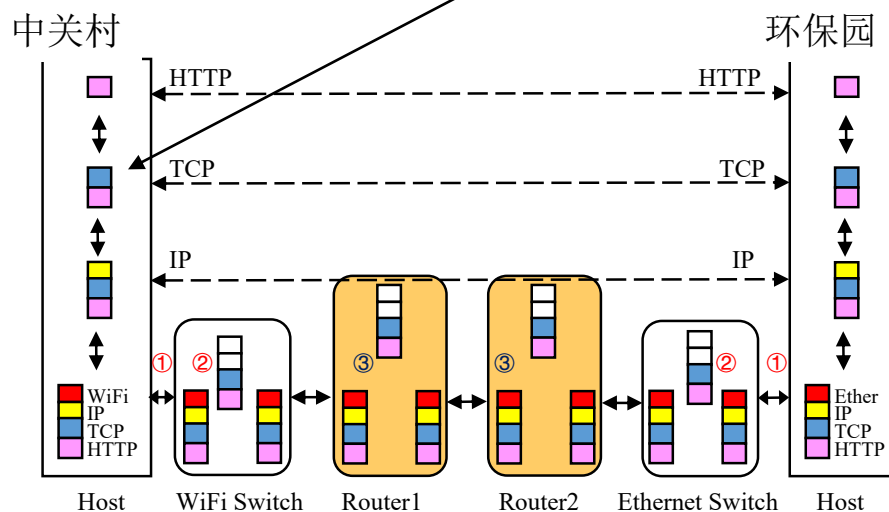
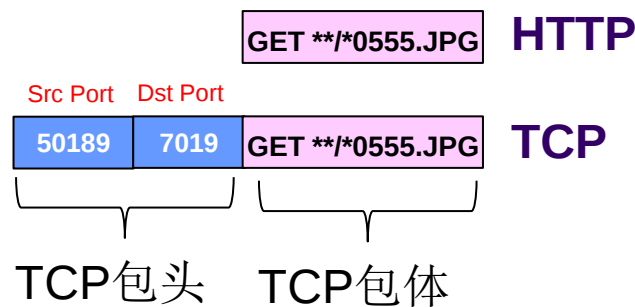


请求消息

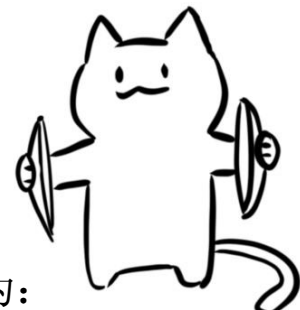


- 从中关村计算所的笔记本电脑通过WiFi访问位于环保园服务器的“猫猫指挥家”图片；
- 笔记本电脑IP为192.168.1.101，服务器IP为10.208.104.3，图片地址为：
http://seafile.solid.things.ac.cn:7019/kitty_band/*0555.JPG

- 包头包含三类信息：地址、查错、控制信息
- 忽略了TCP包头中的查错信息与控制信息
- 目的地端口7019指代环保园服务器计算机上的Web服务器进程
- 源端口号50189指代笔记本电脑上的浏览器进程



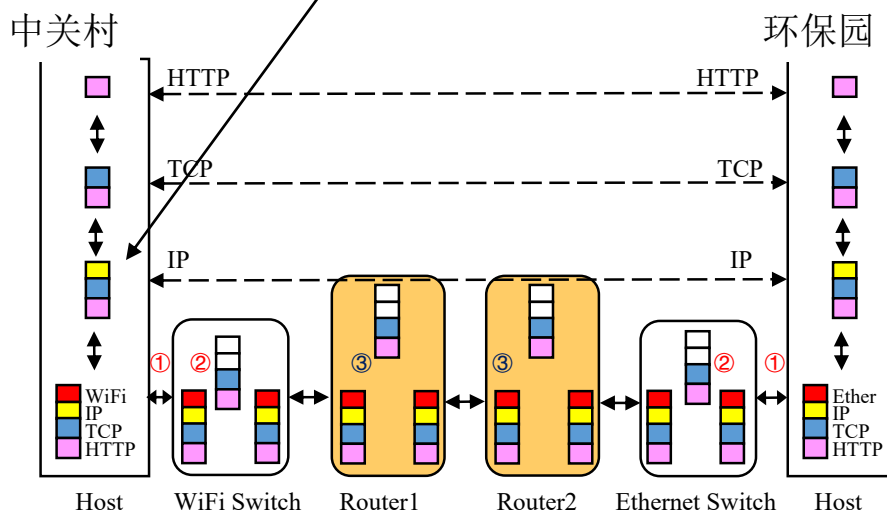
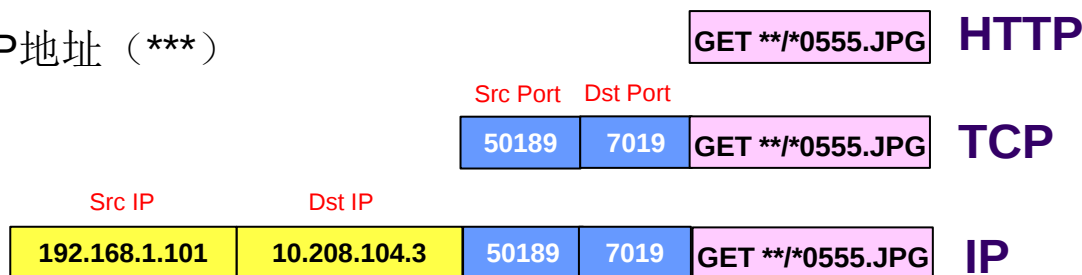
请求消息



- 从中关村计算所的笔记本电脑通过WiFi访问位于环保园服务器的“猫猫指挥家”图片；
- 笔记本电脑IP为192.168.1.101，服务器计算机IP为10.208.104.3，图片地址为：
http://seafile.solid.things.ac.cn:7019/kitty_band/*0555.JPG

- 10.208.104.3是内网IP地址 (***)

- 哪一段是IP包体？
- 哪一段是IP包头？



A simple line drawing of a cat-like creature with a mustache, holding two small circular objects. The creature has a round body, pointed ears, and a small mustache. It is holding two small, circular, disc-like objects, one in each hand. The drawing is minimalist, using only black outlines on a white background.

- GET **/*0555.JPG HTTP

50189	7019	GET **/*0555.JPG	TCP
-------	------	------------------	-----

192.168.1.101	10.208.104.3	50189	7019	GET **/*0555.JPG
---------------	--------------	-------	------	------------------

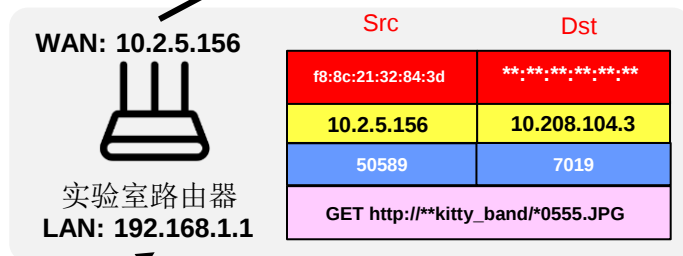
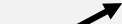
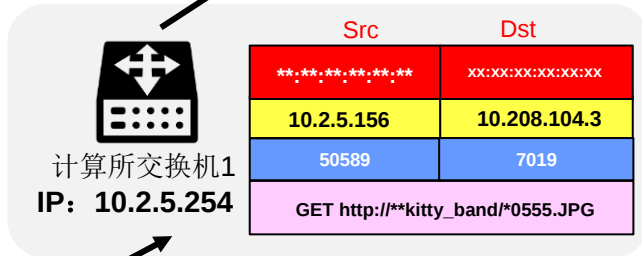
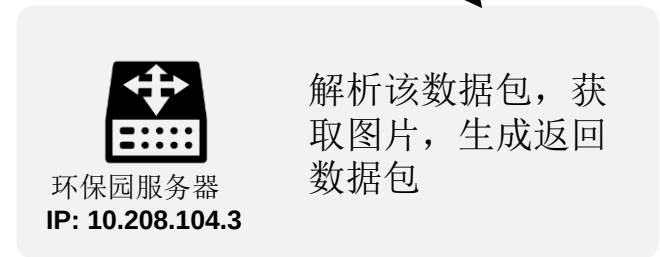
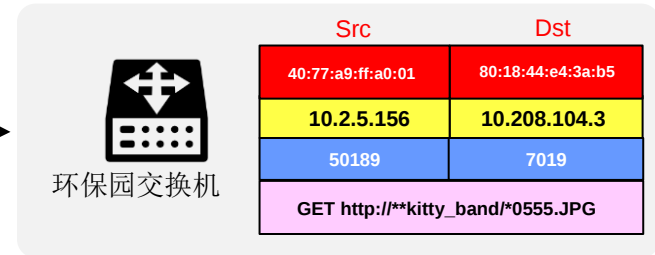
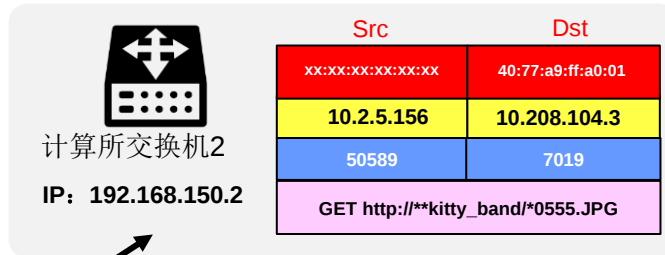
ab:bc:32:86:ee:13	f8:8c:21:32:84:3d	192.168.1.101	10.208.104.3	50189	7019	GET **/*0555.JPG
-------------------	-------------------	---------------	--------------	-------	------	------------------



From Client to Server



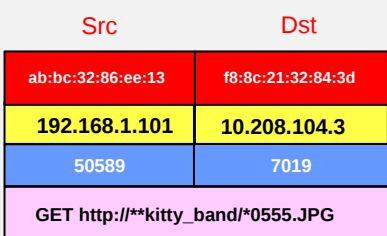
- 个人的笔记本电脑（IP：192.168.1.101）通过WiFi发出请求，经过实验室路由器、计算所交换机和环保园交换机到达环保园服务器，环保园服务器解析数据包。
- 使用tracert软件追踪从计算所到环保园全路径：



NAT地址转换



个人笔记本
IP: 192.168.1.101



```
→ ~ traceroute 10.208.104.3
traceroute to 10.208.104.3 (10.208.104.3), 64 hops max, 52 byte packets
 1 192.168.1.1 (192.168.1.1) 4.152 ms 1.153 ms 1.666 ms
 2 10.2.0.254 (10.2.0.254) 2.945 ms 1.814 ms 3.636 ms
 3 192.168.150.2 (192.168.150.2) 2.089 ms 2.760 ms 3.939 ms
 4 * * *
 5 10.208.104.3 (10.208.104.3) 6.657 ms 3.207 ms 2.875 ms
```

From Server to Client



- 环保园的服务器收到请求后，获取图片，生成响应消息的数据包，并返回给个人笔记本。
- 使用traceroute软件追踪从计算所到环保园全路径：

