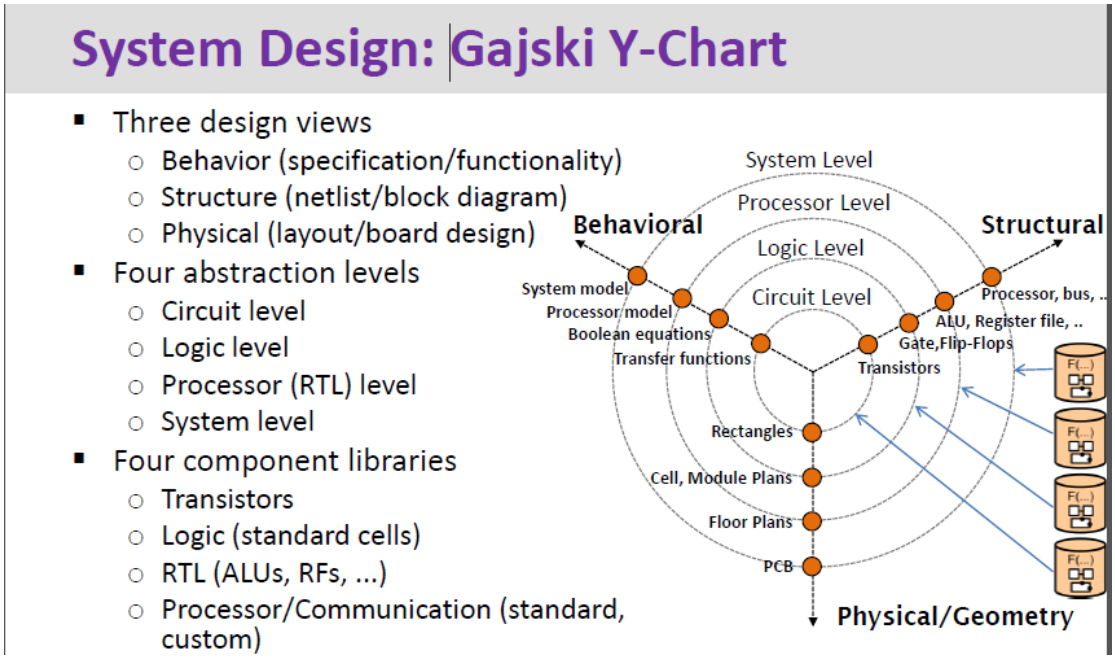
Gajski 的 Y-图(Gajski’s Y-Chart)



三个设计视角：

视角 1：行为：说明设计模型（规范、功能）:传递函数 -> 布尔表达 -> 处理器模型 -> 系统模型

视角 2：结构：说明设计架构（网表、框图）:晶体管 -> 门与触发器 -> ALU 与寄存器文件 -> 处理器与总线

视角 3：物理：说明设计基础（板图、布局）:电路元件 -> 标准单元 -> 平面板图 -> PCB

四个抽象层级：

层级 1：电路层

层级 2：逻辑层

层级 3：处理器（RTL，Register Transfer Level，寄存器转换级电路）层

层级 4：系统层

四个组成库（分别对应四个抽象层级）：

晶体管

逻辑（标准单元）

RTL（ALU，寄存器文件）

处理器/通信

综合

定义：在任意一个抽象层级中将行行为设计视角转换为结构设计视角

分类：电路层综合、逻辑层综合、处理器层综合、系统层综合

please summarize the “Y-chart” theory in Chinese,focusing on design abstraction hierarchy and design representation.

Y-chart 以不同领域的视角和不同的抽象层次来描述系统设计，在 Y-chart 中，三个轴分别表示系统的不同的领域视角。

行为设计：表示系统完成的功能，采用黑盒模型，将黑盒的行为描述为输入值和终止时间的函数。

结构表示：定义系统组件及其互联关系来描述系统的实现。
物理表示：描述组件之间的空间关系等物理属性。
其主要分为 4 个抽象层次，由一系列同心圆来表示。每个同心圆表示一个系统的不同抽象层次，从最里面开始，由低到高分别为：电路层、逻辑层、处理器层和系统层。不同抽象层次上相应各分支的内容不同，其中物理层有：电路元件、标准单元、平面板图、PCB；行为级有：传递函数、布尔表达、处理器模型、系统模型；而结构级有：晶体管、门与触发器、ALU 与寄存器文件、处理器与总线
抽象层次通常以每一层结构方面的主要组成对象来定义，并且高一层的组成对象以低一层的组成对象的组合来构成。例如，从结构领域看，一个嵌入式系统在系统层包括系统模型、PCB、处理器与总线等；而在逻辑层，系统由门与触发器等构成。同一抽象层次的行为描述以抽象功能实体的逻辑组合来描述系统的功能，而结构描述则是实现相应功能的物理结构实体的互连描述。嵌入式系统设计方法学主要是在较高层次上，由行为描述开始，逐渐转向结构描述，然后结构描述中的物理结构实体又在下一个较低抽象层次再次被转换到相应的结构描述，一直到真正地实现系统，具体的电路被制造出来。
系统从行为和结构两个视角分别考虑，体现了关注点的有效分离，即系统功能描述的实现无关性，一方面可以有效的提高开发效率，另一方面具体的实现技术可以变化，从而延长设计生命周期。一般的，同一抽象层次从行为描述自动或半自动地转换到结构描述的过程称为综合。由于不同视角关注的特征不同，并且结构描述趋于实现，要处理的细节更多，因而在综合的设计过程中需要细化和添加设计决策等等。设计者需要在不同抽象层次上自上而下反复的细化和验证直到最后一个实现。因而随着复杂性的增长，设计成本也在急剧的增长。

Y-chart ，Y 图理论首先是由 Gajski 于 1983 年提出的。用于集成电路的设计领域。Y-chart 把设计过程以模型表示出来，层次清晰，结构明了。
Y-chart 模型中抽象出了 3 个轴，分别是 Behavioral 行为， structural 结构，Geometry 几何。设计中的层次及结构用一系列同心圆表示，由内到外越来越抽象，分别代表着一个模型。B S G 这三个轴将平面分为三个区域，分别是行为域，结构域和几何域。行为域描述了系统当中的功能特性，与硬件的设计无关，是描述了诸如算法数理文本等信息。结构域描述了各个子系统的聚集，因此子系统构成与之间的关系是重点。几何域关注的事系统的几何属性与其下属子系统，因此在系统的物理层面上的信息至关重要。
在对着三种域的表示时又分别有三种表示方式：行为表示，结构表示，几何表示。行为表示表达了算法理论层面的设计，研究系统要实现的功能，内部的实现结构并不在考虑范围里。结构表示关心的是系统的抽象设计结构，包括组成部件子系统和它们之间的联系。几何表示关心的是底层物理设计的实现，包括芯片的性能，器件的选取组合等等。
Ychart 模型以抽象的模型把 IC 设计的各个层面的问题展现出来，逻辑结构清晰，对设计研究集成电路有很大帮助。

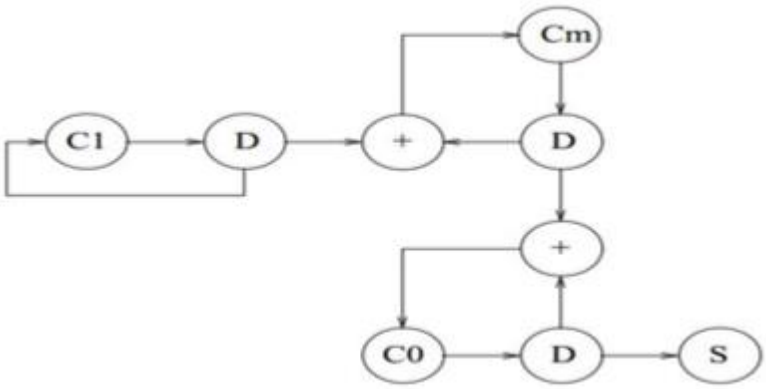
please summarize the “Y-chart” theory, including(1)design representation domains,(2)design abstraction hierarchy and(3)design activities.

Y-chart 理论由 Gajski 和库恩于 1983 年提出，这个图使得数字集成系统的设计意见以及设计层次变得可视化，同时，它也广泛地在 VHDL 设计中使用。这一理论让我们的想法在抽象的层次上建立模型，从不同的角度看，整个设计的布局一目了然。
在 Y-chart 中，三个轴分别表示行为表示、结构表示、几何表示三种领域的视角。也就是行为域、结构域和几何域。各个轴上的点表示设计过程中的各个层级的结构，由内向外，抽象

层级越来越高，从最里面开始，由低到高分别为：电路层、逻辑层、处理器层和系统层。其中行为域从里到外是：传递函数、布尔表达、处理器模型、系统模型；结构域是：晶体管、门与触发器、ALU 与寄存器文件、处理器与总线；而几何域是：电路元件、标准单元、平面板图、PCB。
设计的抽象层次包括行为域和结构域。行为域描述了系统中的功能特性，不涉及具体的硬件实现，在行为域中，一个部件通过定义它的输入/输出响应来描述。结构域是行为域和结构域的连接，在结构域中，一个部件通过一些基本部件的互联来描述。行为域转化到结构域的过程称为综合，结构域转化到几何域的过程称为物理设计，行为域转化到几何域的过程称为编译。其中综合和物理设计包括两个阶段。设计过程开始于行为域，并向下经过系统和处理器层。在处理器层，该层的行为化的数据流表示被转换成结构化的逻辑层描述，逻辑层描述又被转换到电路层。

Kahn 进程网络(Kahn Process Network)
确定性：在给定各信道输入序列的情况下，即使输入序列的到来顺序不同，输出序列也随之确定
公平性：一个信道的输入情况不造成其它信道的饥饿
随机系统：在给定各信道输入序列的情况下，在输入序列到来顺序不同的情况下可能出现不同输出序列的系统称为随机系统
确定性系统：在给定各信道输入序列的情况下，即使输入序列的到来顺序不同，输出序列也随之确定的系统称为确定性系统
在确定性系统中系统的行为和时序无关，从而可以将两者分离开来考虑
Kahn 进程网络的评价
优势：
1、将调度和进程的具体内部功能相分离
2、基于数据流
3、体现了并行性
4、很容易将图映射到实际物理器件上
劣势：
1、很难安排每个进程的处理速度（因此缓冲区里的数据不好估计）
2、调度困难，而且必须要在运行时调度（运行前无法估计调度情况）

根据下列基本进程画出输出序列为 n(n+1)/2 的 KPN 图
加法器：两路输入，一路输出，输出为两路输入相加的结果
乘法器：两路输入，一路输出，输出为两路输入相乘的结果
复制器：一路输入，两路输出，两路输出与输入完全相同
常量生成器：一路输入，一路输出，初始输出为预设的初始值，在此之后输出值等于输入值
汇聚：将输入抛出以显示最终结果
分析：一开始先生成全 1 序列，然后通过这个再生成 1，2，3...的等差数列，最后再生成数列相加后的结果



SDF 调度
相对运行率的计算
步骤 1：将进程看成变量，每条边一个方程，出边为加，入边为减
步骤 2：判断是否可以调度
当第一步得到的线性方程组的列空间的维数（矩阵的秩）为变量数 n - 1 时说明存在周期性调度，此时存在唯一的最小正整数解

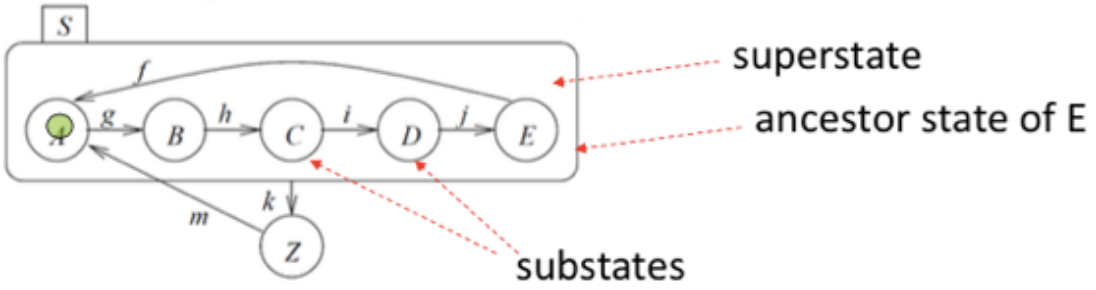
如果第一步得到的线性方程组的列空间的维数为变量数 n 说明这个系统不相容
如果第一步得到的线性方程组的列空间的维数小于变量数 n - 1 说明这个系统不能连通，需要划分
如何计算列空间的维数（矩阵的秩）：将方程进行初等行变换，然后看主元的数目，有多少个主元列空间维数就是多少
如何得到最小正整数解：将方程进行初等行变换后用一个变量表示其他变量，然后令最小的那个为 1，最后再看有没有非整数的，将所有变量扩大 K 倍变成整数
SDF 的评价
优势：可以在编译时确定调度
劣势：
1、过程限制多（进程启动过程中不能中断，而且定量输入输出）
2、对于非功能性属性无法建模（比如能量，可靠性等等）

状态图
状态图就是显示各个状态及状态转移关系的图，其主要由以下要素组成
活动状态(active state)：当前处于的状态

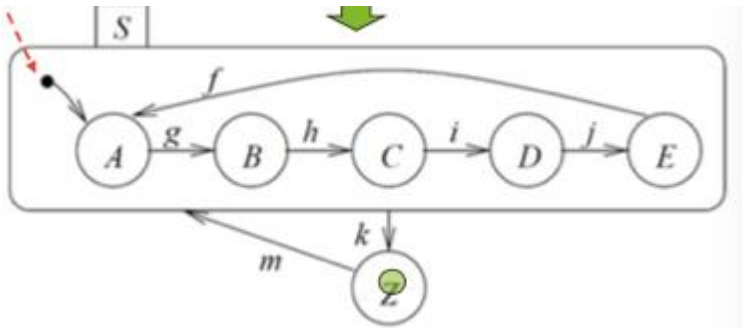
基本状态(basic state)：不包含子状态的状态，用圆形表示示

超状态(super state)：包含子状态的状态，用圆角矩形表示

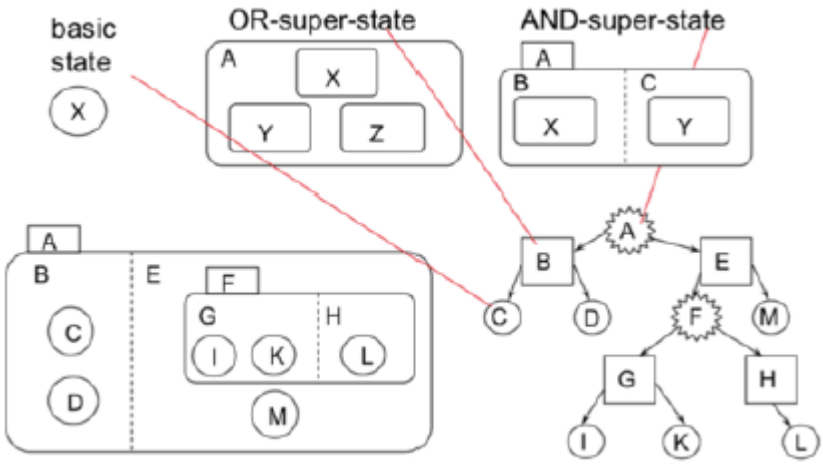
或类型超状态(OR super state)：只要这个超状态中的任意一一个子状态处于活动状态，这个超状态就处于活动状态
与类型超状态(AND super state)：只有这个超状态中的全部子状态处于活动状态，这个超状态才处于活动状态



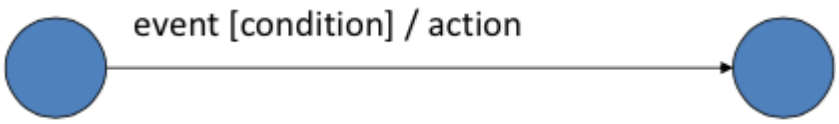
默认状态(default state)：初始时进入的状态，用黑色圆点加带箭头的线指向的状态就是默认状态



状态集的树状表达
基本状态：用圆形表示
或类型超状态：用矩形表示
与类型超状态：用星形表示



通过树状表达提取状态集
三种基本状态集的转换方法
基本状态的状态集就是基本状态本身
或类型超状态的状态集是它所有子状态的并集
与类型超状态的状态集是它所有子状态的笛卡尔积
状态转换三要素
1、事件(event)：外部触发事件，可以是单个事件，也可以是事件的交集、并集、补集
2、状态(condition)：状态机的内部状态
3、动作(action)：在状态跳转过程中伴随的行为，可以是单个事件，也可以是同时并发的多个事件



状态图转换为有限状态机
步骤 1、画好所有的基本状态
步骤 2、对所有的基本状态考虑所有的外部触发事件
步骤 3、去除无法到达的基本状态
评价
优势：
1、层次结构，可以很清晰地表达事件的层次性
2、理解简单，状态关系直观
3、实现简单，有很多工具可以帮助实现
劣势：
1、效率低
2、不适合分布式应用（因为无法体现分布式）
3、不能描述非功能性行为
4、不是面向对象的（面向过程）
5、对层次结构的描述不足

资源访问协议
优先级继承协议(PIP)：当一个优先级较低的线程阻塞了一个优先级较高的线程时会将这个优先级较低的线程的优先级提高至被阻塞的进程的优先级
算法描述：
1、如果进程 Ti 试图进入临界区会出现下列两种情况
如果临界区资源被一个较高优先级的进程占用就会被阻塞，不进行后续操作
如果临界区资源被一个较低优先级的进程占用就会被阻塞，并进行第二步，否则就立即进入临界区
2、当进程 Ti 被阻塞时，它将自己的优先级给到当前拥有信号量的进程 Tk。在当前拥有信号量的进程 Tk 离开临界区之前会保持这个优先级
3、当进程 Tk 离开临界区以后，会使用 V 操作将信号量释放，然后这个信号量会给到当前阻塞队列中优先级最高的进程。如果只有一个进程被阻塞，那么 Tk 会立即恢复为它原本的优先级，否则会改变为现在仍在阻塞队列中的进程的最高优先级
不足之处：
1、可能导致很多进程的优先级都很高
2、可能导致死锁
3、必须固定优先级

优先级上限协议(PCP)：一个进入临界区的进程不会被低优先级进程所阻塞，为每个信号量设定一个优先级上限说明获取这个信号量的最高优先级是什么
算法描述：
1、当一个进程 T 想要得到信号量 Sk 时，Sk 会分配给进程 T 当且仅当进程 T 的优先级大于信号量 S*的优先级上限，S*为被进程 T 以外的进程所拥有的信号量中具有最高优先级上限的的信号量，如果进程 T 不满足拥有信号量 Sk 的条件则认为进程 T 被信号量 S*所阻塞，如

果进程 T 被信号量 S*所阻塞，那么进程 T 的优先级将会给到当前拥有信号量 S*的进程
2、当一个进程 T*释放信号量 S*时，系统将信号量 S*给到当前被信号量 S*所阻塞的最高优先级进程，T*的优先级将设为它还拥有的信号量所阻塞的进程的最大优先级，如果 T*已经不拥有任何进程了，就恢复为初始时的优先级
评价
这个算法比 PIP 的好处在于
1、不会死锁
2、一个进程最多只会被每个低优先级进程阻塞一次，但还是不能应对优先级改变的情况

1、说明私有资源、共享资源和互斥资源三者间的区别，然后用例子说明互斥资源会遇到的一个问题。
私有资源只能被一个进程所使用，共享资源可以被多个进程所使用
共享资源可以同时被多个进程说访问，互斥资源在某一时刻最多只能被一个进程所访问
互斥资源会遇到临界区问题（共享变量的访问问题）
例子：
在生产者消费者问题中：
生产者执行：
register 1 = count
register 1 = register 1 + 1
count = register1
消费者执行：
register 2 = count
register 2 = register 2 - 1
count = register2
假设一开始的 count 值为 5
在一般情况下生产一个再消费一个最后得到的 count 应该是和原来相同，还是 5
但是如果是下面这种情况：
register 1 = count
register 1 = register 1 + 1
生产者执行完这两步就被消费者中断掉了
register 2 = count
register 2 = register 2 - 1
消费者执行完这两步又被生产者中断掉了
count = register1
count = register2
最终得到的 count 值就不是 5 了，而是 4
这个也就是临界区问题：对共享数据的并发访问可能导致数据的不一致
2、请说出解决临界区问题的三种方法

设定进程非抢占，硬件上屏蔽中断，静态调度（一开始就认为规定好调度方方式），信号量

3、请解释信号量是怎样解决临界区问题的
在信号量中定义了两种原子操作
P 操作：请求进入临界区的锁
V 操作：释放临界区的锁

一个进程 A 要想进入到临界区中必须先通过 P 操作请求进入临界区的锁，如果临界区里面有其他进程，那么进程 A 就必须等到哪个进程从临界区里面出来（被放入这个信号量的阻塞队列中），通过 V 操作释放锁以后，进程 A 才可以进入临界区；如果临界区里面没有其他进程，那么进程 A 就可以立即进入到临界区。符合有空让进，无空等待，择一而入，算法可行行的临界区问题解决原则。

Question 2. What are the steps required to initialize a parallel port? Which steps are optional?

Clock 激活时钟

Wait two bus cycles 等待两个总线周期

Direction register (0 for input, 1 for output)设置方向寄存器，0 表示输入，1 表示输出

Clear AFSEL bits (optional, because reset clears them))设置 AFSEL 为 0，选择常用的 I/O 端口（可选）

Clear AMSEL bits (optional, because reset clears them)设置 AMSEL 为 0，禁用模拟功能（可选）

Clear PCTL bits (optional, because reset clears them)设置 PCTL 为 0，选择 GPIO 功能（可选）

Set bits in DEN register 设置 DEN 为 1，允许数据输入输出

读-改-写：

```
PortF_Init
    LDR R1, =SYSCTL_RCGCGPIO_R
    LDR R0, [R1]
    ORR R0, R0, #0x20                    ; set bit 5 to turn on Port F clock
    STR R0, [R1]                         ;激活时钟
    NOP
    NOP                                  ; 等待两个总线周期
    LDR R1, =GPIO_PORTF_LOCK_R           ; 2) unlock the lock register
        LDR R0, =0x4C4F434B                  ; unlock GPIO Port F Commit Register
        STR R0, [R1]
        LDR R1, =GPIO_PORTF_CR_R             ; enable commit for Port F
        MOV R0, #0xFF                        ; 1 means allow access
        STR R0, [R1]
        LDR R1, =GPIO_PORTF_AMSEL_R      ; 3) disable analog functionality
        MOV R0, #0                               ; 0 means analog is off
        STR R0, [R1]
    LDR R1, =GPIO_PORTF_PCTL_R           ; 4) configure as GPIO
        MOV R0, #0x00000000                      ; 0 means configure Port F as GPIO
        STR R0, [R1]
        LDR R1, =GPIO_PORTF_DIR_R            ; 5) set direction register
        MOV R0,#0x0E                                     ; PF0 and PF7-4 input, PF3-
1 output
        STR R0, [R1]
        LDR R1, =GPIO_PORTF_AFSEL_R          ; 6) regular port function
        MOV R0, #0                                       ; 0 means disable
```

```
alternate function
        STR R0, [R1]
        LDR R1, =GPIO_PORTF_PUR_R       ; pull-up resistors for PF4,PF0
        MOV R0, #0x11                   ; enable weak pull-up on PF0
and PF4
        STR R0, [R1]
        LDR R1, =GPIO_PORTF_DEN_R       ; 7) enable Port F digital port
        MOV R0, #0xFF                   ; 1 means enable digital I/O
        STR R0, [R1]
        BX  LR
```