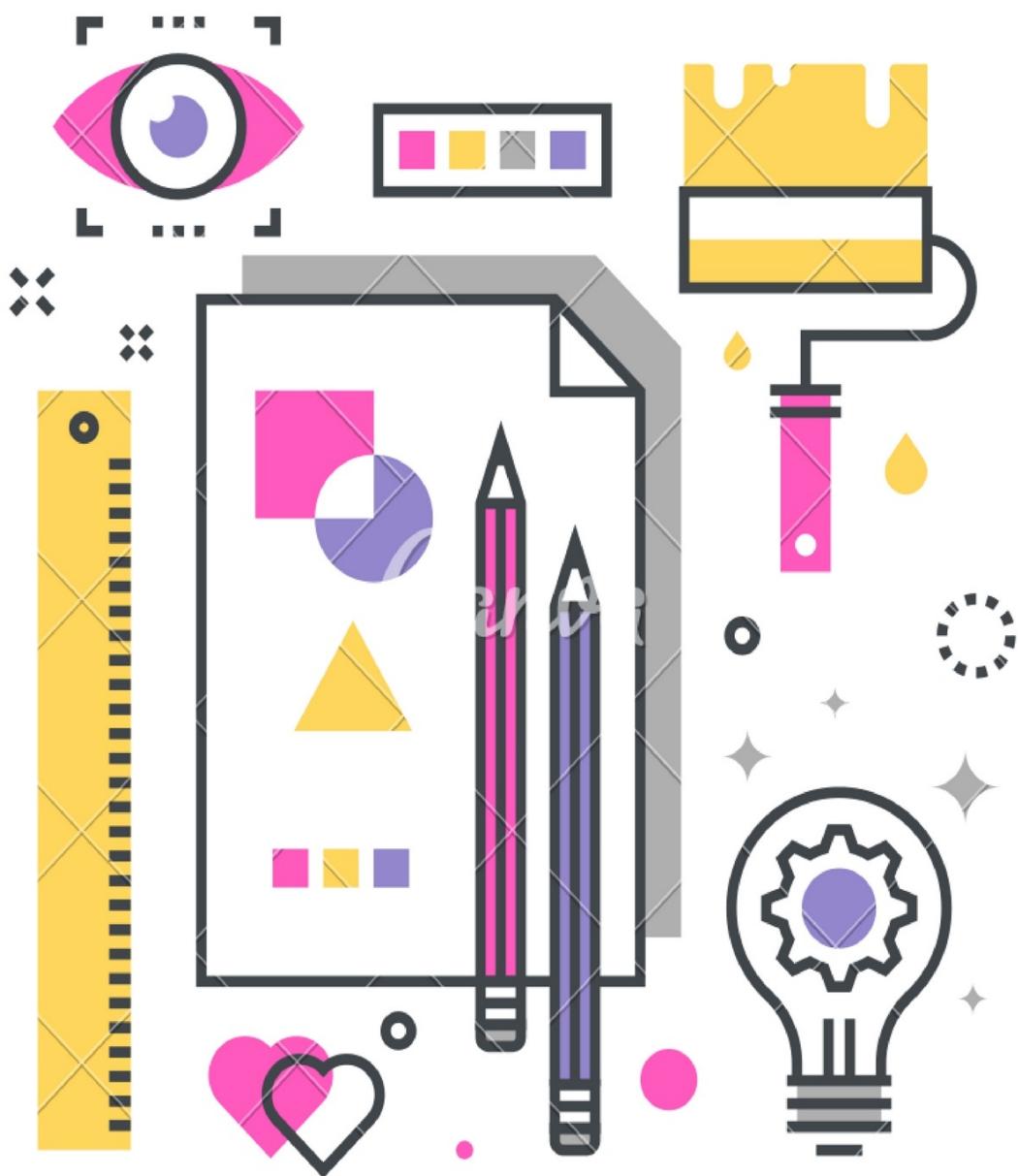


一份开源的IC 读物

了解CAD 知识，了解行业大牛，了解IC 方法论

IC GEEK

IC 极客



BY ICXHUB

目錄

关于本书	1.1
发刊词	1.2
编程@IC	1.3
IC 为何偏恋三十而立的Tcl	1.3.1
IC极客的技术雷达	1.4
终于懂得IP 对于芯片如此珍贵，那么...	1.4.1
流程构建的核心能力	1.5
“刷脚本” 的学术讨论	1.5.1
技术管理的核心能力	1.6
IC设计中观、微观与宏观	1.7
Lab	1.8
Tcl Seminar Day 1	1.8.1
Tcl Seminar Day 2	1.8.2
Tcl Seminar Day 3	1.8.3
Tcl Seminar Day 4	1.8.4
Tcl Seminar Day 5	1.8.5

IC 极客

本专栏为内容开源专栏, 使用Gitbook 进行书本的制作。

本内容遵循“知识共享许可协议”

[Attribution-NoDerivatives 4.0 International \(CC BY-ND 4.0\)](#)

专栏内容已接入 "[原本链](#)" 原创认证。

技术服务支持 : support@icxhub.com

产品购洽联系 : sale@icxhub.com

1. 成书使用的开源工具

- Dia - 流程图 (*.dia)
- GIMP - 图像编辑 (*.xcf)
- Gitbook - 内容组织
- Calibre - PDF 生成
- Node.js - gitbook 依赖
- VSCode - 文本编辑

开刊词



本文经「原本」原创认证，访问[yuanben.io](https://yuanben.io/4waivguz)查询【4WAIVGUZ】获取授权信息。

“IC极客”专栏开栏了，特邀您入圈。这是Alice和她的小伙伴们共同发起的专栏，我是Alice，IC行业的一名CAD老兵。在本栏，“IC极客”不特指IC世界里佝偻在电脑前深宅的程序员，我们将它定义为一种对IC技术的好奇心和让自己、团队乃至整个行业变得更好的执行力。接下6个月，在“IC极客”专栏你会看到丰富、夯实、专业的内容。

1. | AI 已来，如何应对人类学习的焦虑

2016年阿老师战胜世界顶级围棋高手李世石，机器学习正式走进大众视野，引起了人们极大的好奇及『讨论』上的恐慌。2017年，国内半导体行业如火如荼，上有国家战略和政策扶持，下有各路金主爸爸强力进入，Startup公司如雨后大蘑菇，成片冒出。魏老师说AI无疑是当前社会最关注的热点，不管有什么好的AI算法，要想最终得到应用，就必然要通过芯片来实现。未来已来，无论好坏。

当前世界在快轨上狂奔，基于新技术的新机遇层出不穷，却又稍纵即逝，只有那些跟得上时代步伐且掌握新技术的人才能抓住这些机遇。如何跟上世界狂奔的步伐，如何及时掌握最新的技术，如何抓住稍纵即逝的机遇，这些大概就是焦虑的来源。“IC极客”专栏将试着去探讨：IC工程师在各自的工作岗位上需要的学习能力及如何对抗随之而来的焦虑，试试看能不能找到一种行之有效的方法。

2. | 连接进IC 极客圈的思维矩阵

曾被合伙人间及：Alice你刚工作的时候，公司给你们培训什么？我想了想，那时公司有内部的e-learning系统，里面有工具、工艺、设计、编程等各个领域的培训资料，可以自己进去看，讲师都是公司的一线大咖。与此同时在具体工作中每个人都有自己的mentor。

现在想来这样的系统也有其弊端，第一，该培训系统致力于把人培养成细分领域的操作工，而不负责培养某种学习能力，也不负责教授面向行业通用的领域知识；第二，不是每个mentor，都具备做mentor的素养和技术管理能力；第三，很多人在自己职业成长的路上看不到Roadmap。前两天有个刚入行的孩子跑来问我，学姐，我想做到像你一样，有什么方法？这个问题着实不好回答，但那孩子还是很执拗地问，每个阶段需要学习什么内容、达到什么目标、用什么方法、有没有一个guideline？对于这样大而有的问题，简单回答基本没有什么实质帮助。

所以，在这次专栏内容的制作中，我们会找一些真正的IC极客，用六个月的时间来共同探讨这些问题。我们将试图创建一个“IC极客圈”，聚集一群以极客精神对IC设计的细分领域或者整个行业有深刻好奇心和行动力的人。他们是来自IC设计、前端、后端、验证、模拟等各个领域的资深工程师、技术经理或某位行业大咖。文章内容将以多种方式呈现，希望IC极客们的思维方式方法可以给大家在未来持续学习过程中带来一些启发。

我们期待富有极客精神的你愿意把自己连接进这个思维矩阵，大家一起行动。本着极客圈『开源和分享』的精神内核，整个IC极客专栏的文字及音视频内容将全部以开源、共享的形态呈现给大家。

3. | 未来6个月你将得到

在未来6个月里，我们会以编程@IC设计，IC极客的技术雷达，流程构建的核心能力，技术管理的核心能力，IC设计中观、微观与宏观为主线，探讨一些通用和细分领域的知识、技能及方法，分享对相关领域的问题思考、思维模型和方法论。为了夯实，会引入工程实践中的具体问题、具体案例及具体的解决之方。这不是一个完整的课程体系，每个模块，都是每个硅农在职业成长中或早或晚会遇到的通识性问题集结。

1. 编程@IC设计

这个模块主要由几位资深的CAD操刀，梳理编程思想以及不同的编程语言在IC设计各个领域的应用和最佳实践。时下最热门的计算机思维、大数据概念、最热的编程语言都会涵盖其中。

1. IC极客的技术雷达

这个模块由IC设计各个领域的专家，特别是技术管理者，分享工程师在职业成长中的技术雷达和升级打怪指南。也会对每个领域的技术趋势做深入探讨。

1. 流程构建的核心能力

在国内流程构建主要是由领域专家和CAD共同完成，在不同企业里有不同的策略。完成流程构建需要关注哪些核心能力、需要采用何种技术方、如何实现弹性设计？就这些问题我们将做深入探讨。

1. 技术管理的核心能力

成为技术管理者是大部分工程师职业规划中的一个『小目标』，怎样从职场小白攀爬到管理层？技术管理者需要具备怎样的核心能力？作为工程师如何向上管理？就此，希望可以提供一些思维模型和行动指南助你事半功倍。

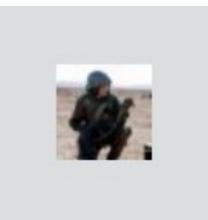
1. IC设计中观、微观与宏观

抱着flow刷脚本，一般出不了什么大乱子，不按规定来，芯片容易做成砖。这里的Flow可以理解为套路，是一个中观的概念。但是，抱着Flow刷脚本是远远不够的，你需要对每个不同的case有微观认知，这才是长在你身上的know-how，是flow不可替代的。宏观是更高一级的抽象，作为一名IC工程师需要具备zoom in和zoom out的能力，要既能看到宏观又能深入微观，这才算具备了高效定位和解决问题的能力。这一模块会探讨如何在升级打怪中一步步地练出这个能力。

期待在这里和你相遇！Alice

文章中所提到的延伸阅读、代码分享、后续更新通知及音视频上线通知都会发在IC极客交流群，同时，IC极客圈的小伙伴们也会在群里和大家一起交流学习。

欢迎扫码进群：



IC 极客交流群



该二维码7天内(4月24日前)有效，重新进入将
更新

或加sgsphoto 好友，拉你进群：



1.2.2 - 加好友

IC 为何偏恋三十而立的Tcl



本文经「原本」原创认证，访问[yuanben.io](https://yuanben.io/52hs9df0)查询【52HS9DF0】获取授权信息。

Tcl，发音做“tickle”，是Tool Command Language的首字母缩写。

有些编程语言单独看并没有优势，脱离开应用场景后甚至是一门即将被淘汰的语言，我认为Tcl就属于这种，本篇将探讨Tcl没有被淘汰、被取代的原因。

2018，Tcl诞生已有三十个年头，它依然做为IC设计的主力语言存在着，为什么IC偏偏恋上了Tcl，本文就Tcl这门语言的特点谈谈我对这个话题的看法，用开放的心态聊聊Tcl。

如果你希望了解更多程序语言发明大牛对Tcl和其他一些程序有趣的论战，可以去搜索“Tcl War”。

- 第一部分有一个Tcl的小测验，看看你可以得几分，了解你和Tcl的亲密度。
 - 槽点：要不要评论回复下你的分数：）
- 第二部分细数了Tcl的优点和不足，快而立之年的Tcl为何依然在IC星球发光发亮。
 - 看点：了解Tcl语言产生的背景、历史与演变，或许可以给我们一点启发，我们为什么还在用它。
- 第三部分会介绍一些我的经验，我心目中好IC Tcl程序的特点以及推荐两个好的实例。
 - 看点：一点学习Tcl的体会，即使你厌恶编程，为了提高效率，为了绩效，为了工资，为了面试，你应该看一眼。
 - 亮点：只用Tcl、Tk、Expect也可以创造一个自己的IC黑客帝国。

1. 测一测你对Tcl的了解

如果不知道或者回答否则不得分，回答是或者知道得1分。6-8分及以上说明你对Tcl的了解还是不错的。

- 不管你得几分，不管你喜不喜欢Tcl - 这篇文章都可以给你一些故事和建议。
- 多家EDA工具的Tcl Shell和Linux/Unix上原生Tcl Shell，你（觉得）哪个好用？
- 你是否（熟练使用）下面全部的命令(1)？
 - set
 - puts
 - array
 - source
 - if/switch
 - for/foreach
- 你是否（熟练使用）下面全部的命令(2)？
 - open/close
 - file
 - regexp/regsub
 - proc
 - clock
- 你是否（写过）procedure？
- 你知道2个以上Tcl中作用域相关的命令吗？

- 你是否(用过)Pakcage, 标准库的Package或者其他人写的Package ?
- 你是否(写过)Package, 并知道它如何被使用 ?
- 你(用过)dict 吗 ?
- 你(知道)namespace 和它的用法吗 ?
- 你(用过)interp 这个命令吗 ?
- 你(知道)Tcl 如何Trace 吗 ?
- 你(知道)有个TK 库Bwidget吗 ?
- 你是否(知道)如何获取http 数据 ?
- 你是否(用过)wait 和send 相关的命令 ?
- 你(编写)过C 语言功能并在tcl 中load 吗 ?

2. Tcl 的诞生历史

看点: 了解Tcl 语言产生的背景、历史与演变, 或许可以给我们一点启发, 我们为什么还在用它。

Tcl/Tk 的发明人是伯克利大学的教授John Ousterhout, 八十年代初, 在教学过程中, 他发现在集成电路CAD 设计中, 很多时间都花在编程建立测试环境上。并且环境一旦发生了变化, 就要重新修改代码以适应。这种费力而又低效的方法, 迫使 Ousterhout 教授力图寻找一种新的编程语言, 它即要有好的代码可重用性, 又要简单易学, 这样就促成了 Tcl (Tool Command Language) 语言的产生。

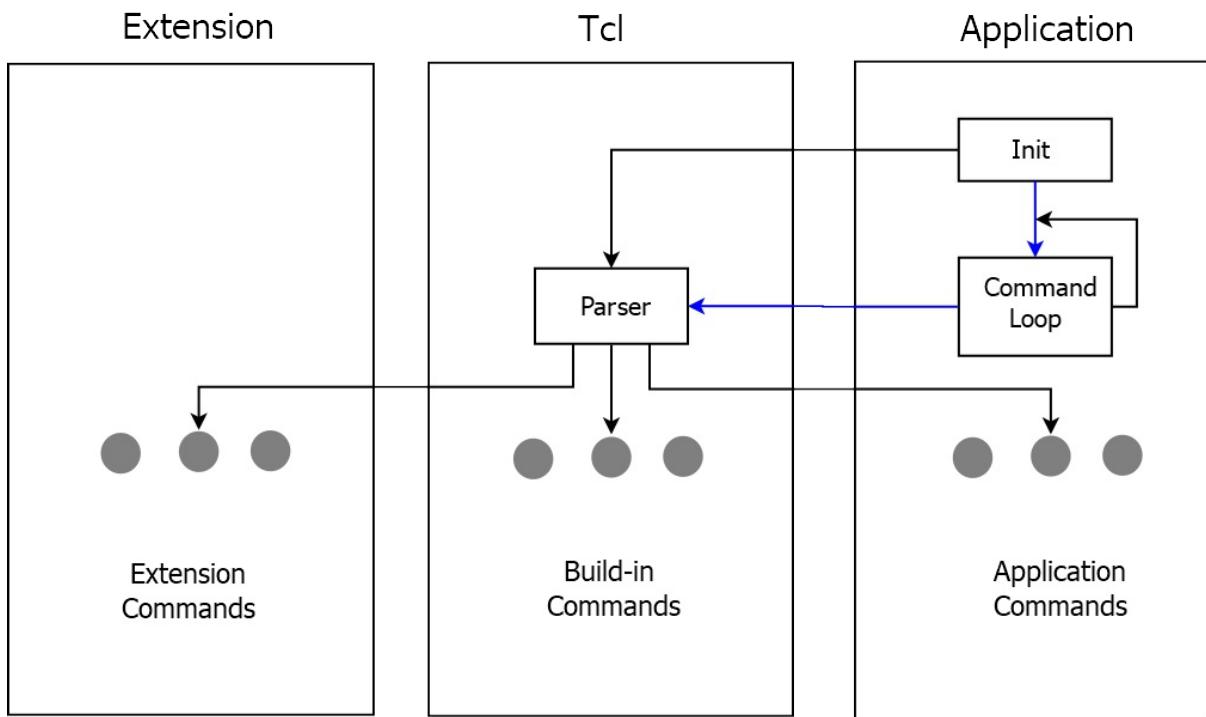
1994年Ousterhout教授被SUN招至麾下, 加入了SUNLab, 领导一个小组从事将 Tcl 移植到所有其它平台的工作, 如 Windows 和 Macintosh。同时为 Tcl 增加了 Safe-Tcl 安全模块, 并为浏览器开发了 Tcl plug-in, 以及支持 Java bytecode 的编译器、大字符集、新IO接口以及与Java的计算平台相连等。Tcl还有自己的浏览器。在面向对象程序设计占主导地位的今天, 又开发了支持面向对象的 incr Tcl。为了鼓励各厂商开发第三方的程序, Tcl 的源代码可免费下载。所有这些努力使 Tcl 成为一个适应当代信息产业潮流的、支持多平台的、优秀的开发语言。

Ousterhout 教授于 1998 年初离开了 SUN, 自立 Scriptics 公司, 继续 Tcl/Tk 的研究和开发工作。

Tcl 最初的构想的是希望把编程做成基于组件的方式 (component approach), 即: 与其为单个的应用程序编写成百上千行的程序代码, 不如寻找一个种方法将程序分割成一个个小的、具备一定“完整”功能的、可重复使用的组件。这些小组件小到可以基本满足一些独立应用程序的需求, 其它部分可在这些小组件的功能基础上生成。不同组件有不同功能, 用于不同目的, 并可为其它应用程序所调用。

当然, 这种语言还要有良好的扩展性, 以便用户为其增添新的功能模块。最后, 需要用一种强的, 灵活的“胶水”把这些组件“粘”合在一起, 使各个组件之间可互相“通信”, 协同工作。程序设计有如拼图游戏一样, 这种设计思想与后来的 Java 不谋而合。终于在 1988 年的春天, 这种强大灵活的胶水 - **Tcl** 语言被发明出来了。

按照Ousterhout 教授的定义, **Tcl** 是一种可嵌入的命令脚本化语言 (**Command Script Language**)。“可嵌入”是指把很多应用有效、无缝地集成在一起。“命令”是指每一条 Tcl 语句都可以理解成命令加参数的形式:



从图中可以了解到一个很重要的概念，输入到Tcl的所有交互指令或者其他接口都被认为是文本，按一定的顺序进行Parsing。这个特性在某些应用场景变得很有用，命令也是数据(某些其他语言在命令和数据方面的互通性更好)。

EDA软件对Tcl的支持现在已经到了8.6，由于其扩展性好，调试简单，C接口友好，使用者学习成本低，这些都是商用软件的价值所在，更重要的是开源，免费。在IC设计流程中，有些文件格式是基于Tcl的，比如SDC。

Date	Event
January 1990	Tcl announced beyond Berkeley (Winter USENIX).
June 1990	Expect announced (Summer USENIX).
January 1991	First announcement of Tk (Winter USENIX).
June 1993	First Tcl/Tk conference (Berkeley). [table] geometry manager (forerunner of [grid]), [incr Tcl], TclDP and Groupkit, announced there.
August 1997	Tcl 8.0 introduced a bytecode compiler.
April 1999	Tcl 8.1 introduces full Unicode support and advanced regular expressions.
August 1999	Tcl 8.2 introduces Tcl Extension Architecture (TEA)
August 2000	Tcl Core Team formed, moving Tcl to a more community-oriented development model.
September 2002	Ninth Tcl/Tk conference (Vancouver). Announcement of starkit packaging system. Tcl 8.4.0 released.
December 2007	Tcl 8.5 added new datatypes, a new extension repository, bignums, lambdas.
December 2012	Tcl 8.6 added built-in dynamic object system, TclOO, and stackless evaluation.

Tcl8.6是我们现在使用的版本,发布于2016年7月27号。

3. 三十而立的Tcl 为何依然光芒四射

看点:优点并不是任何场合都适用的,天天和IC、Tcl打交道的你看看我对Tcl 优点的总结。

- 免费

免费,尤其是对商业免费是很重要的,企业对轮子的选择,如果不想自己造,那就要规避产品的法律风险,而免费开源(某些协议)首先消除了企业的这一顾虑,开源更是给了企业更大的自主权,可以利用整合的力量。免费开源对于个人工作中对工具或者学习材料的选择,就理性而言,是需要考虑风险、成本与产出的;任何人的时间和关系都是成本,企业更加关注成本这类关键字。

- 与C 的结合紧密

Tcl 是用C 语言开发的。它现在可运行在Unix, Windows 和Macintosh 等各种平台上。与C/C++ 的紧密结合,互相调用方便。高可用的商业软件很多使用C++ 开发,选择一种脚本语言做为高级语言的辅助工具很重要。从Tcl 中访问C 语言可以在外部为程序临时打补丁;相反,从C 语言中访问Tcl 可以将输入内容传递给内部功能。虽然在用户最终使用场景中,tcl与C你中有我,我中有你的配合不多。但在IC 具体设计调试、工具特性调试这些异常复杂多变的场景中,Tcl 为C提供了类似于在线调试的能力(在线意思为实时互相通信),这充分利用了编译语言的可靠性和脚本语言的灵活性。

- 开发部署周期短

脚本语言是解释性语言,不需要编译,这个特性保证了支持团队的实时反应。如果工程师在设计中遇到一个问题,发给支持团队,可能是EDA 公司,可能是公司内部的CAD 部门,他们需要重新编写一些功能,对代码进行编译、测试,然后再发给到出问题的工程师手里,如果还是不能解决问题,则需要继续迭代,效率会非常低;但如果使用脚本语言,

不触及最核心的高级语言功能代码，只是对外部的东西进行处理，提供一个接口与内部进行沟通，而且脚本的测试和调试也快很多。总结来说就是开发和部署的周期大大缩短，非常适合IC 行业，IC 行业中的核心功能算法不需要快速迭代，AE 或者CAD 工程师面对的是如何将客户的数据或者输入格式调整到工具能够识别的最佳状态，去得到一个最优结果。

- TK 图形界面

Tk (Tool Kit) 是基于 Tcl 的图形程序开发工具箱，是 Tcl 的重要扩展部分。Tk 隐含了许多 C/C++ 程序员需要了解的程序设计细节，可快速地开发基于图形界面的程序。据称，用 Tcl/Tk 开发一个简单的 GUI 应用程序只需几个小时，比用 C/C++ 的效率要高十倍。需要指出的是这里所说的“窗口”是指 Tcl 定义的窗口，与 X-Windows 和 MS Windows 的定义不同，但它可完美地运行在以上两个系统上。

TK 几乎是各种脚本语言都支持的一种图形界面，它的Component binding 等思想到现在都是很好的方法。图形界面是和用户进行交流的一个重要工具，TK 做为一个快速模型，可以很简陋也可以很优雅。在最后一个部分，我将会看到几个 TK 的著名程序，除了使用它们，阅读它们的源代码也会受益匪浅。

- 调试性语言

把Tcl 叫做调试性语言，是因为Tcl 做为一种粘合工具，很容易开发出一套访问核心数据的接口命令，比如EDA 工具的get以及report系列命令。获取信息后配合set_系列命令，快速调试环境，继而配置需要的内容。而Tcl 本身作为程序语言，虽然比不上Lisp 等语言完备，但是应用起来也不成问题，所以『简单』这个特性在自动化测试(Test)和调试(debug)中成了Tcl 的一个优势。

- 整合性语言

Tcl 和C 之间的便捷接口，让它拥有了强大的整合能力。撇开C 语言，用Tcl 做单纯的外部Wrapper 工具也十分便捷，这里指的是利用Tcl 将不同的可执行程序，或者不同语言的脚本整合到一起，Tcl 本身就是为这个而设计的，有一套处理异常的方法和完整的输入输出能力。我习惯给核心程序外加一层命令或者函数去判断其是否出错，对于按行执行的Tcl 来说，必须加一件外衣才能去catch 命令的错误。

- 入门简单，转移关注度

学习成本低，对于商业软件来说价值很大，对IC 工程师尤为重要。一个复杂的语言，学习成本高，迭代周期长会导致迭代成本高，不利于问题的分解，容易让工程师偏离IC 设计的主要矛盾——设计本身的PPA。简单易用是Tcl 发明的最主要动力之一，对于大多数工程师，掌握第一个level的Tcl知识就已够用。

- 扩展性强(package 的优势)

Tcl 的Package 和其他程序语言的库都很像，Tcl 自身也提供了一套如何维护库的命令。Tcl 的库如果不涉及编译的话(比如需要访问数据库)是纯文本，可以说很简陋，当然也可以把这个扩展用得很高大上。由简单的思想建造出优雅很直接，而将一个复杂的结构简化成简单的思想却很难。

- 跨平台(Tcl 本身)

Tcl包括TK 这个界面程序，在保证功能一致性的情况下，无需编译便可以跨平台运行。

- 网络功能(Tcl 本身)

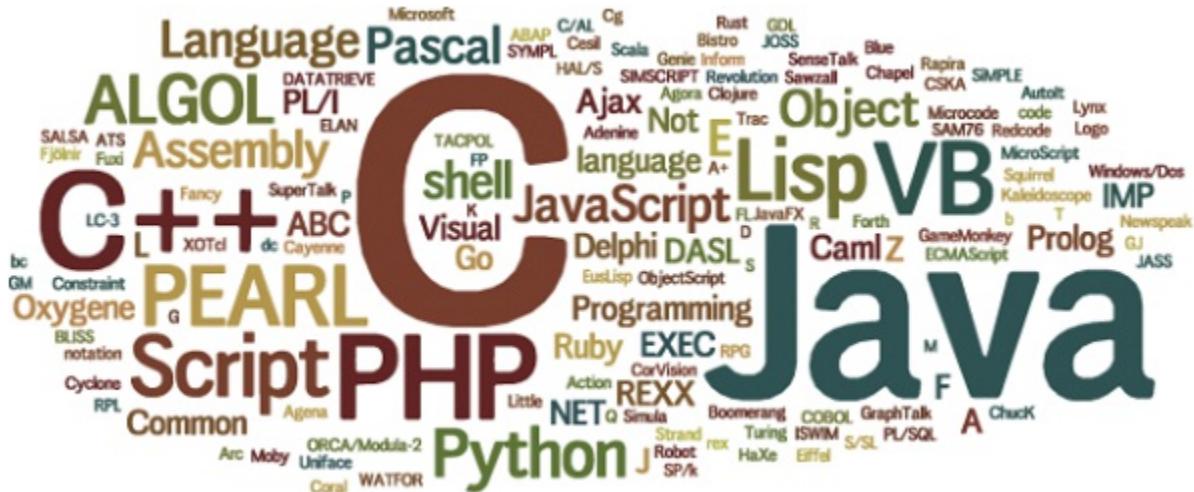
Tcl 通过http 包可以便捷地访问网络，大家可能有用python 或者其他语言做爬虫的经验，Tcl 也是可以的，但是如果用在IC 上，用在系统构建上，这个能力也是不可或缺的。

4. IC 工程师为何要熟悉Tcl 等脚本语言

看点：一点学习Tcl 的体会，即使你厌恶编程，为了提高效率，为了绩效，为了工资，为了面试，你应该看一眼。

前面我列出了十几个问题，涉及了Tcl 的不同方面，仅涉及对程序语言的熟悉，不涉及各种额外的编程思想。

我曾经对物理设计的职位做过词频统计，去除了一些语法词以后，Physical 出现频率最高，Tcl、Perl 的频率也排在前列。



1.3.1.2 - Programming Languages

- Tcl
 - Lisp
 - Scheme
 - SKILL
 - Perl
 - Python
 - Shell

这些Scripting Language 都有各自的背景, Tcl 已经说了很多了, 其产生的原因就是为了解决和电路设计系统的交互问题。而像Perl, Python, 用于处理报告和做数据挖掘、机器学习有它天生的优势。

Lisp 虽然复杂，虽然我也没用过，但是我了解它函数式的特性以及数据和命令的一致性很给力，很多模拟人用的 SKILL 或者 SCHEME，都是 Lisp 的子语言。

我们的工作就是和设计打交道，和工具打交道，脚本语言是设计系统、操作系统开放给我们的交流工具，我们还做不到让系统来理解人类语言，所以要去学习系统能懂的语言。

4.1. 如何学习

如果有编程基础，学习一门编程语言并不难。对于学习Tcl，我分为三个级别：

方法：挑一本Tcl 教程，结合实践从前往后走，就能掌握Level 1 到Level 3。到了Level 3 以后，就会有很多Tcl 脚本的积累。如何积累以及管理这些积累也是一个比较大的话题。



1.3.1.3 - 3 Levels

Level 1: 和EDA工具打交道

- 大概了解Tcl 是如何工作的, 比如前面提到的, 每个命令都是文本, 都需要parsing。
- 会使用语言的基本命令, 变量定义, 条件语句, 循环语句, 异常处理等。
- EDA 工具的命令需要去多看, 多用。

Level 2: 实现一些复杂的功能

- 常用命令的基本用法和Tricky 的用法都要会。
- 会使用包, 熟练使用文件处理和正则表达。
- 理解Tcl 运行的Level 和Domain。

Level 3: 编写界面及复杂的应用

- 会用进程, 以及程序间通信。
- 使用常用的包链接更多功能, 如数据库。
- 制作TK 界面

每个级别并不是只有三个技能, 限于篇幅这里只列举出比较典型的三个。

“有没有捷径可以走”, 悄悄告诉你, “有, 请往下再看两到三章”。

5. Tcl 家族 : Tcl/Tk/Expect

看点: 只用Tcl、Tk、Expect 也可以创造一个自己的IC 黑客帝国。

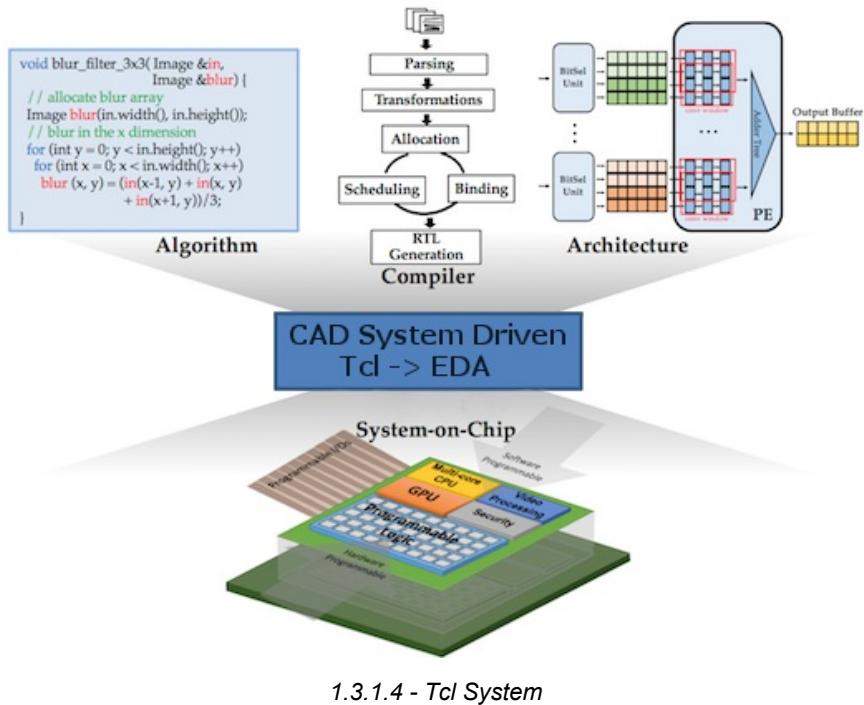
Expect 是Tcl 的一个扩展, 或者说是Tcl 的另外一个可执行的分支, 它的主要作用是用来粘合那些需要有互相交互的程序。

比如你登录某个项目需要输入密码; 比如每次需要输入某个特定的项目名字, 这些需要交互的内容可以用expect 来进行链接, 普通的pipe 只解决输入输出的链接, 对交互没有办法。

你是不是已经跃跃欲试了, 从此可以解放很多冗余操作。

Tk有一个Bwidget 的Package, 所有都是用标准的Tk 组件编写的一个扩展, 好处在于方便修改, 方便升级和维护。

Tcl 用来实现基本的逻辑功能、应用;Expect 用来解决自动化中的交互;Tk 用来编写界面。想造一个自己的系统用Tcl 系的程序语言就可以实现了。



1.3.1.4 - Tcl System

通过学习 $\text{Tcl}/\text{TK}/\text{Expect}$:

1. 更好地与EDA工具交互，扩展EDA的命令集。
2. 可以扩展EDA的界面，虽然没有QT那么华丽，但是基本功能都健全。
3. 可以用Tcl访问数据库(mysql, sqlite3, neo4j)。
4. 可以用Tcl编写各种逻辑功能，完成基本的算法。
5. 更加自动化你的工作，提高效率。
6. 访问网络，和Web协同，数据通信，应用衔接。
7. ...

6. 为什么 Tcl 也可以做系统

- EDA原生支持Tcl，可以有很好地交互和集成。
- 做为系统，界面是一个很重要的交互。
- Tcl有丰富的接口，而且很容易扩展，接口方面和数据库的接口很齐全。
- 方便扩展EDA功能，形成一套自由的体系。
- Tcl的粘合剂作用可以方便的将不同的可执行程序拼接起来。
- Tcl命令和数据基本可以混用，方便数据和命令在同一程序中混合。
- 基于上面一点，命令可以是字符串，是数据也是命令，方便应对各种场景。
- 逻辑层可以方便与数据层分开（软件设计思想方面，Tcl可以应对当下的各种流行场景）。
- ...

7. Tcl 优势的几个实例推荐

有些常用的TK工具，很好用的工具，同时它的源代码也值得一看。由于本章节更偏重实践，这里只做推荐。

7.1. tksqlite

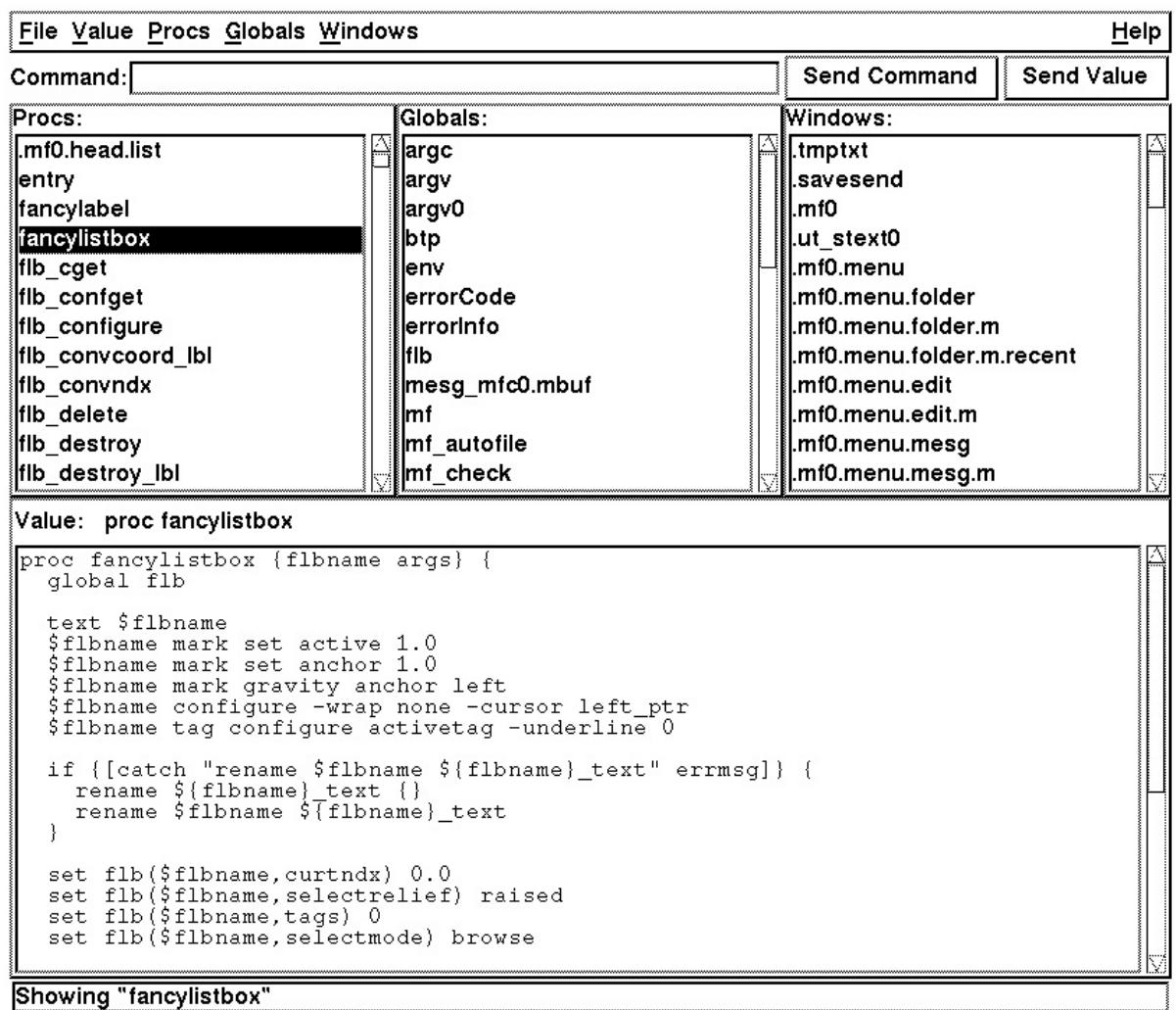
sqlite IDE, 这个是对sqlite 数据编写的一个很好的工具, 是学习数据库访问和处理的一个很好的实例。

The screenshot shows the tksqlite application interface. On the left is a tree view of the database structure under the 'main' schema, including 'Table' (test, test_content, test_semdir, test_segments). The 'test_semdir' table is selected. Below it is a detailed view of the table structure with columns: level, idx, start_block, leaves_end, end_block, and root. The 'idx' column has a value of 1 highlighted. The central part of the interface contains an 'SQL Command' entry field with the query 'select * from test_semdir' and a 'Result' grid displaying the table's data. The 'Command History' section at the bottom shows the same query again. Status information at the bottom indicates 'Version : 3 | Encoding : utf-8 (UTF-8)' and 'Row : 6 line | Time : 1 msec'.

1.3.1.5 - TKSqlite

7.2. tkinspect

TK inspect tool, 这个程序可以分析自己编写的TK 的各个namespace, 各个component, 这个tool 是Tk应用交互的一个很好的实例。



1.3.1.6 - *tkinspect*

8. 文末思考题

Tcl 的好处我也鼓吹那么多了，现在如果你有一个私人Support 团队，CAD 团队为你服务，

你希望有一个什么最能解决你当前问题的Tcl 脚本或应用？大家可以留言讨论或者入群讨论。

这个假设是你有一个强大的Support 队伍，如果没有呢，你如何自己实现？

9. 参考内容

- Steve 笔记
 - Steve 知乎ICCAD 专栏 <https://zhuanlan.zhihu.com/iccad>
- 《Tcl & the Tk Toolkit》
- 《Tcl/Tk : a developer's guide》
- 《Practical Programming in Tcl and Tk》
- http://www.wikiwand.com/en/Scripting_language
- <https://blog.csdn.net/larryliuqing/article/details/20902181>
- <http://scc.qibebt.cas.cn/docs/linux/script/TclTkall.pdf>

终于懂得IP 对于芯片如此珍贵，那么...



本文经「原本」原创认证，访问[yuanben.io](https://yuanben.io/4b34tdul)查询【4B34TDUL】获取授权信息。

The chalkboard contains a dense collection of handwritten mathematical equations and formulas, primarily in black ink, with some red highlights. The equations cover various topics, including:

- Electrostatics: Coulomb's law, electric field, potential, and capacitance.
- Electromagnetism: Ampere's law, Faraday's law, Lenz's law, and Maxwell's equations.
- Quantum mechanics: Schrödinger equation, wave functions, and energy levels.
- Thermodynamics: Ideal gas law, PV=nRT, and related concepts.
- Optics: Ray diagrams and wave properties.
- Relativity: E=mc² and related kinematic formulas.
- Calculus: Integrals, derivatives, and differential equations.
- Algebra: Solving systems of equations and simplifying expressions.

The equations are written in a cursive, fluid style, with many steps and intermediate results visible. The overall layout is a heart shape, with the central area containing the equation $E = mc^2$.

1.4.1.1 - IC 极客

1. 导读

在美国“封杀”中兴通讯事件影响下，国之重器芯片行业受到前所未有的关注，随后阿里巴巴全资收购自主嵌入式CPU IP Core 公司——中天微的消息也一度刷屏。极客群的群友们就此纷纷展开热议，国产IP 的出路在哪里？国产EDA 的出路在哪里？我们能做些什么？是的，“芯”痛不如行动，在IP 和EDA 的交叉领域，沿着大家在群里讨论的思路我希望继续展开谈一下，IP 管理有多重要。

我曾经在不同公司主导或参与过IP 质量检查和IP 管理的流程开发，基于自己的一些经验体会和所思所感，我从以下四个角度来谈一下对这个问题的理解，作为一个开放性的话题，希望可以给读者带来一些启发和探讨。

- IP 为什么重要
- 客户的安全感从哪里来
- IP 管理需要考虑哪些因素
- 延伸思考

2. IP 为什么重要

很多SOC 厂商都依赖IP 来设计和生产一款SOC 芯片，做SOC 的过程其本质就是寻找、验证及整合IP 的过程。如果能够找到满足需求的、质量可靠的、验证过的IP 会极大缩短SOC 的开发周期，这个模式在过去几十年已经非常成熟。

IP 非常重要，对IP 的选择除了质量因素，价格也是重要的考量。随着用户数越多，这个价格会被分摊掉，相对容易承受。

随着芯片复杂度的提高，定制化需求越来越多，IP 产品也有自己的开发和生命周期。是否能够在上市时间和定制化方面更好的满足客户需求，也成为IP 供应商和用户越来越关心的事情。

SOC 对IP 的应用模式随着市场需求、芯片复杂度、上市时间和成本的压力一直在发生变化，当然这也间接导致了行业内IP 的生态发生相应的变化。从SOC 视角看，对IP 的使用可以简单的分为三种情况

- 只为这一个芯片定制的IP，通常发生在企业内部
- IP的选择来自于IP 平台，成熟的hardening IP。有些企业内部有自己的IP 平台，同时也会选择第三方IP 平台的产品。这类IP 的开发就是为了被大量的用户重复使用以降低成本
- IP的选择来自于IP 平台，但在SOC SPEC 阶段就一起规划IP 开发，为满足用户的schedule，其研发周期几乎与SOC 同步

不同的IP 策略选择对于验证和芯片回来后的测试也会有不同的影响。

3. 客户的安全感从哪里来

综上所述，SOC 芯片设计的各个环节都重度依赖于IP 的选择，IP 供应商和客户之间的关系也因此而越来越紧密。但是并非所有的IP 都有良好的质量保证，做过充分的验证，有符合主流EDA 流程需要的完整的数据和清晰的文档。SOC 芯片流片失败有超过40% 的原因都会和IP 有关，譬如非常常见的设计本身错误，产品和需求不匹配，版本用错，有些design for reuse 的IP，会打包发布，这时候容易出现工艺用错，金属层用错。还有datasheet 和IP view 不一致，IP 自身质量不合格或者严格一点说是和SOC 自身的验证流程不匹配等等。SOC 设计师拿到IP 之后还需要再次做质量检查和验证，而SOC 集成时候检查出来的问题或者更严重的IP 本身错误原因导致的流片失败等问题都导致IP 客户的不安全感，以及SOC 厂商和IP 供应商在不得不在彼此依赖的密切合作中形成一种越来越紧张的关系。通常，对于IP 的选择会有如下几个方面的考量

1. IP 选择

通常需要考虑的因素有以下几个方面，成本，成熟度，之前项目的经验等等，PPA 和datasheet 与当前项目需求是否吻合。如果是第一个项目试用，那么IP 的开发和SOC 的开发节奏是否能够契合也是一个很重要的考量。

对于设计服务公司而言，每次在IP选择的这个环节都要大费周章。即便是产品公司，IP的选择也并非一成不变，毕竟市场随时都在变化。

1. IP 的 view

在做选择的同时也不能忽略IP view的完整性和你的SOC设计流程是否匹配。主要看是否满足我们的设计流程以及工具的需求，这其中包括前端、后端、dft等等。当然这是SOC和IP vendor双方妥协磨合的过程，通常问题不大，但出问题的地方也不是一件小事。所以，这也是做设计之前需要考量的一个方面。

1. IP 质量检查

对于一些不太知名的，或者第一次合作的IP厂商（包括企业内部自建IP），在把它应用于SOC设计之前，必要的QA环节是非常重要的，同样重要的就是QA流程的自动化。

1. IP 的版本跟踪

版本跟踪的重要性不言而喻。项目的重要里程碑特别是Tapeout检查，我们需要和IP vendor反复确认版本是否正确。

4. IP 管理需要管什么

前面我们讨论了IP为什么重要以及客户在选择和使用IP时需要考量的因素。对于IP，需要管理的事情之琐碎，信息量之庞大已经远远超过了人工所能负荷的范畴，我们需要结合实际的应用场景寻找一种更加简单、优雅、更低成本的做法。

首先，需要先看IP管理的使用者是谁，可能是SOC厂商或者IP供应商，也可能是SOC设计师或IP设计师，或者将他们基于需求连接，在这简单的几句话里隐藏了大量的应用场景，他们分别需要怎样的信息和数据以及如何使用这些信息和数据。从来源看无论是为一个芯片定制的IP，还是IP平台，IP的来源都分为企业内部和第三方。从IP生命周期的视角看，企业对内部IP的管理，分研发管理和应用管理，对第三方IP的管理主要需要满足信息化和与SOC开发协同。从IP开发流程的视角看，IP分digital, analog, mix-signal。这些都是做IP管理可以考虑的切入点。

第二，IP的质量检查和一致性。质量检查最通用的做法对于SOC工程师而言就是基于你已有的流程搭建测试平台，重跑一遍DRC, LVS, ERC, and antenna，并且利用现有的EDA工具的命令或API针对Verilog, LIB, LEF, and GDS等你关心的view做pin to pin完整性和一致性分析。如果要做的比较细的话，可以深挖出更多的需求。对于IP交付而言，除内建质量检查外，同时还需要考虑满足不同客户，不同EDA工具和企业流程他们所需的数据格式不同，这些数据之间的完整性和一致性问题。此外，还有一些客户需要在系统设计和验证以及post-silicon之后有一些额外的需求，也是IP交付管理中需要考量的内容。

第三，SOC与IP协同开发平台，需求管理，文件管理，信息同步管理。如何满足点到点的对接，如何利用这些链接产生新的价值，从而提高整体研发平台的设计效率和自动化。如何在整个过程中积累和分享整体运作的知识和方法。

5. 思考几个问题

- 如果你是SOC厂商，在选择IP的策略上有哪些考虑
- 如果你是IP供应商，如何管理你的产品和连接你的用户，以及他们在不同时间段的不同需求
- 如果你是SOC设计师，在使用IP之前会去重复检查质量和做验证吗，分别使用什么方法和策略
- 如果你是IP设计师，如何做交付质量管理和持续交付

6. 给自家IC极客群打个广告

本群由IC 行业的几位工程师发起，以公益，开源，分享为宗旨，致力于推广IC 极客文化，组织大家深入交流IC 设计领域知识，经验及方法学，打造IC 设计圈的思想国。群内交流的方式为，每周二，五定期推送固定主题的干货文章，发起相关主题的群内头脑风暴和技术分享。也欢迎群友或IC 极客玩家随机发起不固定主题的讨论。欢迎联系文末的微信号小主人群参与分享交流。在未来一周内，我会在IC 极客群内陆续分享一些之前做过的IP 质量检查和IP 管理流程开发经验，同时预告下周五的主题“IC 设计中的数据管理，版本管理及配置管理”，如果你身边有对此类话题感兴趣的朋友，不妨邀请他入群一起参与讨论或围观。另外，你们心心念念的DFT 专场也会在近期开场，如果您希望参与开讲或讨论老样子联系文末的微信号小主。



1.4.1.2 - 入群

“刷脚本”的学术讨论

DNA 5R9GIIKQ
CC BY SA

本文经「原本」原创认证，访问[yuanben.io](https://yuanben.io/5r9giiq)查询【5R9GIIKQ】获取授权信息。

作为一个经历多年IC Tapeout 后转做的IC CAD 从业者，我想聊一聊“刷脚本”这件事，把它上升到理论的高度。

1. “刷脚本” = “使用流程 (Flow)”

流程是指一个或一系列、连续有规律的行动，这些行动以确定的方式发生或执行，导致特定结果的实现。

国际标准化组织ISO9000认证认为流程是一种将输入转化为输出的相互关联和相互作用的活动。



1.5.1.1 - Flow

我比较赞同这段话的原因在于它诠释了几个关键的点，流程的作用是把事情做出来，IC 就是完成一个个SPEC 要求的内容：为功能，为生产完成 Closure 的目标。其次后半句揭示了我最赞同的部分，相互联系 (Connection)，单一的功能可以很强大，可以很深入，但是没有了联系，就不能产生流动，没有价值的传播和再造。

所以我的观点是，比如IC 实现方面，单一的Logic Synthesis 技术提升有没有意义呢？是有很大意义，但是能和具体功能结合，后端物理实现结合的技术能发挥的更多。

解读完我引用的一段话，说说“刷脚本”，话分两类人，新手很少提及这个词，看上去没有技术含量，对于新手来说，需要的是究其（流程）内涵；而对于有经验的人来说，一个流程“刷”下去，胸有成竹，心里清楚整个过程和原理，那么工作慢慢就变成了“刷脚本”。

上次讲了Tcl，那么这次我把自己对流程相关的学习和体会，做个分享。

2. 建立 IC Flow 的意义

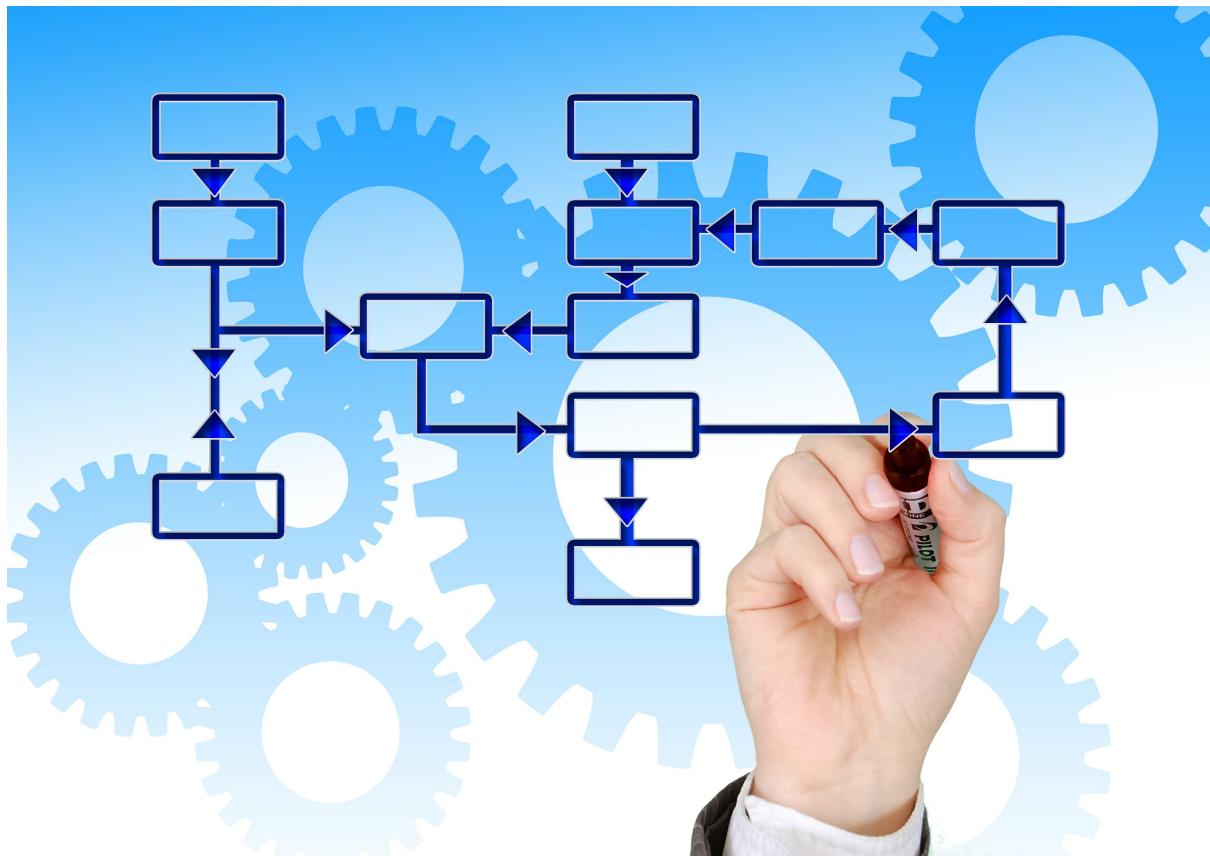
流程更多的是一个管理方面的概念，对于IC 各大公司都有基于自己工具的Flow，而我通过阅读思考得到的是脱离了工具的普适的流程管理在IC 上的理解。当你觉得遇到瓶颈的时候，无论什么方面，这个时候更应该去学习一些理论知识，并试着做些理论到工程的转换，当然如果能做为一种习惯更好。关于流程的意义这一部分，我觉得各个文章和书讲得都很好，我收藏的引文1，它表述的观点，我觉得放到IC 的流程管理上一点也没有什么违和感，我借用它作为小标题。

- 建立关系，呈现逻辑

这个是显而易见的，IC Flow 需要很强的逻辑性和前后依赖关系。很多时候工程师看到一个脚本都会去问为什么这么做，然后可以牵出一堆写该脚本的理论基础，有点基于工艺生产，有的基于经验，有的基于法务等等。

【体会】做为工程师如何看待Flow，Flow 是必须要用的（公司规定），保证了质量和效率（后面会提到该好处），有人说会弱化工程师的能力，是有一点这方面的问题，久而久之看上去你是个“刷脚本”的人，但从大的方面看，这个流程是基于通用的IC 工程实现做起来的，从定SPEC 到写算法，到仿真和实现以及各类检查。我以前觉得没办法有一个IC 全局观，其实从Flow 里去学习总结，很容易获得IC 全局观。

Flow 建立了关系，呈现了逻辑，我每次使用，都着重理清关系，读懂逻辑，这个更多的我叫他软技能；对单点技术，比如CTS 怎么做，这是一个精致知识点的工作，是个人硬技能的体现。对IC 硬技能培养专家，软技能培养管理（技术管理我也归为此类）。



1.5.1.2 - Connection

- 细分工作, 明确职责

流程的切分带来了流水线, 将IC这个复杂的过程分为若干步, 随着规模的越来越大, 这个也是大势所趋。从我熟悉的IC实现上, 小的设计一个人可以做整个IC实现, 没有太多知识和Schedule上的压力, 而大的设计, 可以分得很细, 综合一拨人, Floorplan一拨人, 写约束一拨人, CTS一拨人, 解Route Congestion一拨人, 做Timing一拨人, 物理验证一拨人, 和Foundary打交道的一拨人...

上面说了流程除了告诉你怎么做, 也会有对输入和输出的规定, 就是互相的Connection, 这个过程如果定义地很清楚, 那么细分工作下的权责分明, 质量保证都蕴含在Flow里了, 消除扯皮。

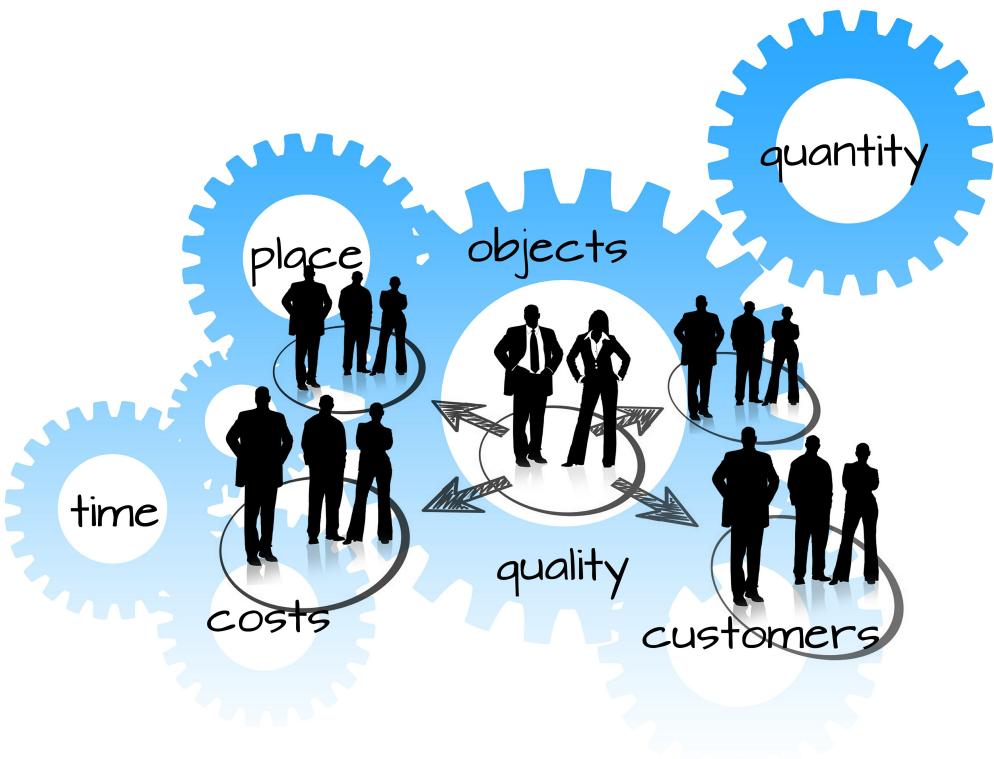
【体会】从自己这一步的流程出发, 假定你的输入是完全有问题的, 并且假定你需要给出一个完全没有问题的输出; 做到上述程度后, 去了解你的输入是怎么产生的, 你的输出需要给怎样的流程用(在自己硬技能保证的前提下)。

- 精简工作, 提高效率

对于一个芯片工程, 无论是行内人还是外人, 我觉得都会认同这是一个很复杂很复杂的“沙子的逆袭之路”。好的流程有更多的自动化, 更多的质量保证检查, 更多的管理理念来共同推动效率的提高。流程和流程管理工具本身就可以看成是一个软件, 简单来看软件是对数据的信息化, 批量化处理。

【体会】实现流程化管理, 可以把常规的、重复的、固定的职能固化, 这样管理者将会有更多时间做其它事情。使用一个好的流程, 先学会它, 然后抛弃它(将流程纳入自己的知识体系), 把更多的精力放在设计本身。

- 规范管理, 精益生产



1.5.1.3 - Logistics

对于IC 实现, 我们的Flow 不是简单的Synthesis + Floorplan + Place + CTS + Route + STA + Physical Verification。还包含数据的管理, 数据的交付等。

【体会】精益生产, 对于IC 我能看到的是很多经验知识被不断迭代到Flow 中去, Flow 成为了一个沉淀知识的容器, 可以帮助、提醒IC 工程师, 及时完成流程中的所有环节。说了这么多是不是有兴趣再去好好理解一下你用的Flow 呢?

3. IC Flow Reengineering (IC 流程重建)

IC 流程我们可以看作是生产芯片的一个过程, 加入了很多方法学, 半导体知识, EDA 的使用, 以及个人的经验, 可以说不是一个简单的物件构造过程, 可以说是一个标准化下的创造过程。

计算机技术的发展, 人工智能的出现都在不断地影响着我们的流程构建, 虽然重建流程成本是巨大的, 但是IC 工艺越来越复杂, 新的计算技术越来越成熟, 让我们明白在传统 IC 制造基础上建立的流程需要不断适应新的变化。

Why-How-What 是一种思考方式, 而我要表达的是你能看到什么, 进而说明我们为什么要这么做以及怎么做。

3.1. What

IC 流程重建根据新的技术, 新的需求, 对IC Flow 进行升级改造甚至是全盘重来。很多团队不会去做一次很大的重建, 更多的是小规模的升级, 但是有时候重建又是迫在眉睫的。

所以一个好的迁移方案: 流程融合+渐进式替代, 回事很多团队愿意去接受和尝试的, 也是比较正确的一个做法。

3.2. Why

为什么要重建？

- 适应新的技术
- 提高团队能力
- 提高输出质量

主要动力应该差不多就是这几个，流程的上一层就是流程管理框架，一个好的流程管理框架可以帮助你适应新的技术，沉淀知识经验，提高输出质量。

3.3. How

研究自己的流程，选择合适的管理框架，在不影响现有Tapeout 的基础上，在框架内对流程进行单点的提升和替换。同时要预留旧系统的继续服役期和新系统的上线适应期。

4. IC 流程管理中的点

从流程的定义来看，我们关注的是当前这个点的输入，当前点本身，和当前点的输出。对于IC 过程来说：

- IC 生产资料管理
 - 库，技术文件等
- IC 生成过程管理
 - 人员，流程等
- IC 生产成果管理
 - 交付等



1.5.1.4 - Business

5. 参考资料

1. <http://articles.e-works.net.cn/bpm/article129044.htm>
2. 《流程再造》
3. 《流程的永恒之道》

图片均来自网络免费可商用图库，仅做为配合主题的装饰。

6. 给自家IC 极客群打个广告

本群由IC 行业的几位工程师发起，以公益，开源，分享为宗旨，致力于推广IC 极客文化，组织大家深入交流IC 设计领域知识，经验及方法学，打造IC 设计圈的思想国。群内交流的方式为，每周二，五定期推送固定主题的干货文章，发起相关主题的群内头脑风暴和技术分享。也欢迎群友或IC 极客玩家随机发起不固定主题的讨论。欢迎联系文末的微信号小主入群参与分享交流。未来两周为dft 后端专场，如果你身边有对此类话题感兴趣的朋友，不妨邀请他入群一起参与讨论或围观。另外，你们心心念念的DFT 专场也会在近期开场，如果您希望参与开讲或讨论老样子联系文末的微信号小主。



1.5.1.5 - 入群

実践

To be added.

Tcl Seminar: 学习Tcl (Day 1)

我在文章中也讲到三个级别，对于一个基本工具而言，虽然没有什么技术含量，但又不得不去学习的东西，大家掌握的程度就有点千差万别。

1. 初学者

有人建议去学别人的总结，比如XX大牛的文档，这也不失为一个好的方法，快速的方法。

当初我也是从一本24小时系列学起的，这种系列能解决你的语法问题，对这类书的定位是解决你的hello word 问题，无法给你各种事物之间的联系。如果英文看了犯困，可以先看中文，在初步实践以后再把英文版看一遍。

对于工具的Command 手册，要随手看，就像背单词一样，时不时去翻翻，各大公司的用户网站上也会有很多内容，时不时拿些下来消化。

1.1. 练习

1. 阅读英文版 Tcl & Tk Toolkit 前N 章。
2. 用Tcl 数组构建一个数据集，先记录员工的信息，然后编写一个简单的抽奖程序，每次多个人中奖，如何保证你一定在其中？
3. 用Tcl 完成一个房贷还款计算器，输入是房子价值，贷款比例，年限，借贷人收入等等，判断能不能给借贷人批贷款，如果可以每月本金利息各占多少，给出完整的列表。

2. 非初级

对于Tcl，或许你已经运用很熟了，我们可以讨论一个话题，有人提到trace 很有用。的确我们可以做很多事情用trace。

2.1. 练习

1. 利用trace 编写一个对用户来说是readonly 的变量。
2. 利用trace 跟踪用户，如果退出Tclsh(exit)，自动清理当前目录内的tmp 文件夹。

Tcl Seminar: 学习Tcl (Day 2)

从Day 2开始不分初学者与非初学者了，两者差异仅在24小时的阅读中。

练习都是有点tricky 的问题，但是仔细想想总有在IC 中的有用武之地。

通过练习，引起兴趣，并看书查资料，此过程中掌握80%，另外20% 靠思考。

1. 练习

1. 如果有三种阈值电压的cell(BVT>AVT>CVT)，按D, driven strength来看，D越大，驱动能力越大，而不同的VT下，约定数字越大肯定驱动能力越大(为了题目做的简化约定)，给下列cell 按驱动能力排序 BUFAVTD1 BUFAVTD2 BUFAVTD4 BUFAVTD10 BUFBVTD2 BUFBVTD4 BUFBVTD10 BUFCVTD2 BUFCVTD4 BUFBVTD10，使用lsort 的command 选项，不能编写一个含了各种for 的脚本。
2. 隐藏工具的内部命令，用自己的命令替代完成相同的功能，并加入自己的一些个性需求。

Tcl Seminar: 学习Tcl (Day 3)

使用Packge 是一个很好的习惯;正则表达, 匹配是我们日常处理中一个很大的工作。

1. 正则表达式到哪里都是神器, 你有什么方法跨行匹配?
2. 我最近有个需求, 每次用package 不知道这个package 被load 以后是来自哪里的? 我对自己编写的package 想了个办法, 不一定是最好的, 你有什么办法吗?

Tcl Seminar: 学习Tcl (Day 4)

今天来看看 Tcl 对几个数据库的接口, Tcl 的扩展能力很强, 所以基本上编写一个数据库接口对于熟悉 C 的程序员来说并不复杂。请尝试熟悉使用连接以下数据库:

1. Mysql
2. Sqlite3
3. neo4j

推荐随便翻翻 SQL 的知识, 比如《Sql入门经典》, 虽然现在 No-SQL 也很流行, 但是并不见得 SQL 没有优势。

Tcl Seminar: 学习Tcl (Day 5)

对于基础知识, 如果恶补的话, 第五天应该补得差不多了, 今天为止, 我们还没有提到GUI, 资料在这里。

- 学习TKinspect 这个程序 <http://tkcon.sourceforge.net/tkinspect/>

1. Tkinspect Introduction

tkinspect is an inspector for Tk applications. It uses Tk's send command to retrieve information from other Tk applications. When you choose an application through the File>Select Interpreter menu, lists will be filled with the names of the procs, globals, and windows the the application. Clicking on one of those names will fill the value window with the definition of the proc, the value of the global variable, or various information about the window. The value window is editable and its contents can be sent back to the selected application. Tcl commands can be sent to the selected application using the Command: entry, or through a command line (via the File/New Command Line menu) interface to the application. The command line is nearly identical to Tk's demo rmt.

For further information, select a topic from the help window's Topics menu or from the list below:

- [Tkinspect Introduction](#)
- [The Value Window](#)
- [The Lists](#)
 - [The Procs List](#)
 - [The Globals List](#)
 - [The Windows List](#)
 - [The Images List](#)
 - [The Canvases List](#)
 - [The Menus List](#)
- [Miscellany](#): how tkinspect starts up, how to customize, etc.
- [Notes](#): notes about tkinspect
- [Whats New](#): what's new in this version.
- [Tkinspect ChangeLog](#): a detailed list of changes.