北京大学
PEKING UNIVERSITY

基于AI的数字系统设计

第一讲：课程介绍与基于AI的数字系统设计简介

主讲：陶耀宇

2026年秋季

课程简介

- 培养学生初步把握AI时代的硬件芯片的工作原理、设计原理与未来发展方向

指标	课程信息
课程号	04835810
学分	3
课程性质	专业任选
地址	二教401
优秀率	无强制限制
考核方式	**平时成绩（30%，实验报告3次，每次 10%）、期中大作业（30%）、期末大项目（40%）**

扫描二维码：加入【人工智能硬件基石】群添加说明：年级-姓名

前置知识要求：无强制先修要求、建议具备最初步编程能力

编程技能：简单Python、Verilog（助教将通过习题课逐步进行教学）

推荐教科书：

物理器件/逻辑电路方面：

- Digital Integrated Circuits: A Design Perspective - Anantha Chandrakasan
- CMOS数字集成电路：分析与设计 - 康松默

智能计算架构方面：

- Computer Architecture: A Quantitative Approach - John L. Hennessy
- 智能计算系统 - 陈云霁
- 人工智能芯片设计 - 尹首一

课程团队-主讲教师

陶耀宇

北京大学集成电路学院助理教授、博士生导师

国家优青（海外）

博雅青年学者

华为未名青年学者

经历	学校或单位	学习或任职
教育经历	上海交通大学	电子与计算机工程 学士
	美国[illegible]大学	电子工程 硕士
	美国[illegible]根大学	电子与计算机工程 博士
工作经历	美国德州仪器公司	Kilby实验室 研究实习员
	美国[illegible]文公司	VLSI实验室 电路工程师
	[illegible]高通公司	无线研发部 主任研究科学家（2019-202[illegible]）
	北京大学	人工智能研究院 副研究员（2022-2025）
	北京大学	集成电路学院 助理教授（2026-至今）

思想自由 兼容并包

研究领域	长期从事**后摩尔新器件芯片架构与集成电路**研究，研究成果获**人民日报、新华社、科技日报、TechXplore、MSN、Science News、Yahoo**等海内外权威媒体完[illegible]报道，成果入选**2025中国半导体年度十大进展**候选
研究成果	发表[illegible]0余篇（**一作/通讯30余篇**），**一作/通讯论文包括：[illegible]、Nature Electronics、Nature Communications，集成电路与芯片架构设计顶级期刊/会议IEEE JSSC、IEDM、VLSI、MICRO、HPCA等，电路系统设计顶级期刊/会议IEEE TCAS-I、FPGA、Globecom等，作为主[illegible]发明人申请/获批数十项中、美专利**
奖励荣誉	2025中国半导体年度十大进展候选、2025 Chip中国[illegible]十大进展候选、**2025年华为未名青年学者奖**、2024中国电子学会青年年会[illegible]题报告奖、**2024年华为计算产品线最佳技术合作奖**、2023年Wiley Open Science Excellent Author、2022年北京智源人工智能青年科学家、**2021年世界通信大会（IEEE Globecom）最佳论文奖、2019~2021年连续3[illegible]通技术明星奖**（Qualcomm QualStar Awards）、2019 IEEE VLSI最佳论文奖提名、2018年Rackham科学研究STG奖、2013 IEEE ISCAS最佳论文奖提名等荣誉
主持项目	**国家高层次青年人才项目、国家自然科学基金[illegible]大项目、科技部国家重点研发计划项目（课题负责人）、国家自然科学基金委重点项目（课题负责人）、科技部"智能电网"重大专项（子课题负责人）、科技部重点研发计划"战略性创新合作"项目**、北京市自然基金项目、华为计算产品线技术合作项目、华为数通产品线技术合作项目、纵慧芯光半导体有限公司技术合作项目、武汉东湖区-北京大学技术合作项目等

课程内容概要

• 课程简介

- 当前，数字集成电路与系统的设计复杂度呈指数级上升，传统数字电路与系统设计是基于硬件描述语言（HDL）的手工设计，正面临着开发周期长、人力成本高昂，以及功耗/性能/面积（PPA）难以实现全局最优的严峻挑战。在人工智能（AI）时代，随着深度学习、强化学习以及大语言模型的飞速发展，为数字集成电路与系统设计领域带来了颠覆性的变革。
- 本课程在原有的《基于HDL的数字系统设计》课程基础上，**全面引入AI的方法进行数字系统设计，致力于打破传统数字电路设计与日新月异的AI算法间的壁垒，聚焦于"AI for Digital Circuits and System Design”（人工智能赋能的数字电路与系统设计）以及“利用AI优化系统架构”的前沿交叉领域**。学生将系统掌握如何将先进的人工智能算法应用于数字电路的设计、验证及系统级优化中。

助教：张坤
信息科学技术学院本科4年级
主要负责CLAB服务器平台、编程Lab等

助教：钟林峰
集成电路学院博士1年级
主要负责课程网站、作业批改等

助教：王翌鸣
信息科学技术学院本科3年级
主要负责CLAB服务器平台、作业批改等

课程内容概要

- **课程目的**

- 本课程旨在应对摩尔定律极限下，芯片复杂度指数级增长与传统基于HDL的手工设计难以兼顾PPA最优闭环、开发周期长、迭代速度慢的严峻挑战。**课程聚焦基于AI的数字前端设计，致力于打破学科壁垒，培养既精通逻辑设计与时序分析，又具备“AI for Hardware”思维的交叉型人才。**
- 课程紧紧跟前沿，**培养学生利用机器学习模型生成从简单到复杂的数字集成电路与系统设计代码，推动从自然语言需求到硬件逻辑的自动化转化，并进行参数寻优、拥塞预测、数据流编排、功耗优化等，提升硬件性能、缩短迭代周期。学生将在实战中解决真实数字电路与架构设计中的PPA优化难题**，树立产学研结合的工程实践观，成为掌握AI赋能工具链的新一代数字前端设计专家。

课程内容概要

- 课程目的

  - **第一章：AI算法基础与工具链**
  - **第1周：AI与IC数字设计导论**
  - 核心内容：从手工设计到AI协同的设计范式演变，AI工具概览
  - **第2周：AI算法基础（硬件视角）**
  - 核心内容：监督学习与回归模型，神经网络与大模型基础，回归在延迟/功耗预测中的应用
  - **第3周：Prompt Engineering for HDL**
  - 核心内容：高效编写Verilog/SystemVerilog的提示工程，SystemVerilog规范与LLM交互逻辑

  - **第二章：数字前端设计与验证**
  - **第4周：AI辅助组合逻辑设计**
  - 核心内容：复杂逻辑功能的描述与AI生成的代码优化，LLM上下文理解与逻辑综合预测
  - **第5周：AI辅助时序与状态机设计**
  - 核心内容：有限状态机(FSM)的自动化生成与规范检查，状态空间描述与逻辑错误预测
  - **第6周：AI驱动的功能验证（上）**
  - 核心内容：AI辅助测试向量(Testbench)的生成与覆盖率分析，生成式模型在激励产生中的应用

课程内容概要

- **课程目的**

- **第7周：AI驱动的功能验证（下）**
- 核心内容：AI辅助调试(Debug)：异常模式检测与代码审查，异常检测(Anomaly Detection)算法
- **第8周：课程中期项目演示与复盘**
- 核心内容：基于AI的前端设计流程实践（AI芯片中的脉动阵列及其状态机设计）

- **第三章：数字系统架构设计**
- **第9周：数字体系结构基础（上）**
- 核心内容：指令集架构、顺序流水线、冯诺依曼结构基础知识
- **第10周：数字体系结构基础（下）**
- 核心内容：乱序执行流水线、缓存基础知识、[illegible]基础知识
- **第11周：架构设计空间探索(DSE)基础（多级流水线为例）**
- 核心内容：性能(P)、功耗(P)、面积(A)权衡分析，多目标优化理论基础
- **第12周：基于机器学习的DSE（多级流水线为例）**
- 核心内容：利用回归模型预测架构参数对PPA的影响，梯度提升树(GBDT)在性能建模中的应用

课程内容概要

- **课程目的**

  - **第13周：缓存与存储子系统调优**
  - 核心内容：利用数据分析进行Cache一致性与容量规划，聚类分析在工作负载分析中的应用
  - **第14周：复杂系统级NoC集成设计**
  - 核心内容：AI辅助复杂片上网络(NoC)的拓扑与参数寻优，强化学习(RL)在路由算法中的初步应用

- **第四阶段：基于AI的数字系统前沿发展与展望**
  - **第15周：课程总结与技术演进**
  - 核心内容：AI辅助芯片设计的发展趋势与职业规划，大模型AI模型的发展对数字电路与系统优化的展望
  - **第16周：期末大项目汇报**
  - 核心内容：复杂数字系统设计及架构寻优报告，全流程AI工具应用展示

课程内容概要

- **教学方式**

- **课堂讲授，3次实验（组合逻辑、时序逻辑/状态机、AI辅助验证）+1次期中项目（AI芯片中的脉动阵列及其状态机设计）+1次期末项目（复杂系统级NoC集成设计），重点训练学生不再手动进行数字电路设计，而是熟练使用AI工具进行数字电路设计与优化**
- **设计中不调整参数，而是利用AI算法（如强化学习、AI大模型）等自动生成数字电路、体系架构并进行设计空间寻优（DSE），让学生深刻理解并学习基于AI的数字系统设计流程。**

- 平时成绩（30%）：实验报告（3次，每次10%）。
- 期中大作业（30%）：基于AI的前端设计与验证流程（AI芯片中的脉动阵列及其状态机设计，考察代码质量、仿真完整性、时序约束合理性、设计报告规范性）。
- 期末大项目（40%）：复杂数字系统设计及NoC架构寻优报告、全流程AI工具应用展示（基于AI的Verilog代码编写、时序分析、综合约束、架构参数寻优等）

课程实验环境

- **培养学生初步掌握智能时代的硬件芯片的工作原理、设计原理与未来发展方向**
  - 分别从**逻辑电路与计算架构2方面**进行全栈式介绍
  - 项目采用CLAB平台：https://clab.pku.edu.cn/

    具体使用方式请参考： CLAB使用手册
  - 实验采用Linux环境进行开发
    - 所需软件环境已为各位同学安装好，**不需自己配置环境**
    - Linux运行Lab的说明请参考：Linux使用参考信息
  - 助教正在为各位同学建立CLAB账号，具体事宜请同学们联系助教
  - 如有任何问题，欢迎联系授课老师或助教！

智能芯片的计算能力是未来新的生产力

- 数据是新的生产资料，计算能力是新的生产力，是支撑科技发展的源动力

古代

基于人力的算力

算盘

18世纪以前

机械化时代

基于机械的算力

机械式计算机

18世纪-20世纪初

电气化时代

基于电气元件的算力

电子管计算机

20世纪上半叶

电子信息时代

基于集成电路的算力

晶体管计算机

1960 – 2030

大数据智能时代

云边端多样化算力

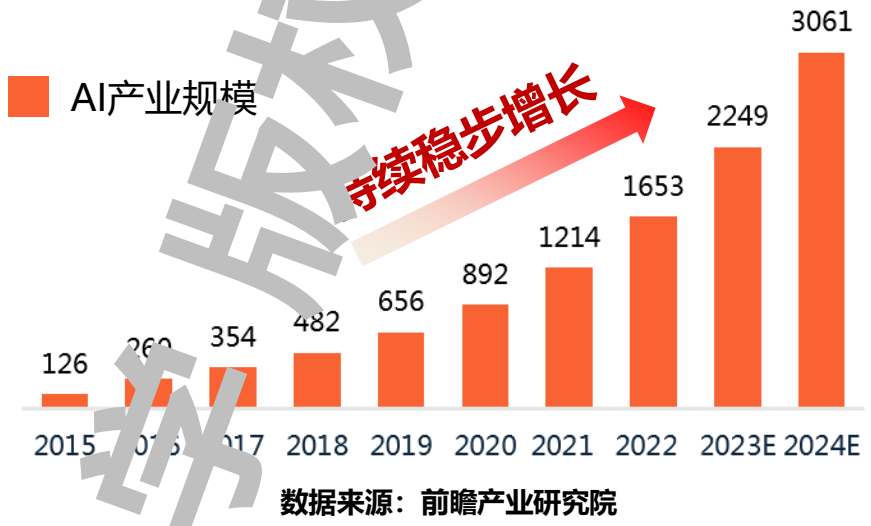
新原理、新架构计算机

2030以后

兼容并包

人工智能产业蓬勃发展

- 人工智能产业是推动我国未来新质生产力发展和经济转型升级的核心驱动力，且持续增长



国内外大模型相关产业现状

OpenAI

以OpenAI为代表的AI大模型公司持续发布GPT系列模型，包括GPT-2/3/4、GPT-o1、Sora等，复杂推理能力大幅提升，并整合进微软多款产品中，是全球领先的大模型研发公司

Google DeepMind

自2018年期发布BERT、AlphaGo、LAMDA、PaLM、AlphaFold系列AI大模型，领域覆盖科学计算、语言、视觉等，并最新推出多模态大模型Gemini

Meta AI

Llama系列开源大模型，在文本分析、视觉任务等多领域表现优异，Llama Lite等轻量化大模型用于边缘/端侧应用

HUAWEI

发布盘古大模型系列，面向视觉、自然语言、科学计算等多个场景；提出五大基础大模型，细分为多个行业大模型，深度渗透政务、金融、制造、气象、医学等多个行业，提升智能化水平

文心一言 Baidu 百度

通义千问 阿里巴巴

腾讯混元 Tencent 腾讯

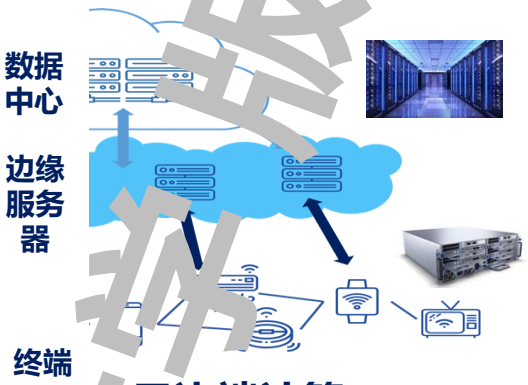
DeepSeek deepseek

人工智能产业蓬勃发展

- 提升全社会、全产业智能化水平，助力产业颠覆性发展，服务国家重大战略需求

以AI大模型为代表的先进智能技术

云边端计算



到 2025 年，全球 云计算系统产业 规模将突破 约 5.6 万亿元

到 2025 年，全球 边缘计算产业 规模将达到 约 1130 亿元

视频安防、智能手机、物联网、工业机器人等典型应用

AR/VR

到 2025 年，AR/VR 产业规模将达 约 1740 亿元

自动驾驶/无人机

到 2025 年，自动驾驶计算系统产业规模将达 3588 亿元

自动驾驶计算系统

环境感知	CNN/RNN
地图定位	SLAM/GRU
运动规划	RL/LSTM
控制决策	RL/LSTM

语音/图像

到 2025 年，智能语音/图像市场规模将达 约 3612 亿元

国家重大需求

航空航天、空间探索、国防工业等场景

海量工业制造场景

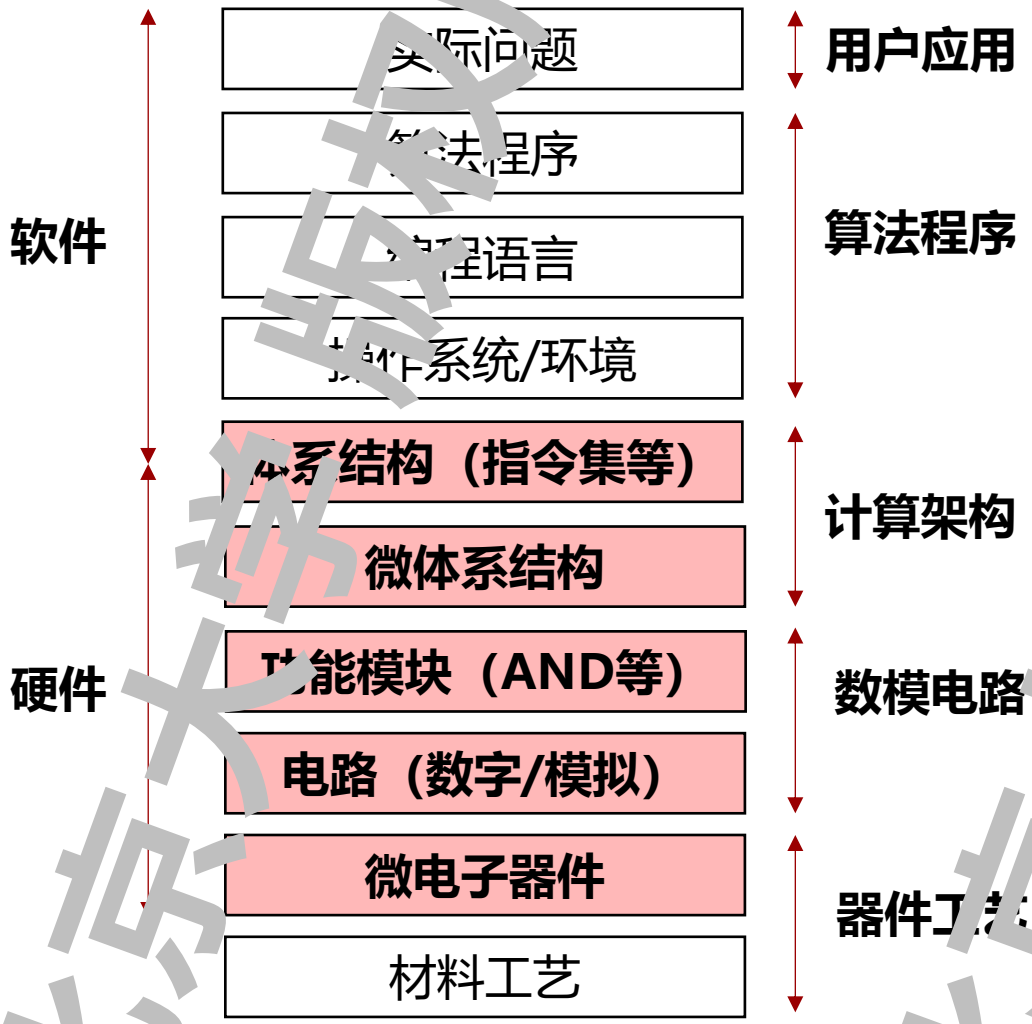
赋能电力、智能制造等工业场景，提升制造业自动化、智能化水平，产生数量级级别的生产效率提升

社会治理/经济管理

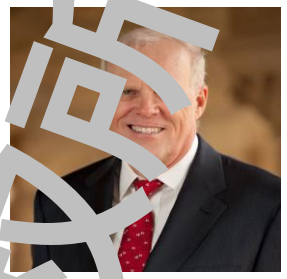
利用大模型提升社会治理水平和经济运行效率

人工智能硬件基础：芯片

- 学习人工智能硬件芯片是如何工作的至关重要

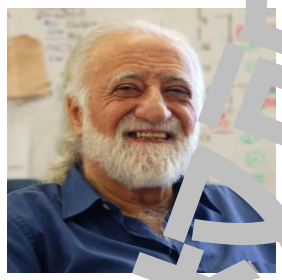


硬件发展的历史关键人物



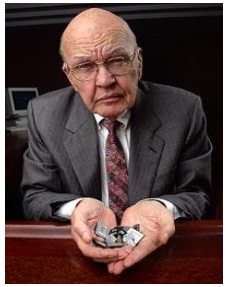
John Hennessy
Stanford
美国科学院/工程院院士
图灵奖获得者

David Patterson
UCLA/UC Berkeley
美国科学院/工程院院士
图灵奖获得者



Yale Patt
UMich、UT Austin
美国[illegible]
富兰克林奖章得者

Robert Noyce
Intel创始人
美国国家技术奖章



Jack Kilby
集成电路发明人
诺贝奖获得者

Gordon Moore
美国工程院院士
美国总统自由勋章

胡正明 UC Berkeley
美国工程院院士、FinFet发明人
美国国家技术奖章

William Shockley、Walter Brattain、John Bardeen
[illegible]晶体管的3位发明人
均为美国科学院院士、诺贝奖获得者

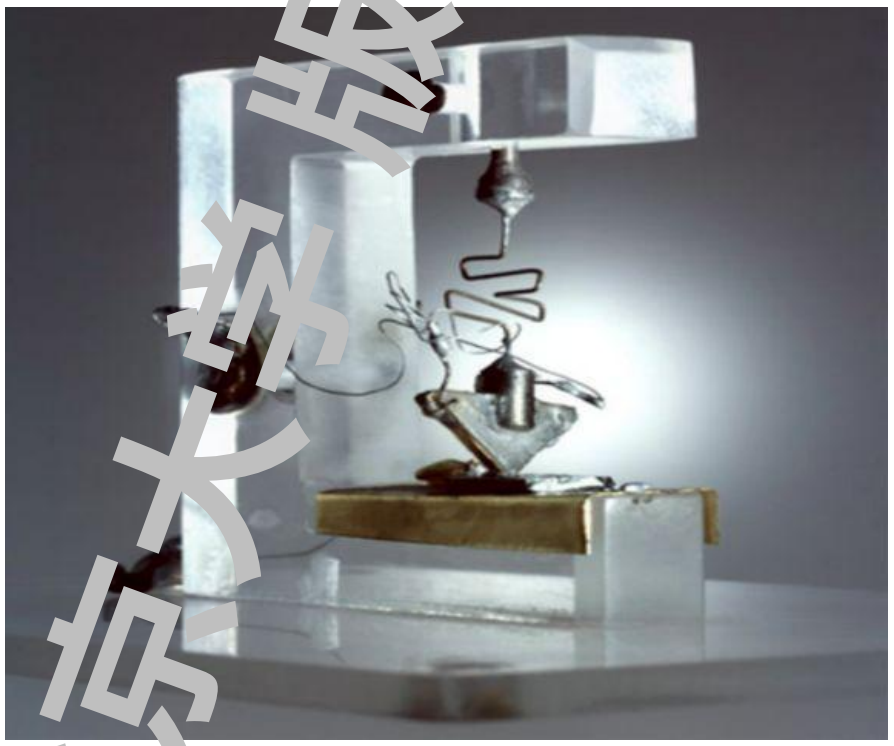
人工智能硬件芯片——数模电路

- 学习人工智能硬件芯片是如何工作的至关重要

半导体锗晶体管的发明 – 1947年

- 半导体晶体管被誉为“21世纪最伟大的发明”，深刻地改变了人类历史发展进程

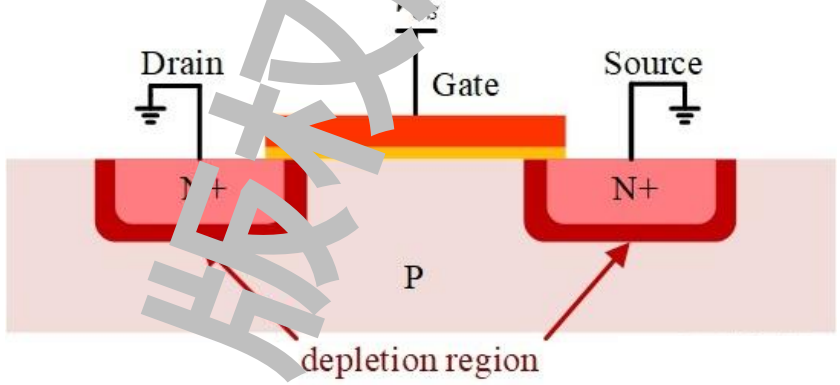
肖克利（前）、巴丁（后一）、布拉顿（后二），半导体锗晶体管的发明，共同获得了1956年的诺贝尔物理学奖



点接触式晶体管，把间距为50 μm的两个金电极压在锗半导体上，微小的电信号由一个金电极（发射极）进入锗半导体（基极），并被显著放大，然后通过另一个金电极（集电极）输出，这个器件在1kHz的增益为4.5

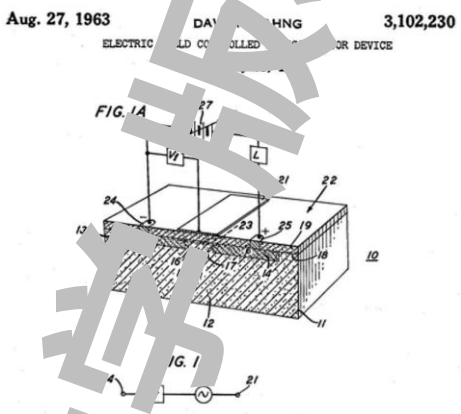
硅基MOSFET是实现现代芯片的器件基石

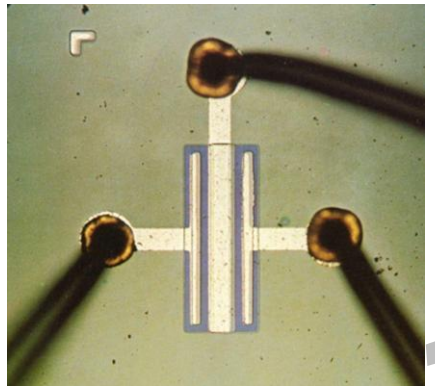
- 艾塔拉 (Martin Atalla) 和姜大元 (Dawon Kahng) 共同发明了硅基MOSFET场效应晶体管



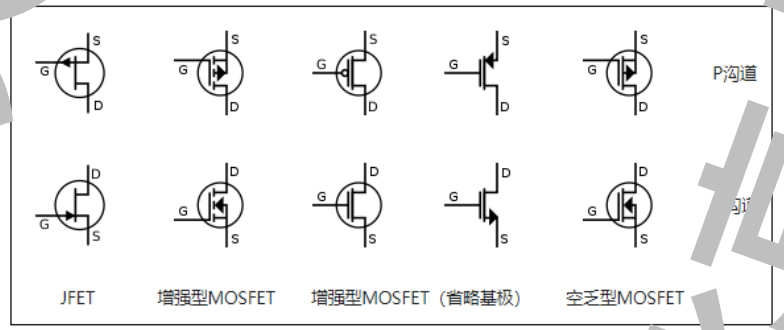
艾塔拉 (Martin Atalla) 和姜大元 (Dawon Kahng)

Metal-Oxide-Semiconductor Field-Effect Transistor (MOSFET)





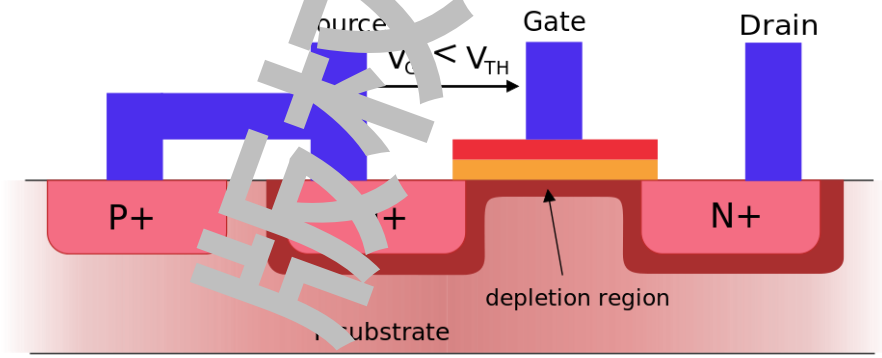
PMOS场效应晶体管实物图

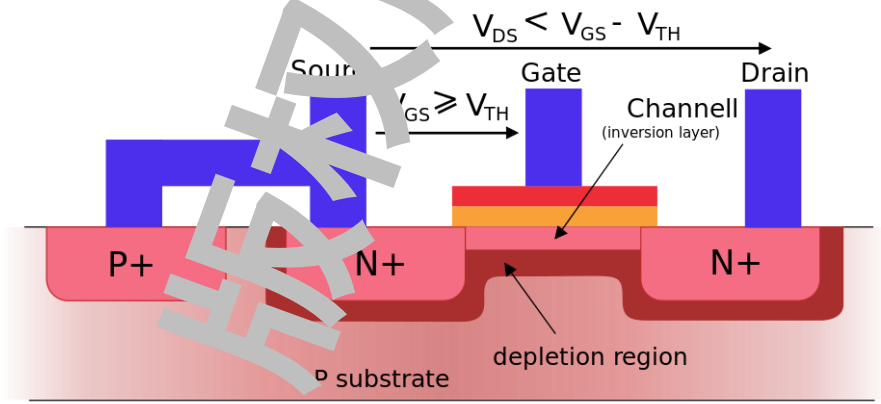


MOSFET已经成为集成电路的基本组成单元

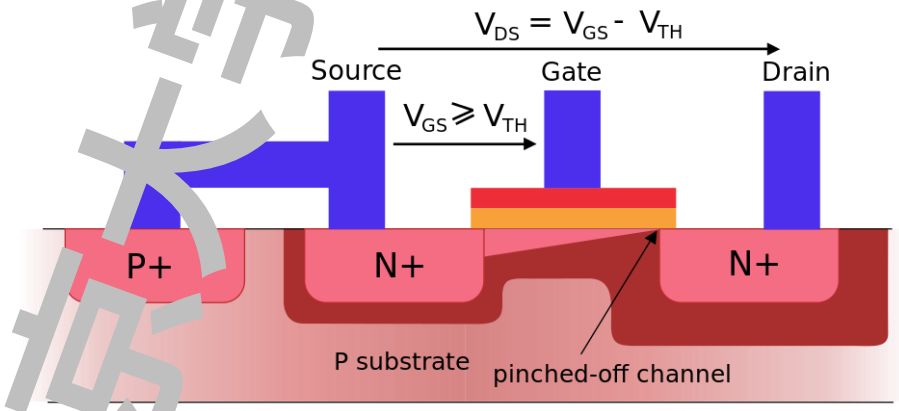
MOSFET晶体管工作原理 – 可控开关

- MOSFET有三个工作区间：断开、线性（欧姆区间）、饱和（电压不随电流线性增加）

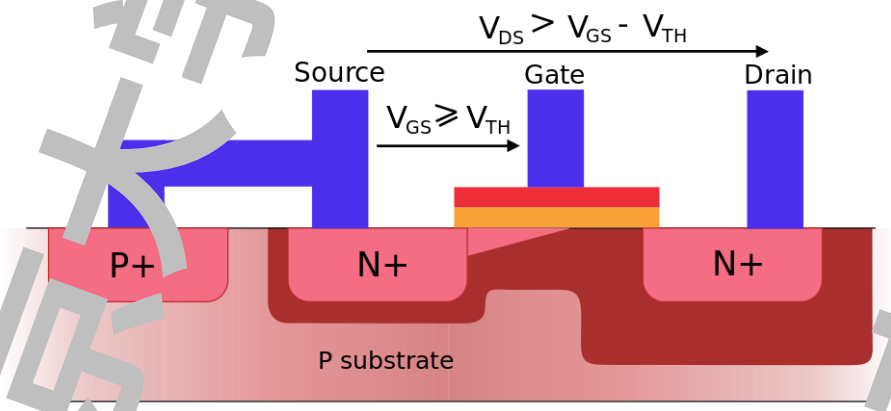




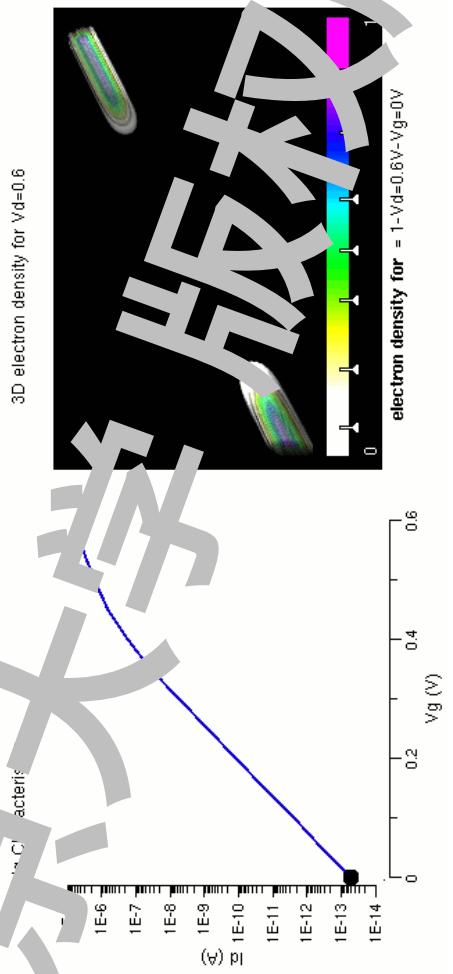
Linear operating region (ohmic mode)



Saturation mode at point of pinch-off



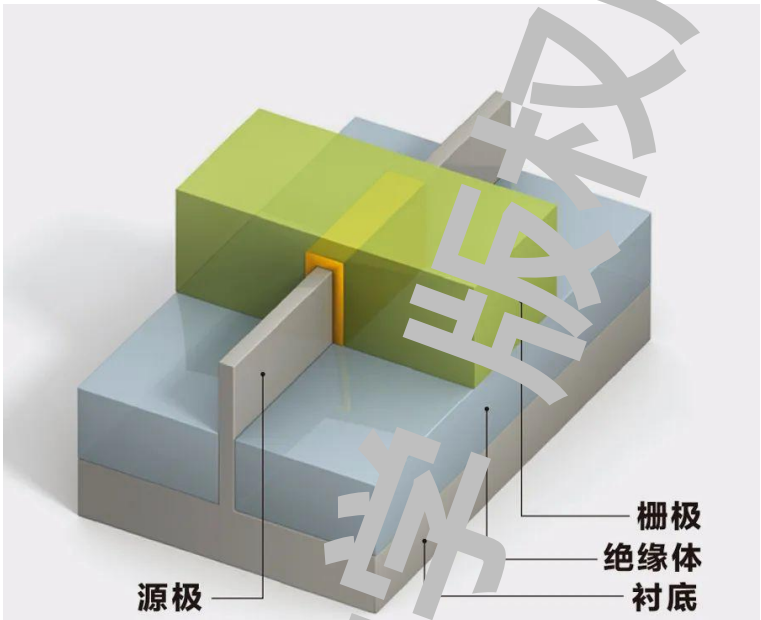
Saturation mode



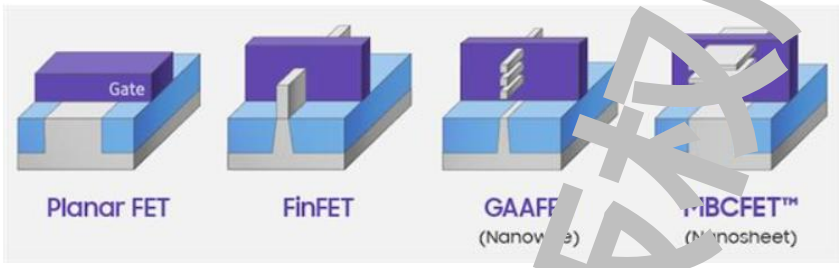
鳍式三维晶体管FinFET – 1999年

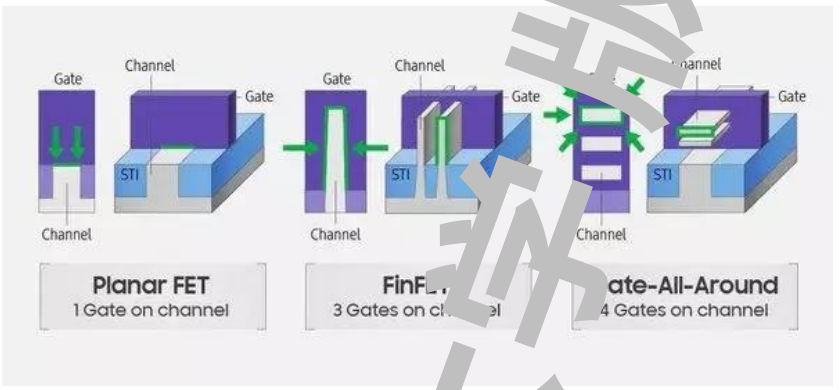
- 原本预计2010年后传统MOSFET在20nm走到尽头，胡正明的发明进一步推进制程缩小

加州大学伯克利分校的**胡正明教授**
（IEEE Fellow，美国工程院院士，
中国科学院外籍院士）



FinFET的特点：一个晶体管包含一个源极、一个漏极、一个连接两者的导电通道和一个控制电流沿通道流动的栅极。在FinFET中，鳍通道抬高，如同鲨鱼鳍高出芯片表面，栅极三面环绕，从而给予栅极更大的控制力。





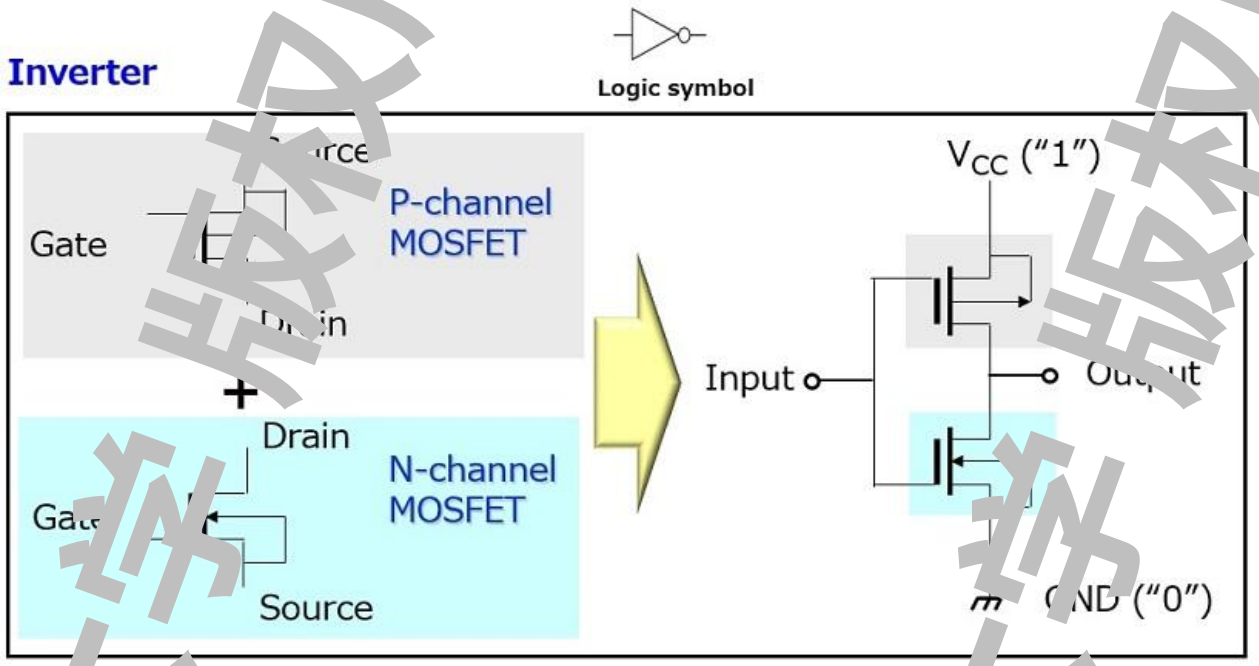
由FinFET演化出多种三维晶体管构型，推动制程向3nm/1nm演进

人工智能硬件芯片——逻辑电路

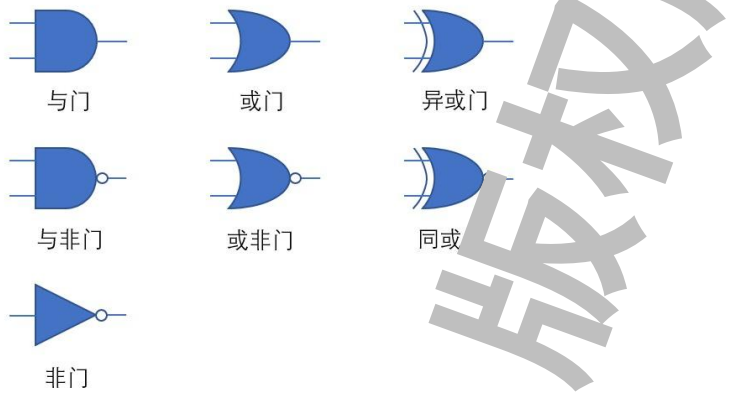
- 学习人工智能硬件芯片是如何工作的至关重要

CMOS数字逻辑电路 – 当前芯片的主流电路技术方案

- 仙童半导体于1963年首次发明互补金属氧化物半导体（Complementary Metal Oxide Semi...）



CMOS非门电路图



互补式金属氧化物半导体只有在晶体管需要切换启动与关闭时才需消耗能量的优点，因此非常节省电力且发热量少，且工艺上也是最基础而广泛使用的半导体器件

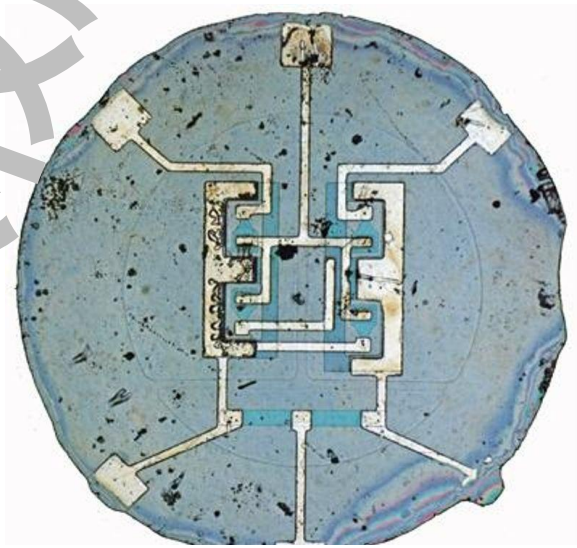
硅质晶圆模板上制出NMOS（n-type MOSFET）和PMOS（p-type MOSFET）的基本器件，由于NMOS与PMOS在物理特性上为互补性，因此被称为CMOS

重要历史节点：集成电路的发明 – 1958年/1959年

- 德州仪器公司的工程师基尔比 (Jack Kilby) 发明了第一块集成电路

195[illegible]年8月28日世界第一块集成电路
尺寸7/16×1/16英寸

基尔比获2000年
诺贝尔奖



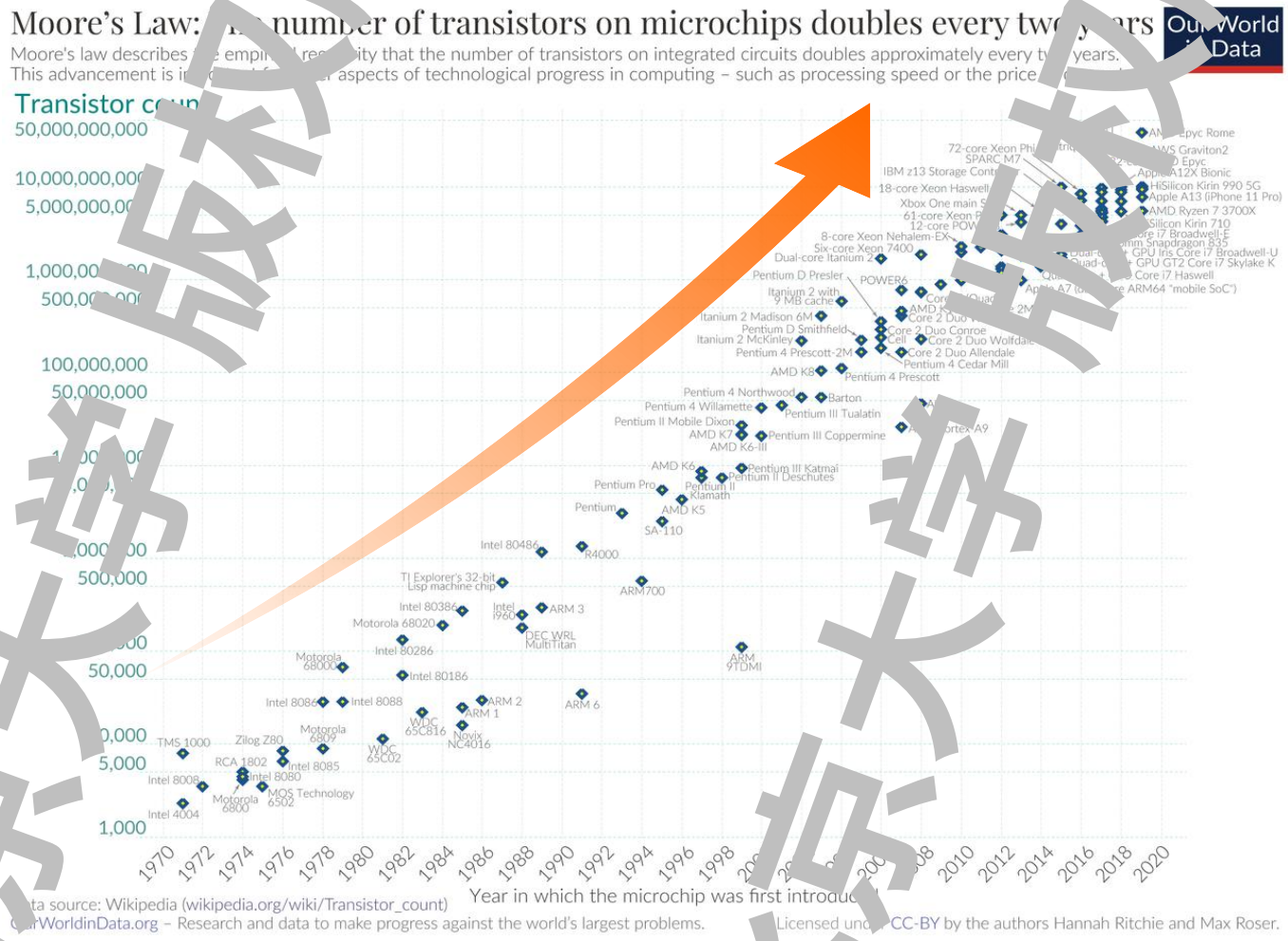
罗伯特-诺伊斯于1959年8月发明第一块
硅集成电路

将包括锗晶体管在内的五个元器件集成在一起，基于锗材料制作了一个叫做相移振荡器的简易集成电路

参与创立仙童半导体 (Fairchild) 和英特尔 (Intel) 公司，奠定了硅谷的基石

重要历史节点：摩尔定律的提出 – 1964年

- 仙童半导体/英特尔的联合创始人戈登摩尔提出了著名的“摩尔定律”



戈登·摩尔

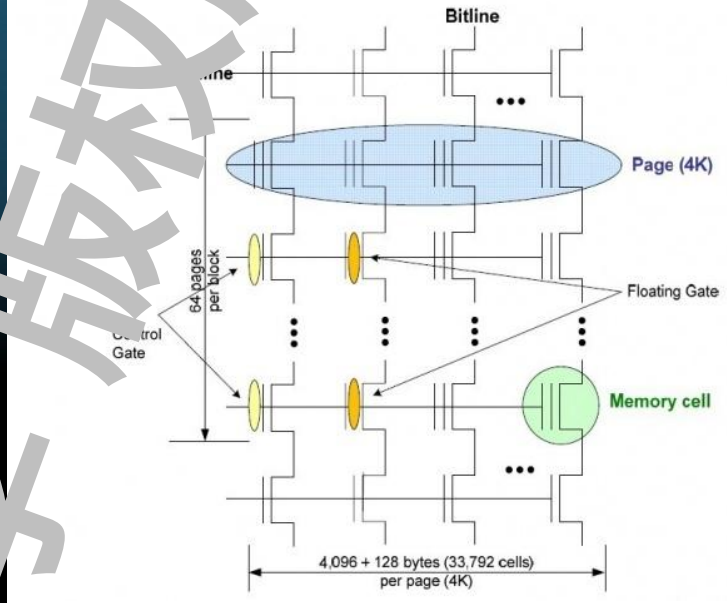
集成电路上可容纳的晶体管数目，

每隔两年便会增加一倍

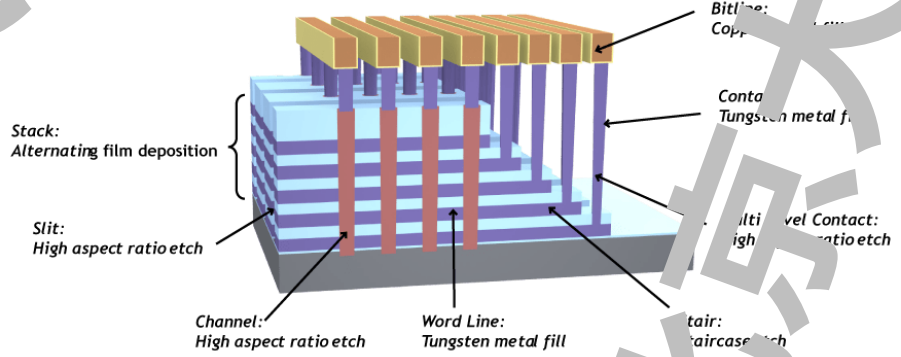
非易失性存储器Flash的发明 – 1967年

- 除了计算场景之外，存储也是占据智能芯片重要份额的典型应用场景

Dawon Kahng (韩) 和 Simon Sze (华裔) 在贝尔实验室发明了非易失性存储器浮动门（Floating Gate），并发表为“A Floating Gate and Its Application to Memory Devices”（贝尔系统技术期刊）



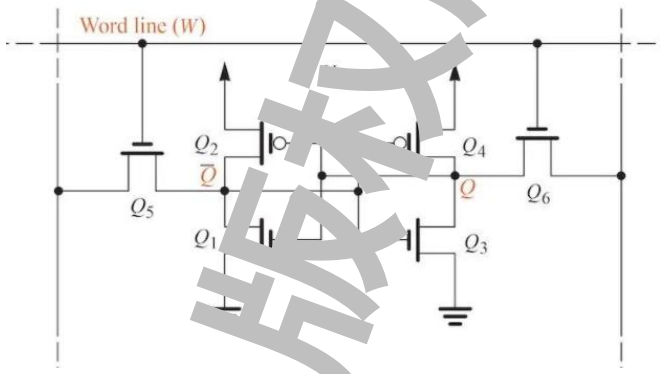
传统平面NAND Flash非易失性存储器



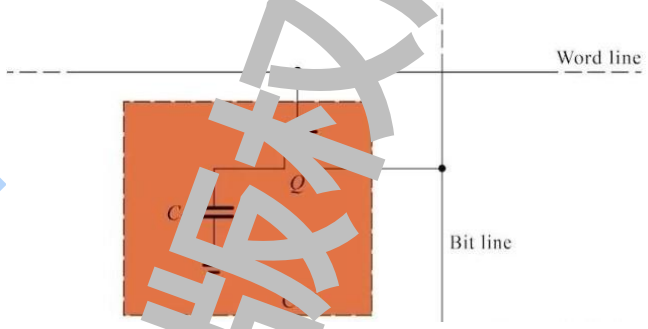
现代三维堆叠NAND Flash

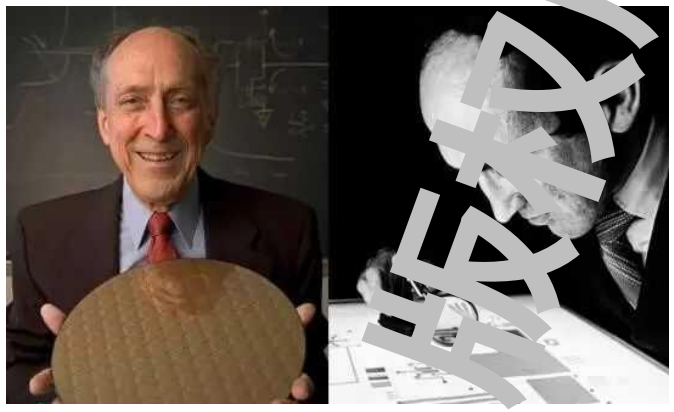
易失性存储器DRAM的发明 – 1968年

- SRAM/DRAM是两种最常用的易失性存储器件，广泛应用于现代智能芯片中



SRAM需要6个CMOS晶体管来存储数据





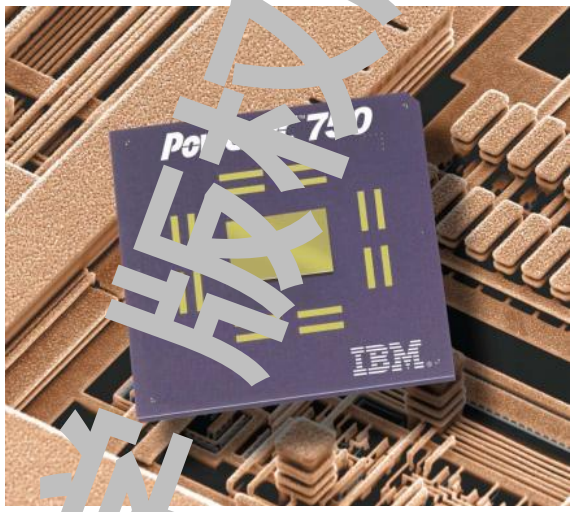
罗伯特·丹纳德发明了DRAM (动态随机存取存储器) 存储器

SRAM（静态随机存取存储器）的优点是它的速度快，它的存取速度比DRAM（动态随机存取存储器）快得多，因为它不需要每次访问数据都要重新刷新电容。

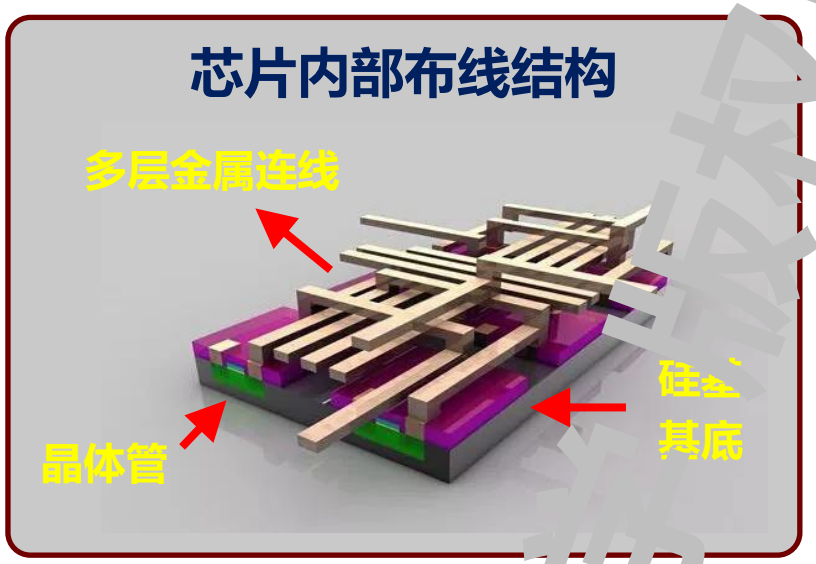
与SRAM相比，DRAM的优势在于结构简单—每比特都只需一个电容加一个晶体管来处理，相比之下在SRAM上一个比特通常需要六个晶体管。 正因这缘故，DRAM拥有非常高的密度，单位体积的容量较高因此成本较低。 但相反的，DRAM也有访问速度较慢，耗电量较大的缺点。

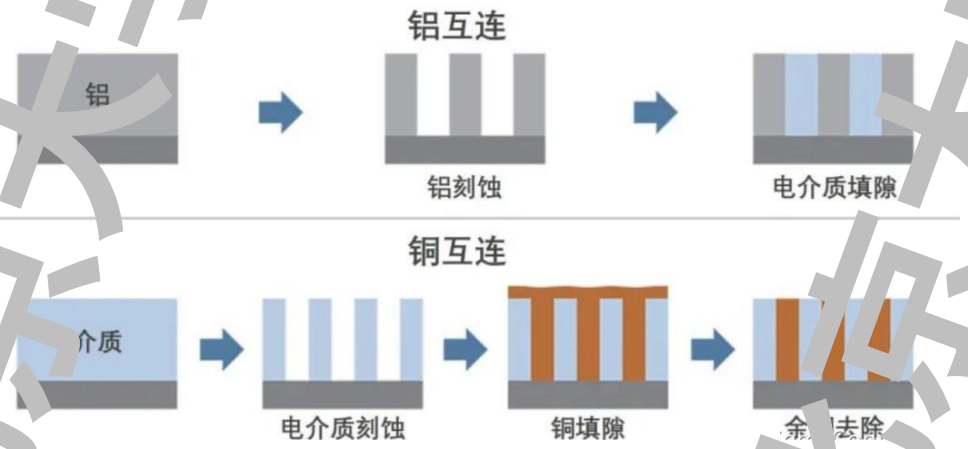
电路间的互连方式：智能芯片的铜互连技术 – 19[illegible]年

- IBM率先从铝互连转向铜互连，并推出了第一个铜基微处理器 IBM PowerPC 750



IBM PowerPC 750 最初是采用铝设计的，其工作频率高达300 MHz。采用铜互连之后，同一芯片的速度至少能达到400MHz，提高了33%





集成电路金属互连线制造工艺达到纳米级后，因为超高纯铜具有更佳的电阻率和抗电迁徙能力，很快高纯铜替代超高纯铝合金成为金属互连线的主要材料

人工智能硬件芯片 计算架构

- 学习人工智能硬件芯片是如何工作的至关重要

为什么要学习硬件计算架构?

- 硬件计算架构是为了解决上世纪60年代出现的实际工程问题 – 如何链接多种算法与单一硬件?

IBM Compatibility Problem in Early 1960s

By early 1960's, *IBM had 4 incompatible lines of computers!*

701 ➡ 7090

650 ➡ 7074

702 ➡ 7080

140 ➡ 7010

Each system had its own:

- Instruction set architecture (ISA)
- I/O system and Secondary Storage: magnetic tapes, drums and disks
- Assemblers, compilers, libraries,...
- Market niche: business, scientific, real time, ...

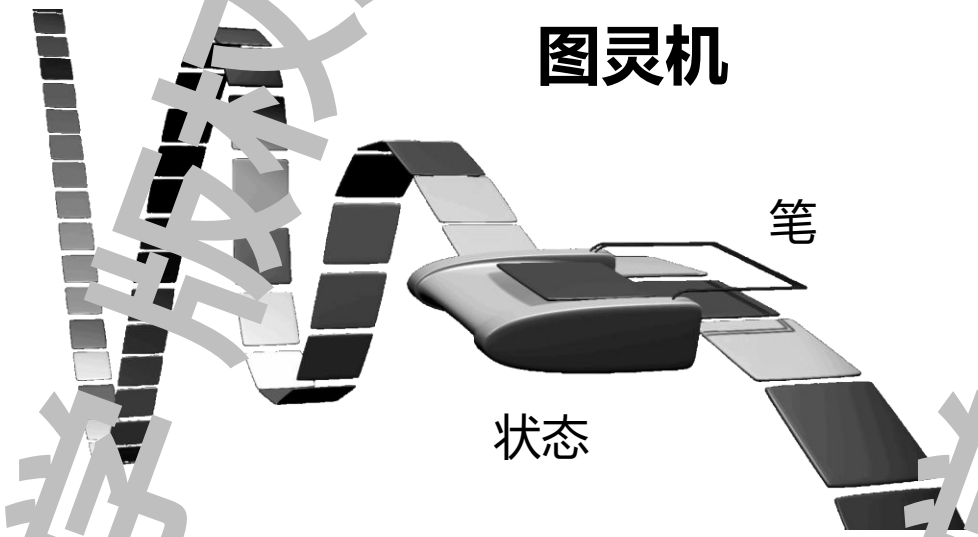
Stanford

海量不同的计算任务

同一块硬件

将计算任务分解为图灵机可运行的离散化操作

- 图灵计算理论作出以图灵机为理论支撑的现代智能计算体系结构



运算法则

纸带符号	状态A			状态B		
	写	移动	状态	写	移动	状态
0	1	右	B	1	左	A
1	1	左	A	0	右	B

模拟人们用纸笔进行数学运算的过程

- **纸带：**一条无限长的纸带（TAPE），被划分为一个接一个的小格子，每个格子上包含一个来自有限字母表的符号
- **笔：**一个读写头（HEAD），可以在纸带上左右移动，能读出当前所指的格子上的符号，并能通过写操作改变它
- **运算法则：**一套规则（TABLE），根据当前状态及当前读写头所指格子上的符号来确定读写头下一步的动作
- **状态：**一个状态寄存器（STATE），保存图灵机当前的状态

图灵计算理论

- 图灵机的数学理论框架由一个七元有序组定义

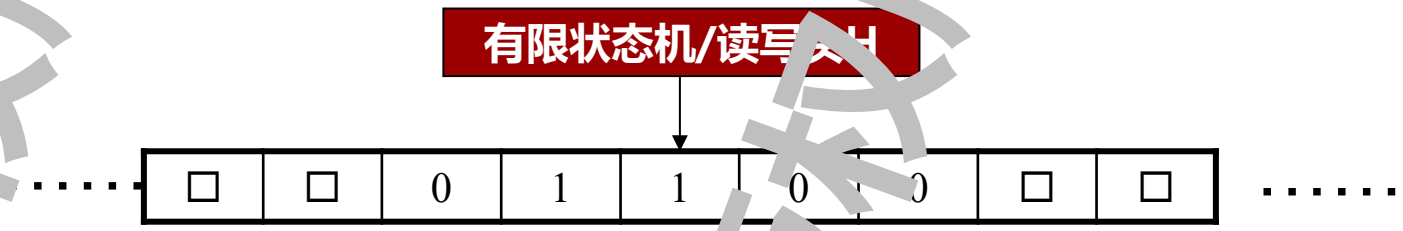
一台图灵机可被定义为 $T = \{Q, \Sigma, \Gamma, q_{10}, q_{accept}, q_{reject}, \delta(q,s)\}$

- **Q**：是非空有限状态集合
- **Σ**：非空有限输入符号表，其中特殊空白符 $\square \notin \Sigma$
- **Γ**：非空有限带符号且 $\Sigma \subset \Gamma$，空白符 $\square \in \Gamma - \Sigma$，也是唯一允许出现无限次的字符
- $q_0 \in Q$ 表示图灵机起始状态
- $q_{accept} \in Q$ 表示接受状态
- $q_{reject} \in Q$ 表示拒绝状态, 且$q_{reject} \neq q_{accept}$
- **δ(q,s):** $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ 是转移函数， 根据当前读入符号s和当前状态q决定下一个状态、写入的符号、纸带移动方向和距离， L, R表示读写头是向左移还是向右移，- 表示不移动

图灵计算理论

- 图灵机的计算方式与工作流程

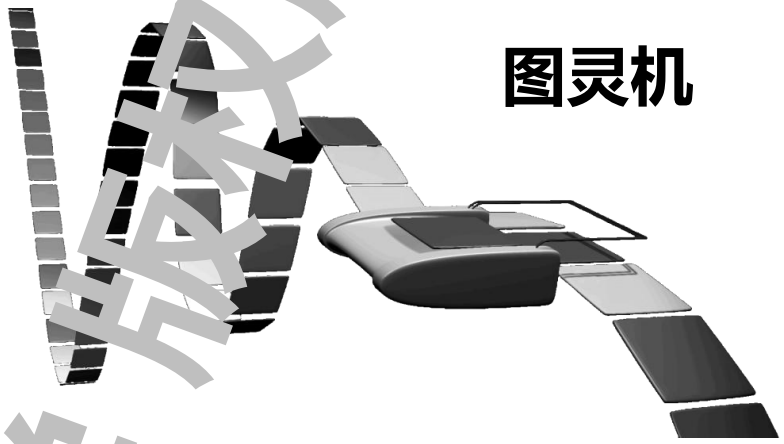
$T = \{Q, \Sigma, \Gamma, q_0, q_{accept}, q_{reject}, \delta(q,s)\}$



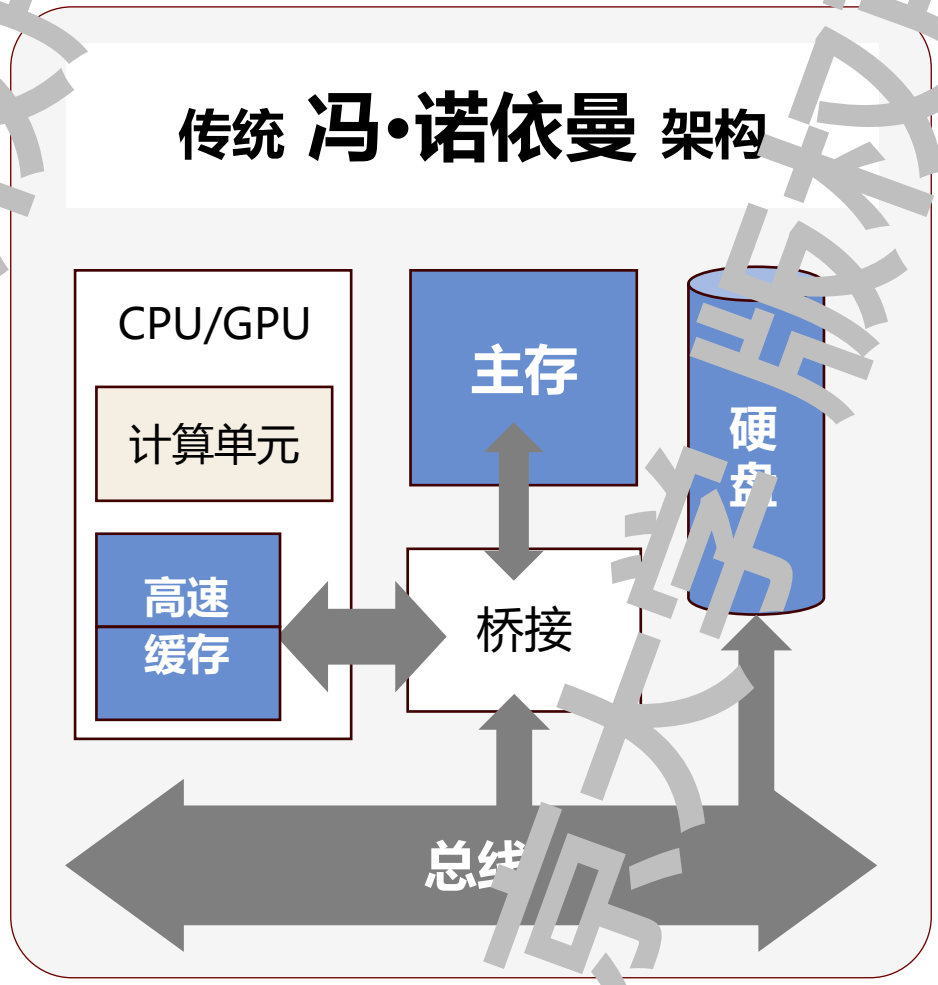
- **初始状态**：将输入符号串 $\omega = \omega_0\omega_1 \ldots \omega_{n-1}$ 左对齐 $\Longrightarrow$ 纸带第0,1, ..., n-1 号格子
  - 读写头H指向0号格子，T @ q_0状态
- **运行方式**：T按照转移函数所描述的规则运行计算
  - T@q状态，H = x，设 $\delta(q,x) = (q', x', L)$
    - T→q'，H→x'，读写头左移一格
    - 若某时刻H指向0号格子，但根据 $\delta(q,x)$ 将继续左移，则T原地不动
- **停机情况**：1）若某时刻T @ q_{accept}或q_{reject}，T停机，并接受或拒接ω；
  2）$\delta(q,s)$对某些q和s可能无定义，T停机

由图灵计算理论衍生出的冯诺依曼体系结构

- 图灵机计算范式中的元素可在冯诺依曼架构中找到对应



图灵机



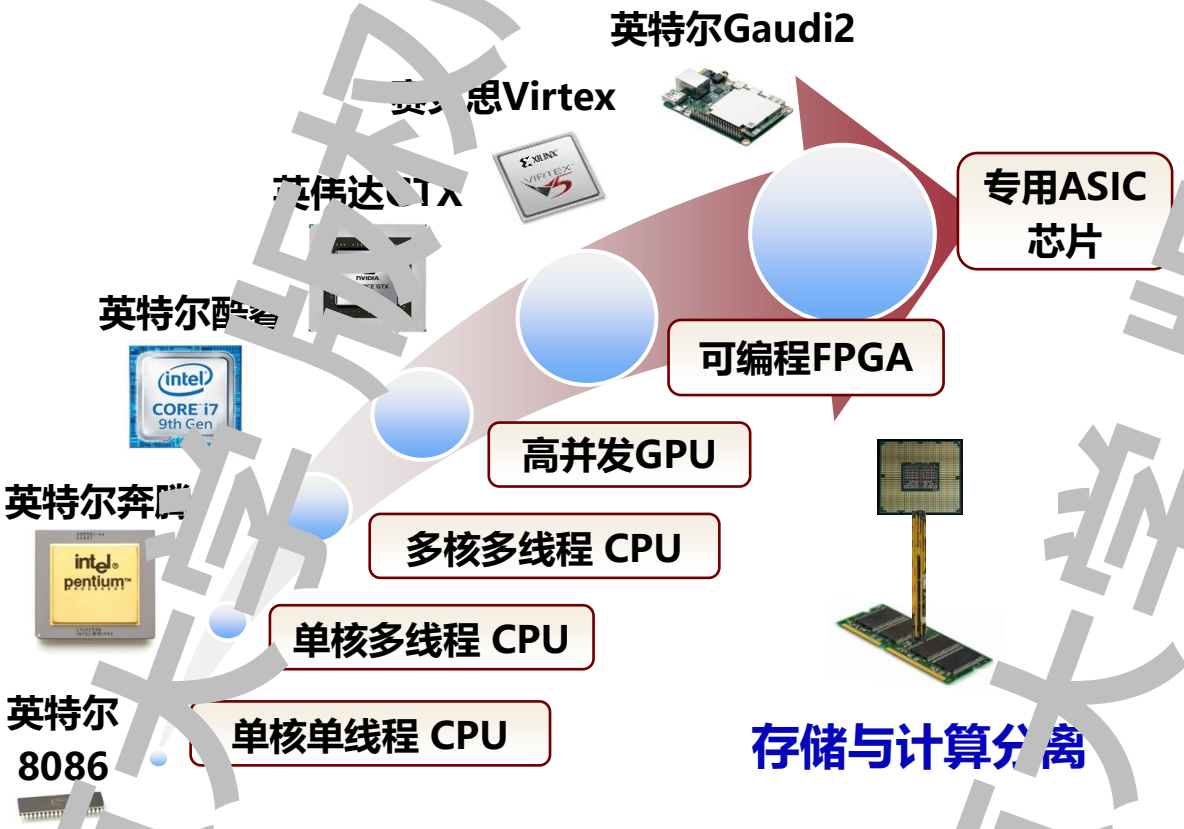
笔 & 状态 → 数据存储器

[illegible]带 → 指令存储器

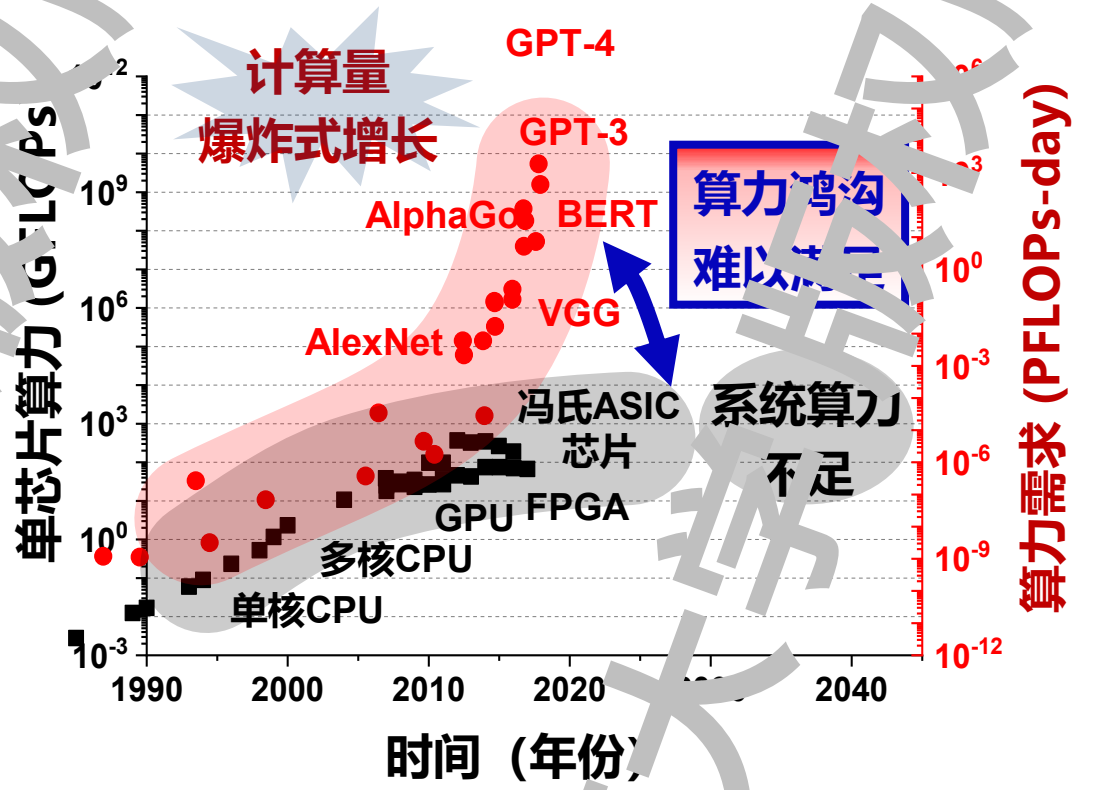
运算法则 → 控制逻辑&计算单元

冯诺依曼体系结构

- 从偏向通用计算任务的CPU、GPU到偏向定制化设计的FPGA、ASIC演进



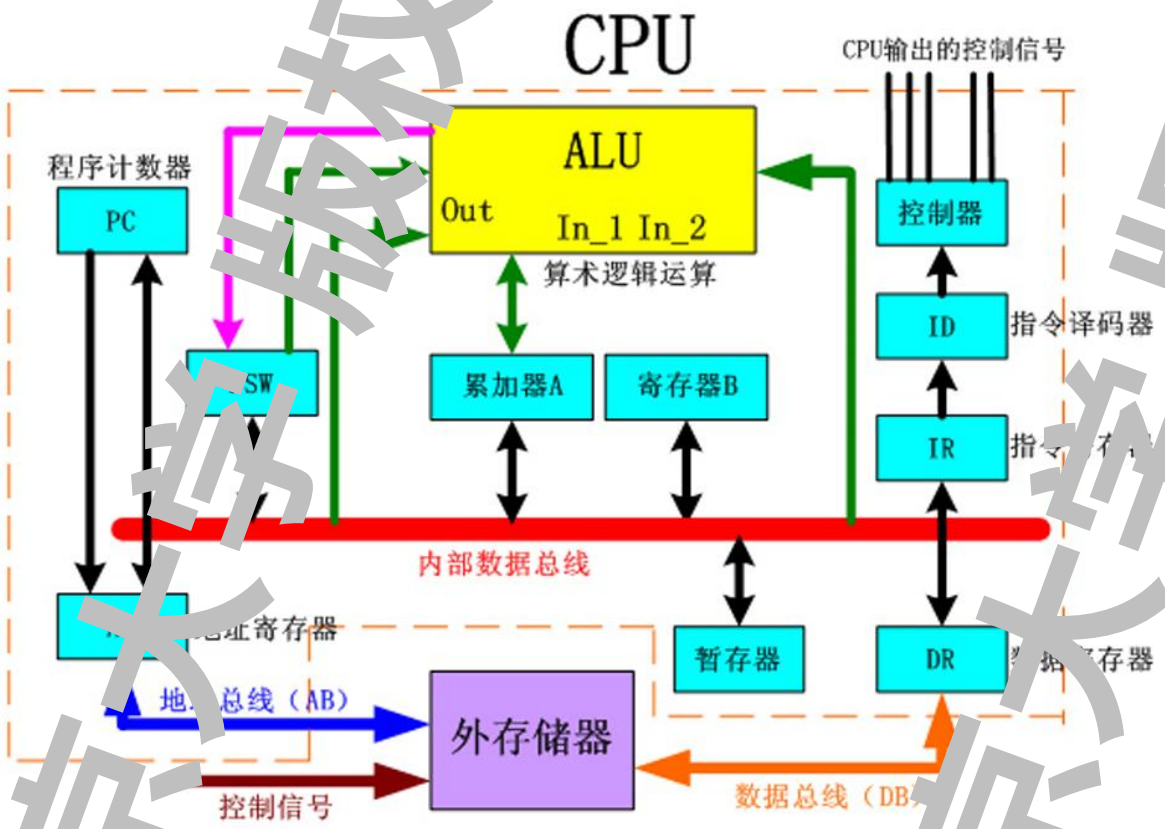
智能芯片体系结构演进图

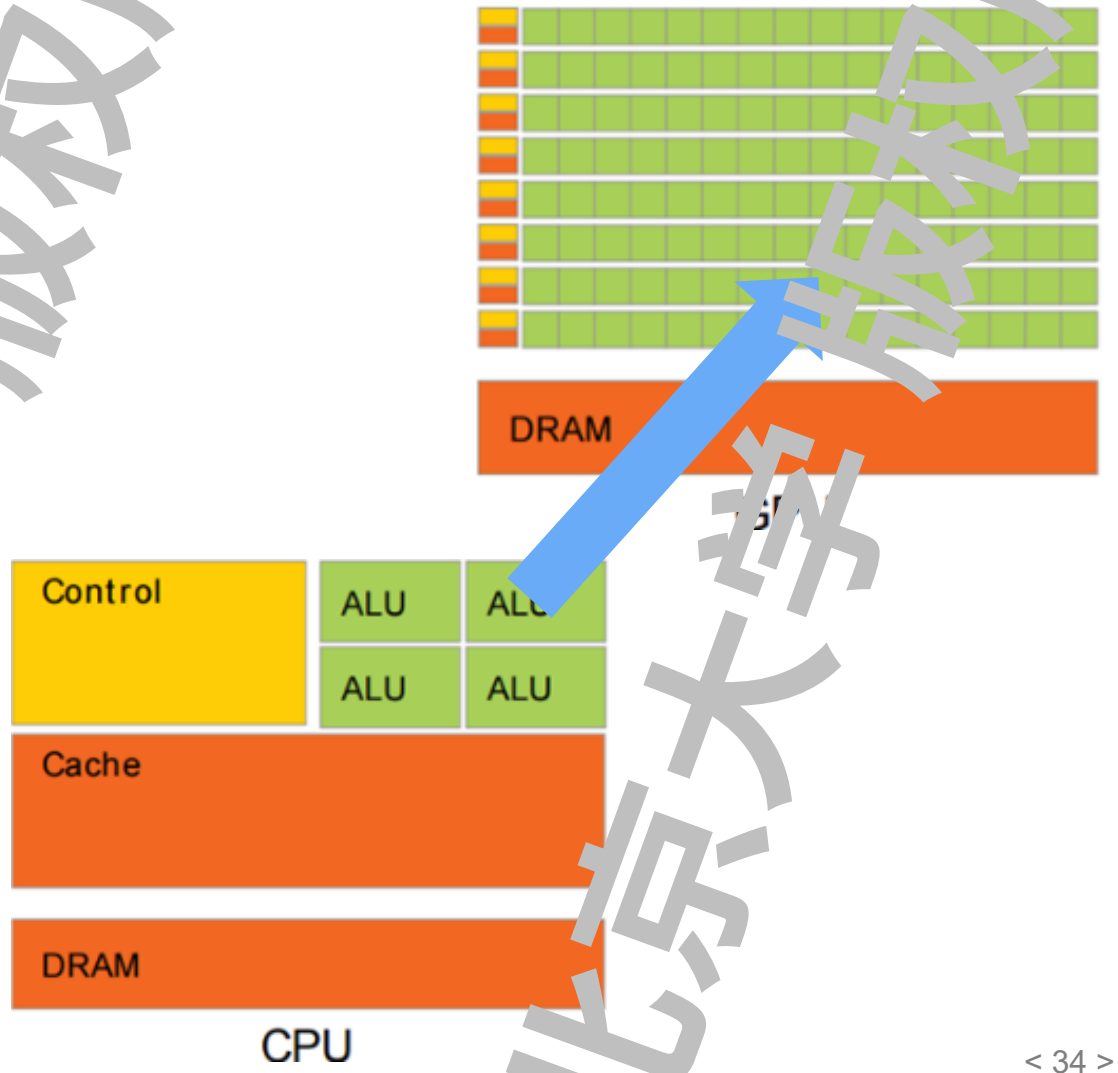


新兴计算任务所需的计算量呈爆炸式增长

典型智能芯片体系结构 – CPU/GPU

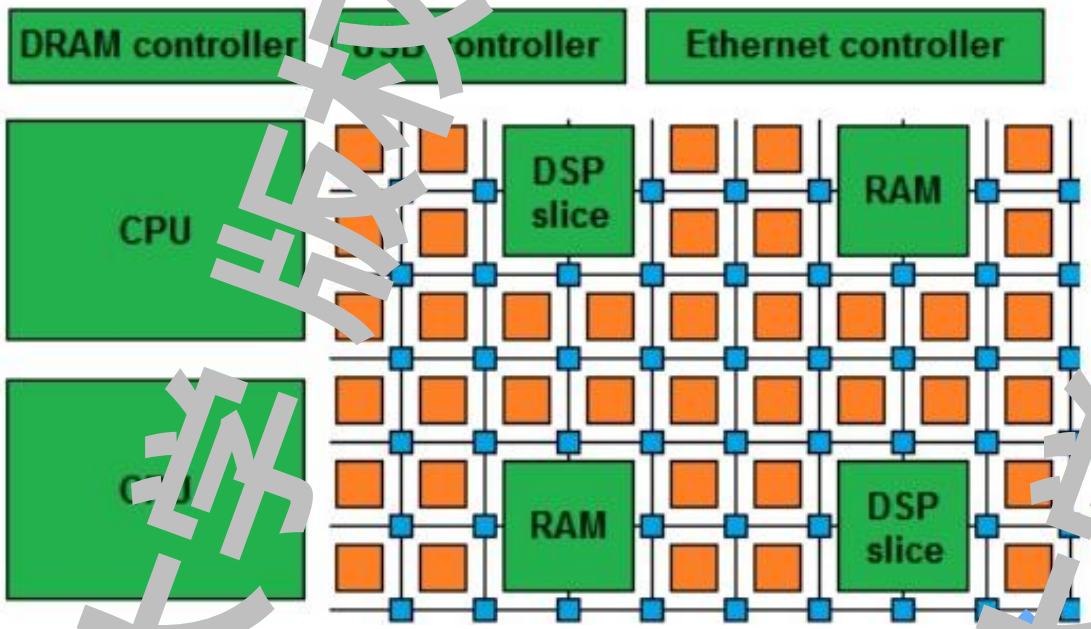
- 从偏向通用计算任务的CPU、GPU到偏向定制化设计的FPGA、ASIC





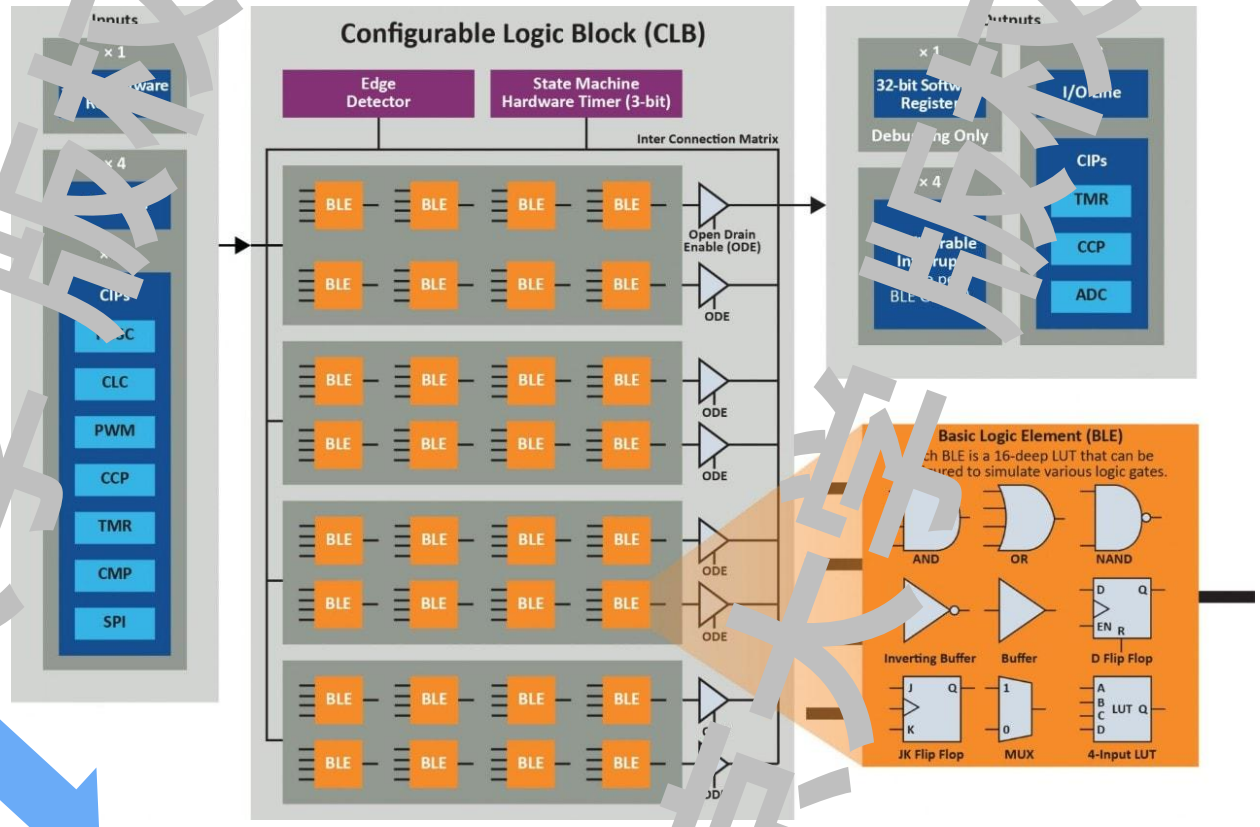
典型智能芯片体系结构 – FPGA

- 从偏向通用计算任务的CPU、GPU到偏向定制化设计的FPGA、ASIC



可编程FPGA

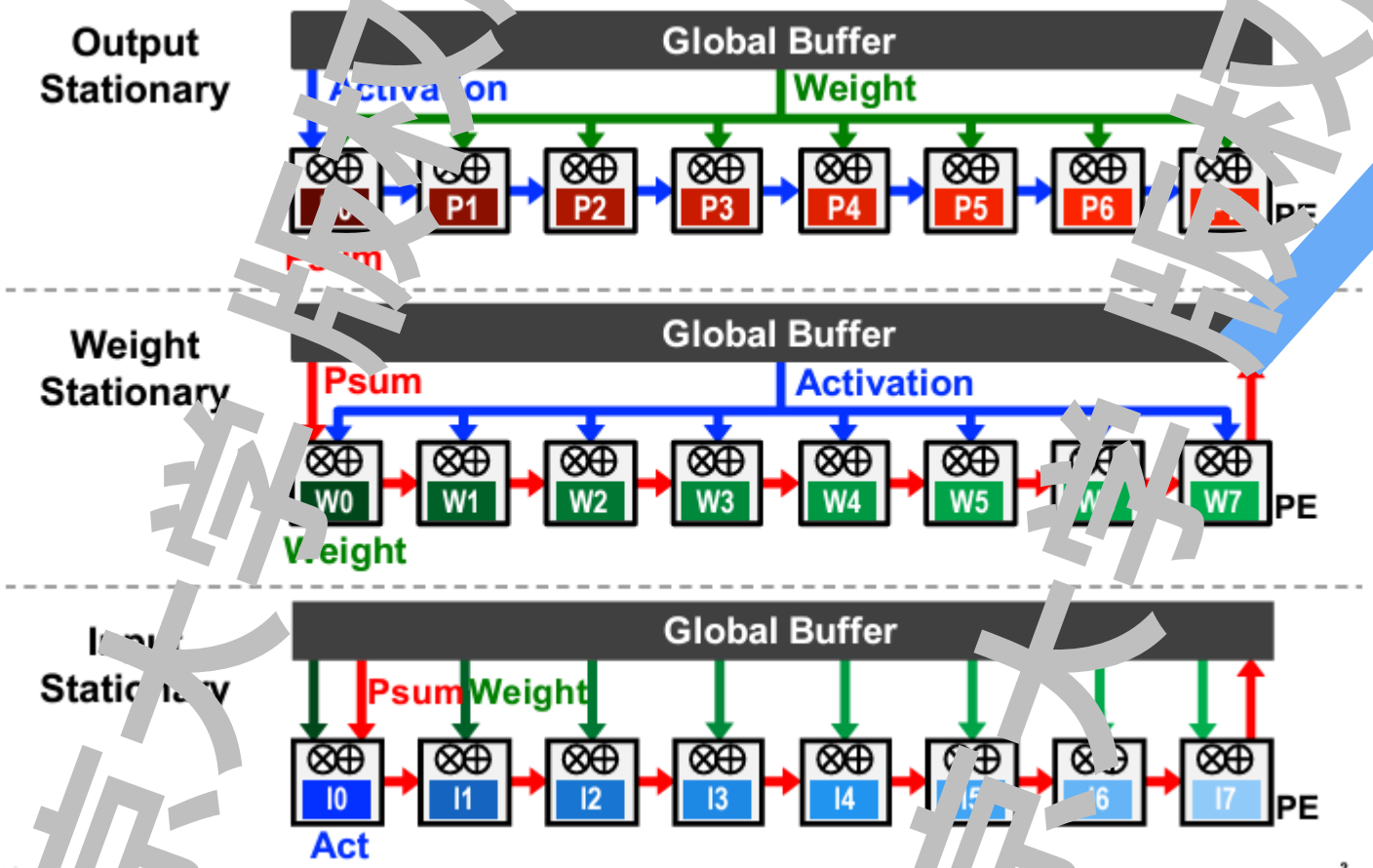
- 比CPU快
- 比GPU省功耗
- 比ASIC便宜流片周期短



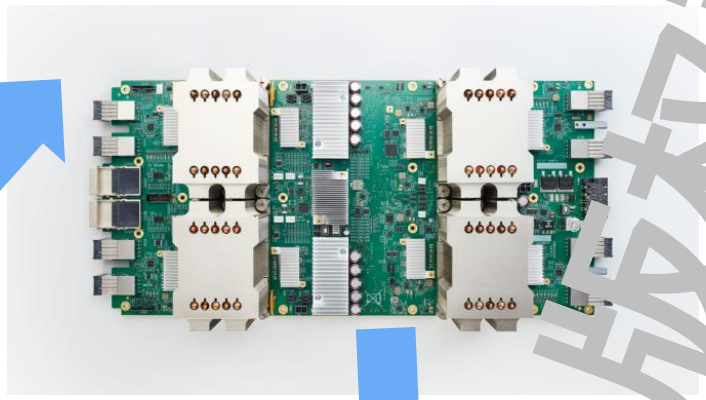
可编程逻辑模块CLB

典型智能芯片体系结构 – ASIC

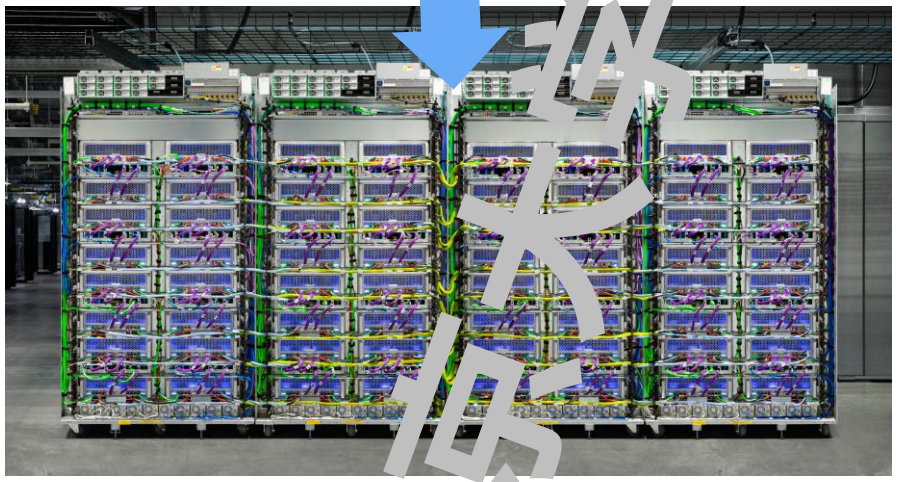
- 从偏向通用计算任务的CPU、GPU到偏向定制化设计的FPGA、ASIC



经典的神经网络加速器ASIC体系结构



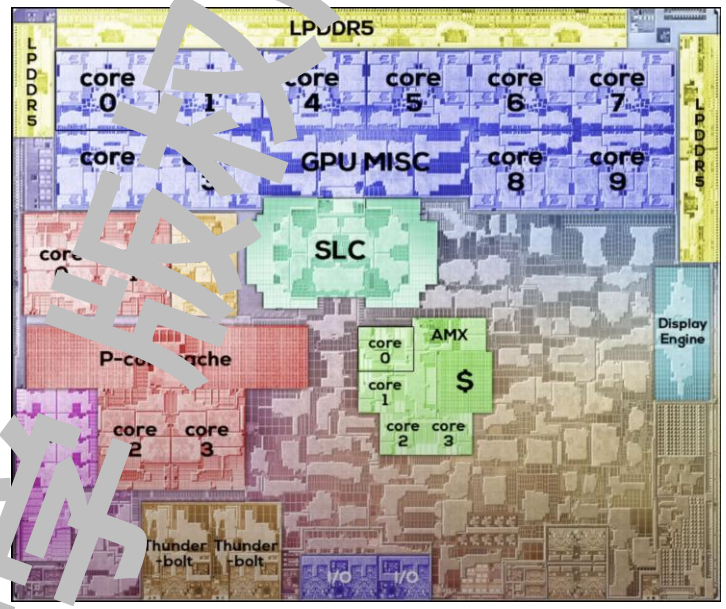
TPU
ASIC



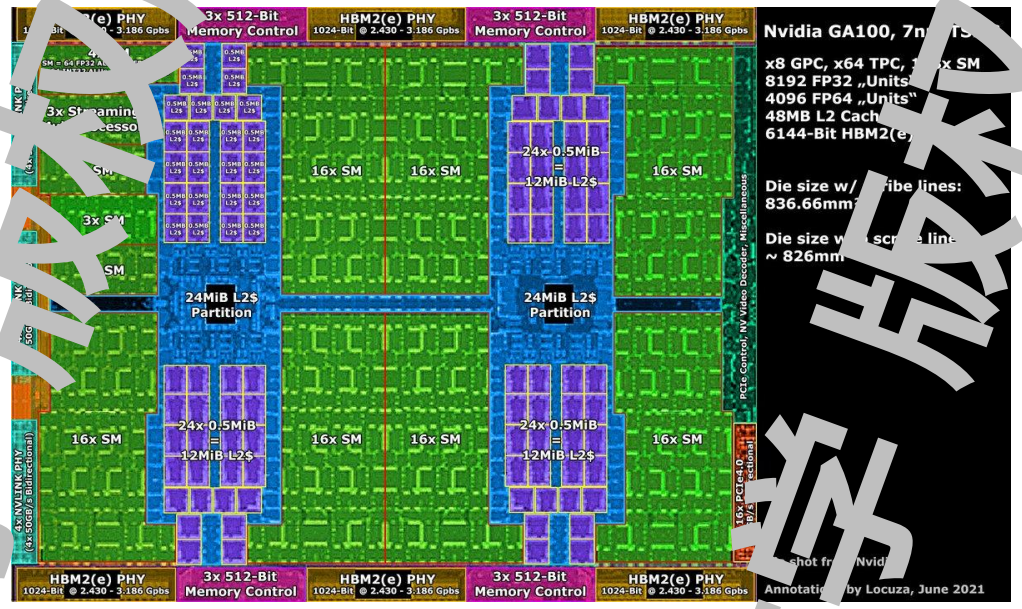
TPU组成的大型AI服务器

冯诺依曼体系结构

- 目前的成熟商用芯片基本均采用冯诺依曼体系结构



Apple M3



NVIDIA H10

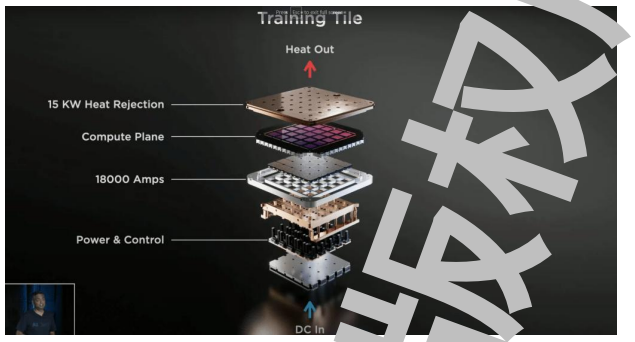
特点：存储与计算单元分离，依靠总线进行连接

执行程序时需要来回搬运数据（读出→计算→写入）

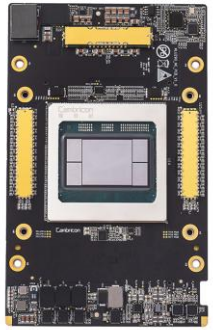
推动智能时代飞速发展：AI芯片 – 2014/2015年至今

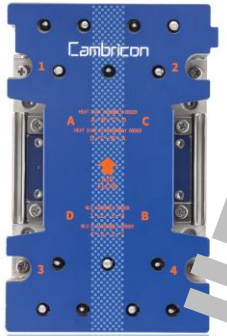
- 高性能AI芯片成为推动智能时代发展的算力基石，将定义未来十几年的技术革命

Google TPU

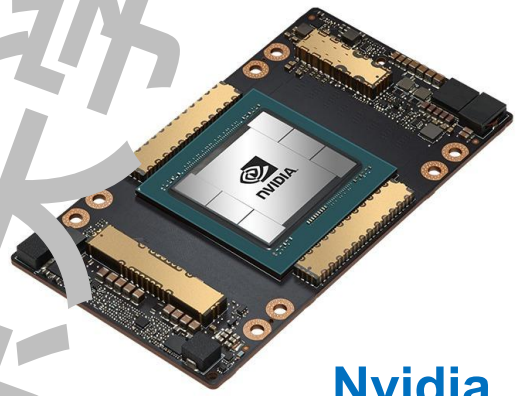


Tesla Dojo

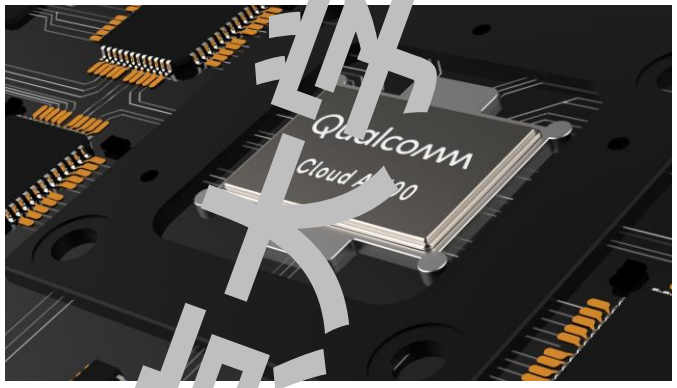




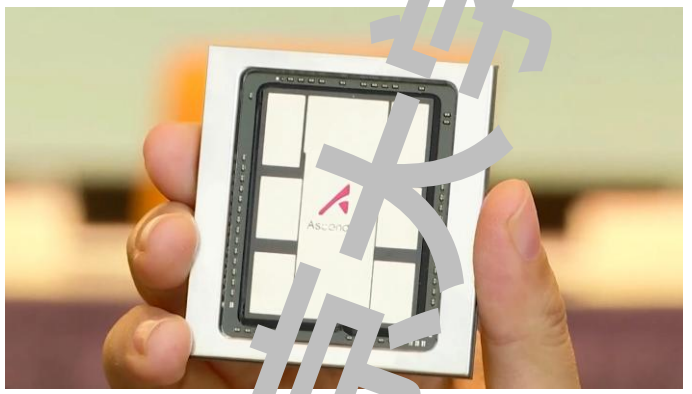
寒武纪



Nvidia GPU



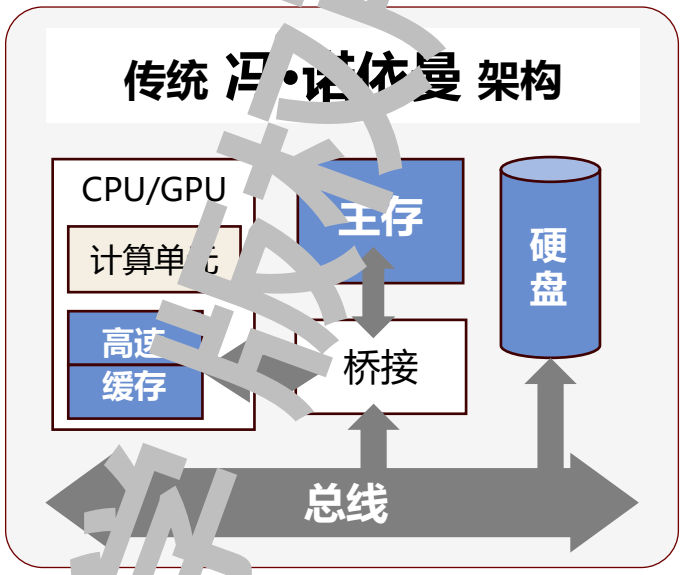
Qualcomm Cloud AI

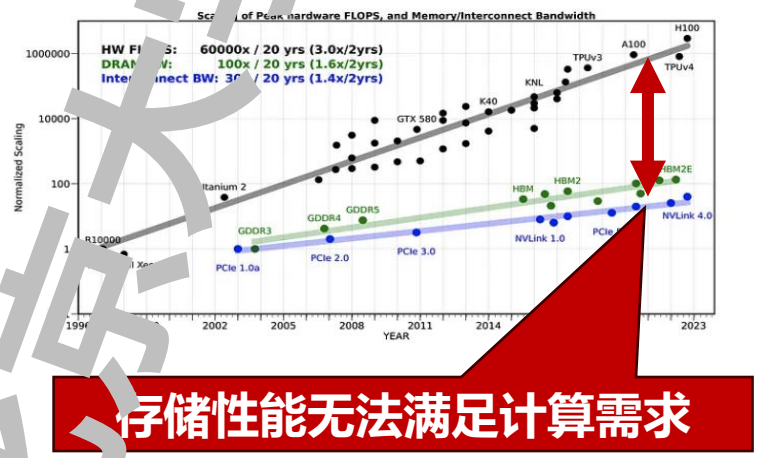


华为昇腾

冯诺依曼体系结构

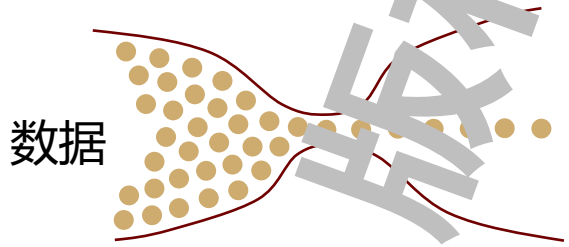
- 当前智能芯片体系结构的瓶颈



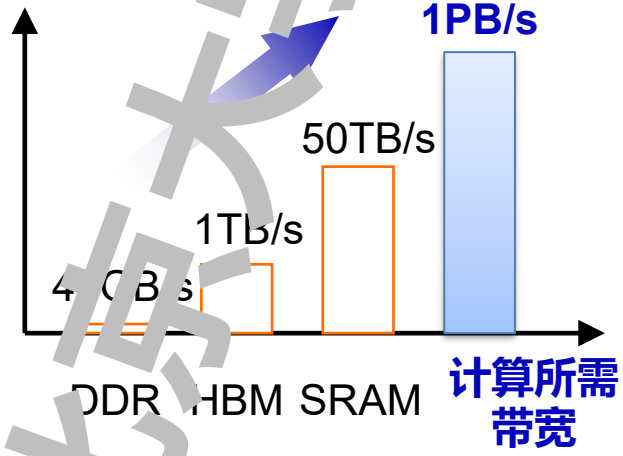


存储性能无法满足计算需求

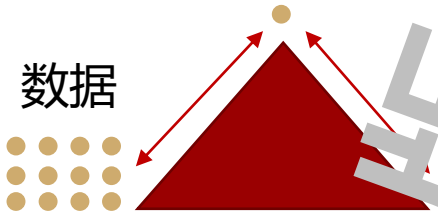
存储带宽限制算力



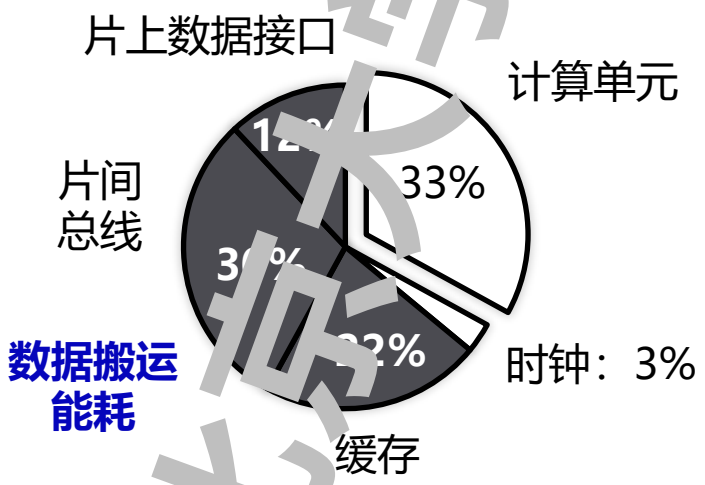
存储器 处理器



数据搬运限制能效



存储器 处理器



融合新器件、新架构、新计算是后摩尔时代体系结构的发展趋势

- 融合新器件、新架构、新计算是突破后摩尔时代大算力、高能效瓶颈的重大关键技术领域

数字设计是什么：一条抽象层次的链条

每一层的输入输出都有明确格式 —— 这正是 AI 能够逐层切入的前提

1	规格与架构	性能 / 功耗 / 面积目标，微架构划分	C、SystemC、性能模型
2	RTL 设计	用寄存器传输级描述电路行为	Verilog、VHDL、Chisel
3	逻辑综合	RTL + 约束 → 门级网表	综合工具、标准单元库
4	物理设计	布局规划、布线、时钟树综合	P&R 工具、宏单元布局
5	验证与签核	功能、时序、功耗、DRC / LVS	仿真、形式验证、STA
6	流片交付	输出 GDSII，交由代工厂制造	掩模、晶圆厂

为什么必须分层？

- 分层是复杂度的分而治之，每层只回答一类问题
- 层间接口可被机器读取，可自动化、可评测
- AI 因此能逐层切入，而不必一次吃掉全流程

贯穿全课的一句话

设计范式的每一次进步，都是把人从一个抽象层次，解放到更高的一层。

五十年，四次范式跃迁

抽象层次每抬升一级，人交给机器的东西就更多一层

1970s　1980s　2000s　2010s　2020s

手工时代

- 手绘版图、胶带贴图
- 设计单位 = 晶体管
- Intel 4004：2300 管

EDA 自动化

- 逻辑综合诞生
- Verilog / VHDL 普及
- 设计单位 = RTL 语句

IP 复用与 SoC

- 平台化、IP 集成
- 设计单位 = IP 核
- 验证成为最大工作量

云与大规模并行

- 分布式回归与形式验证
- 自动化设计空间探索
- 用算力换设计时间

AI 协同

- RL 布局、LLM 生成 RTL
- 智能体驱动流程
- 设计单位 = 设计意图

规律：每一次跃迁的本质，都是把某一层的“怎么做”交给机器，让人只需要回答“做什么”。

阶段一（1960s–[illegible]70s） 手工时代：人即是工具

极致定制，但不可扩展

- **版图靠手绘。**工程师在坐标纸上画出每个晶体管，再用红/蓝胶带逐层拼贴，规模[illegible]数千管量级。
- **正确性靠[illegible]。**没有可依赖的自动检查，设计规则与[illegible]靠心算、走查和经验判断。
- **试错几乎不被允许。**一次流片失败意味着数月周期与全部预算的损失。
- **优点是极致定制。**面积与性能可以压到接近器件的理论极限。
- **局限：不可扩展。**复杂度指数增长，而人力只能线性增长。

五十[illegible]年，规模跨越约七个数量级 —— **靠手是接不住的。**

2,300

Intel 4004（1971）全芯片晶体管[illegible]量

1000 亿+

当代先进节点单颗大芯片的晶体管量级

阶段二（1980s–1990s） EDA 自动化：把门级交给机器

关键突破是逻辑综合：从 RTL，机器出门级网表

RTL 描述 → **逻辑综合** → 布局布线 → 时序签核 → 版图输出

- **抽象层次抬升一级。**设计生产率随之获得约一个数量级的提升，百万门芯片第一次成为可能。
- **引入优化算法。**人用 SDC 给出时序与面积目标，工具在解空间中搜索一个可行实现。
- **“设计”的含义改变了。**从“画出电路”变成“描述行为并给定目标”。

但它留下了两个问题

- 工具是确定性算法：同样的输入永远给同样的输出，不会从历史项目中积累任何经验。
- “怎么写约束、怎么调参数”这类知识，始终只存在于资深工程师的脑子里，无法复制、无法规模扩张。

→ 这恰恰是 AI 要接手的部分

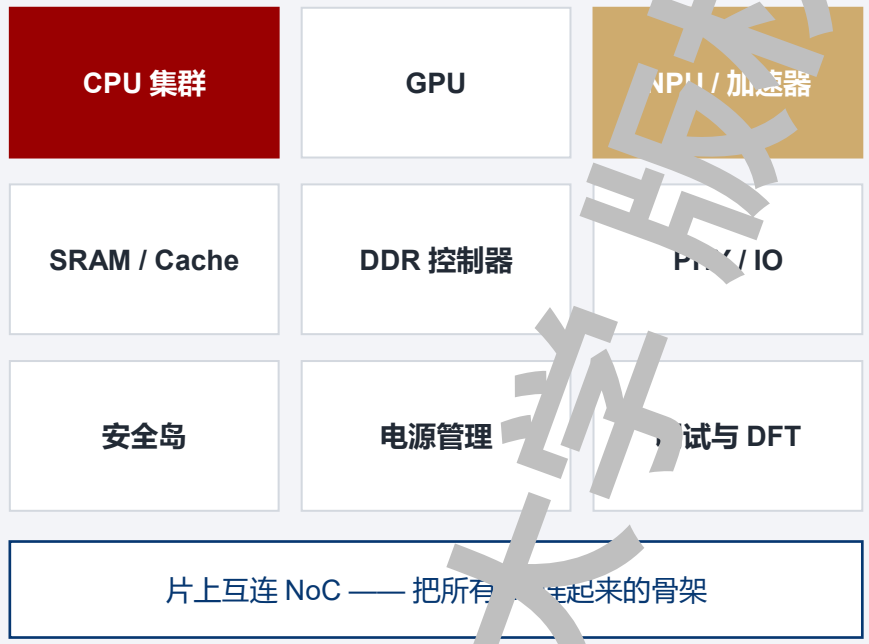
阶段三（2000s–2010s） IP 复用与 SoC：设计单位变成模块

瓶颈从“怎么设计出来”变成到“怎么证明它是对的”

- **基本单位从语句变成 IP 核。**CPU、GPU、NPU、DDR 控制器、PHY、SRAM 编译器……
- **造芯片更像做系统集成。**选 IP、连互连、跑验证、做软硬件协同设计。
- **复用率决定成本与周期。**自研与外购的边界成为一个战略问题，而不只是技术问题。
- **新瓶颈：验证复杂度爆炸。**验证与集成通常占整体工作量的六成以上，逻辑设计本身反而不再是难点。

状态空间是组合爆炸的：集成度每提高一倍，需要覆盖的交互场景远不止翻倍。

典型 SoC 的构成



集成度越高，验证状态空间越大

阶段四（2020s–） AI 协同的三个层次

L1 已普及，L2 已进入…，L3 仍在早期 —— 判断一项工作在哪一层，是本课的基本功

L1 AI 辅助 · Copilot

人主导，AI 补全

- RTL 片段生成、testbench 生成
- EDA 脚本编写、文档问答
- 代码审查、注释与重构

今天已大规模可用

L2 AI 增强 · Optimizer

人设目标，AI 搜索解空间

- PPA 多目标寻优
- 布局规划与工具参数调优
- 回归用例智能筛选

已进入商用 EDA 产品

L3 AI 自主 · Agent

AI 闭环执行，人做审核

- 意图 → 实现 → 验证 → 迭代优化
- 自动定位并修复缺陷
- 跨工具的流程编排

研究与早期试点阶段

课程立场：目标不是取代工程师，而是把工程师从“实现”推到“意图与判断”这一层

三种范式的横向对比

注意最后一行 —— 知识沉淀方式的改变，才是最深层的影响

对比维度	手工时代	EDA 自动化	AI 协同
设计输入单位	晶体管与版图图形	RTL 代码 + 约束	设计意图 + 优化目标
人的角色	执行者	约束设定者	目标设定者与审核者
单次迭代周期	数周 ~ 数月	数天 ~ 数周	数小时 ~ 数天
主要瓶颈	人力规模	工具运行时间与专家经验	验证可信度与数据质量
结果可复现性	低，依赖个人	高，算法确定性	中，需固定种子与评测基线
知识如何沉淀	师徒口传	脚本与方法学文档	数据集、奖励权重与评测集

为什么拐点出现在现在？四个驱动力

摩尔定律放缓之后，增长必须来自设计效率，而不再只来自制造工艺

1 复杂度失控

先进节点单芯片晶体管数已达千亿量级，设计规则与物理效应深度耦合，人工穷举不可行。

2 成本曲线陡峭

先进节点一次流片加设计投入可达数千万至数亿美元，试错空间被极度压缩。

3 经验出现断层

资深物理设计与验证工程师供给远低于需求，而这些经验高度隐性，难以通过文档传递。

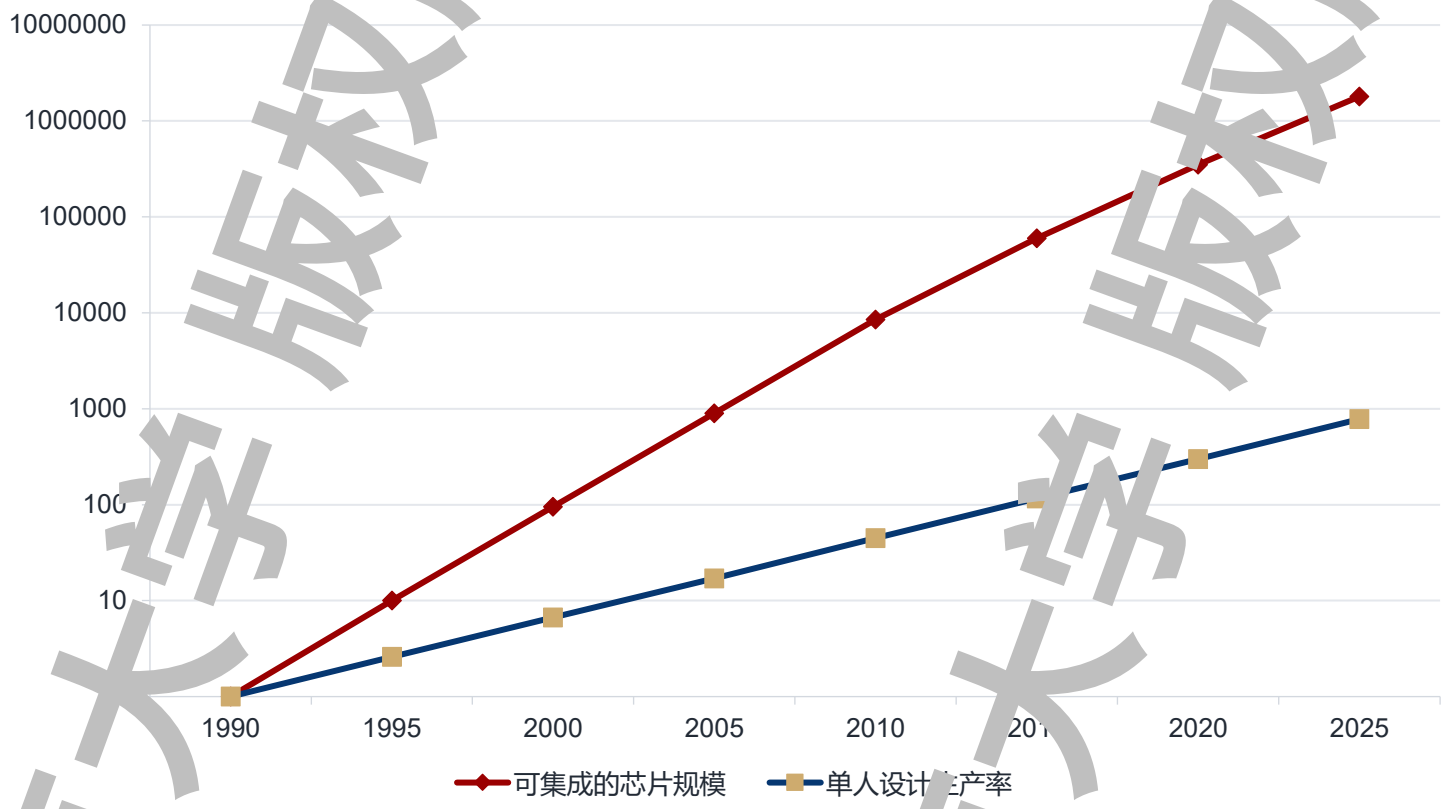
4 模型能力跃迁

大模型在代码生成与结构化推理上跨过可用门槛；EDA 积累的日志、网表、报告本身就是可学习的结构化数据。

四条曲线在 2020 年后同时到达临界点 —— 这就是“为什么是现在”。

设计生产率鸿沟：这一次靠什么填？

芯片可容纳规模年增约 58%，而单人设计生产率年增约 21%



示意图，以 1990 年为 1 的相对增长指数，纵轴对数刻度。增速取自业界对设计生产率[illegible]的长期观察（ITRS），非精确测量值。

58% vs 21%

年均复合增速之差，逐年以复利累积

- 两条曲线的斜率差每年复利累积，缺口不断放大。
- 历史上每一次缺口拉大，都逼出了一次抽象层次跃迁：门级→RTL，RTL→IP。
- 这一次能填补缺口的候选，就是 AI 协同设计 —— 让人只描述意图。

自我加速的飞轮：AI 既是设计对象，也是设计工具

飞轮转起来之后，速度取决于最慢的那一环 —— 今天通常是验证

接下来的五个案例，正是这个飞轮上不同位置的真实证据。

案例一 · AlphaChip：宏单元布局为什么难？

Google DeepMind | Nature 2021（2020 预印本，2024 发布附录并命名）

- **任务。**在给定的芯片画布上，为数百至上千个宏单元（SRAM、寄存器堆等）选定位置与朝向，再把确定性算法摆放标准单元。
- **目标。**同时压低线长、布线拥塞与单元密度——三者互相冲突。
- **代价。**一个大模块的平面规划，资深工程师通常要迭代数周到数月。

组合爆炸有多大？

一块的状态空间约 10^{360}；而一块含 1000 个宏单元的芯片，其布局解空间常被估计在 10^{2500} 量级以上。穷举与人工试错都不成立。

真正的困难：评估一次太贵

给出一个布局方案

↓

跑布线与时序分析

↓

得到真实 PPA 与标

单次评估：数小时

- 于是必须学一个便宜的代理奖励，去近似昂贵的真实指标 —— 这是全文方法的出发点。

图示为依据论文思路重绘的示意图

AlphaChip 方法（一）：把布局建模为序贯决策问题

像下棋一样，一次放一个宏单元；棋盘是离散化的芯片画布

依据 Mirhoseini et al., Nature 2021 的方法描述重绘

- **为什么是序贯的。**先放的宏单元会改变后面所有位置的可行性，天然是决策序列而非一次性优化。
- **为什么是稀疏奖励。**只有全部放完才能算出线长与拥塞，中间步骤奖励为零。
- **动作掩码是关键工程。**把重叠、越界、违反约束的网格点直接屏蔽，极大缩小有效动作空间。

与传统方法的差别

- 模拟退火 / 力导向：每块新芯片都从零开始搜索，历史经验不留存。
- 强化学习：策略参数就是经验的载体，见过的芯片越多，起点越好。

AlphaChip 方法（二）：边特征图神经网络与代理奖励

让模型学到“电路连接关系”的通用表示，从而在新芯片上也能泛化

策略 / 价值网络结构（依据论文重绘示意）

- **关键创新在“边”。**把连线本身也作为可学习的对象，模型因此学到的是电路的连接模式，而不是某一块芯片的坐标。
- **于是可以泛化。**在 A 芯片上学到的经验，能迁移到从未见过的 B 芯片 —— 这是预训练能起作用的根本原因。
- **价值网络的作用。**在稀疏奖励下预测“这一步之后大概能得多少分”，为策略提供稳定反馈，否则训练几乎无法收敛。

代理奖励函数

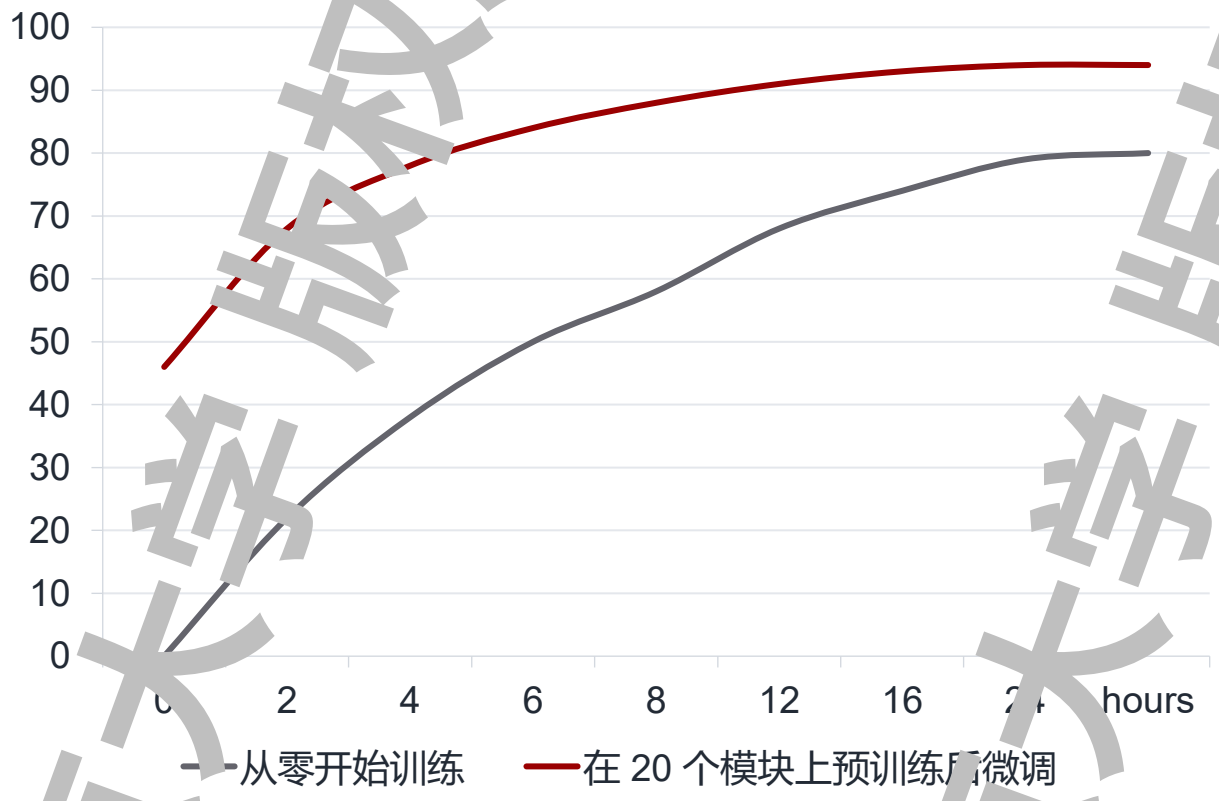
$$R = -(\text{线长} + \lambda \cdot \text{拥塞} + \gamma \cdot \text{密度})$$

- **线长。**用半周长线长（HPWL）近似，计算便宜。
- **拥塞。**用布线资源使用量的估计值代替真实布线。
- **密度。**约束单元过度堆叠，保证后续可布通。

代理奖励是整套方法的软肋，也是后来争议的焦点：它必须便宜到可以在训练中调用上百万次，又要与真实签核指标足够相关。一旦代理与真实指标脱钩，模型优化的就不再是我们真正想要的东西。

AlphaChip 方法（三）：预训练让它“越用越强

在 20 个历史 TPU 模块上预训练，再在新模块上微调 —— 这是与传统优化最根本的差别



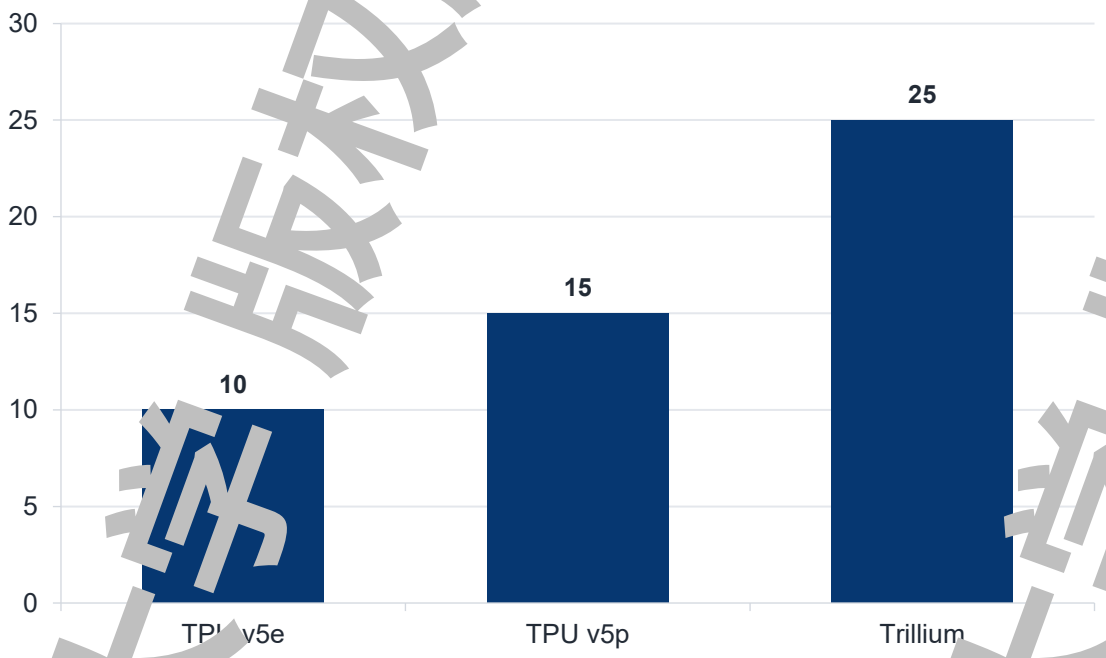
示意曲线：纵轴为布局质量（越高越好），横轴为微调时长；基于论文对预训练收益的描述重绘，非原始数据。

- **样本起点更高。**预训练模型在没见过的新模块上，第一次输出就已接近可用。
- **收敛更快。**达到同等质量所需的微调时间显著缩短。
- **经验会累积。**见过的芯片模块越多，模型越强 —— 这与人类专家的成长方式一致，而确定性算法做不到。
- **已公开预训练权重。**2022 年开源 Circuit Training 代码，2024 年随 Nature 附录发布在 20 个 TPU 模块上预训练的检查点。

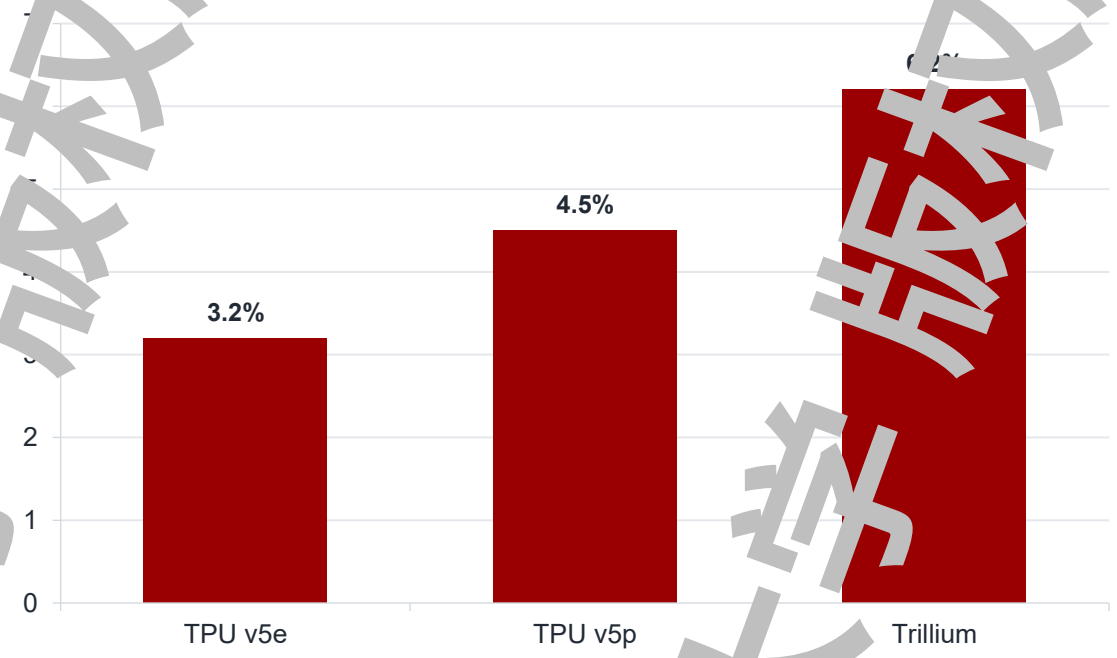
教学要点：“会积累经验”是 AI 方法相对传统 EDA 算法唯一的结构性优势，其他都是量的差别。

AlphaChip 结果：三代 TPU 上的量化收益

官方数据：AlphaChip 布局的模块数与相对人类专家的线长缩短，逐代提升



AlphaChip 布局的模块数量（个）



相对人类专家的平均线长缩短（%）

- **时间：**生成可用平面规划从数周至数月缩短到数小时。
- **扩展：**同一方法已用于 Google Axion（Arm 架构数据中心 CPU）等芯片；联发科亦在其先进制程芯片上做了扩展应用。

数据来源：Google DeepMind 2024 年 Nature 附录及官方博客公布的数值。

AlphaChip 的争议：一条必须完整讲的时间线

这不是八卦 —— 它定义了“AI 设计结果如何才算可信”的行业标准

2020.04

arXiv 预印本发布

《Chip Placement with Deep RL》

2021.06

Nature 正式发表

同期配发 News & Views 评论

2022

开源 Circuit Training

代码公开，但预训练数据未公开

2023.02–

两项质疑先后发布

Cheng 等（UCSD，Kahng 组）复现失败；

Mirhoseini 发表《The False Data》

2023.09

Nature 加注编辑说明

称性能主张受到质疑，正在调查；

次日撤下配套评论文章

2024.09

Nature 附录 + 命名 AlphaChip

公布三代 TPU 数据，公开预训练权重

2024.11

《That Chip Has Sailed》

Google 逐条回应质疑，

指出复现方未做预训练、算力不足

争议的三个技术焦点

- 对照基准是谁：与“某位未具名的人类专家”比，还是与模拟退火、商用工具在公开基准上比？
- 预处理是否被计入：论文未明确说明布局前的处理用商用 EDA 工具做过准备；复现方未做预训练、算力也远低于原作
- 双方对“是否复现了原方法”各执一词。

从这场争论中，我们该学到什么？

把它当成一份 AI 设计成果的验收清单

1 对照基准必须公开可比

不能只与一个“专家”比。要与模拟退火、商用工具、公开基准（如 ISPD、MacroPlacement）同台。

2 完整流程必须披露

预处理、后处理、使用了哪些商用工具，都会影响结果归属，必须写清楚。

3 算力与超参必须报告

同一算法在不同算力预算下差异巨大；不报告算力的对比没有意义。

4 随机性必须被固定

报告多次运行的均值与方差，公开随机种子，而不是只报最好的一次。

5 数据与权重尽量开放

能否复现，最终取决于别人能否拿到同样的起点。

6 结论要与代理指标脱钩

代理奖励好看不等于签核指标好看；最终结论必须落到真实 PPA 上。

本周作业的评分标准，就参照这六条。

案例二 · PRIME：用离线数据设计 AI 加速器架构

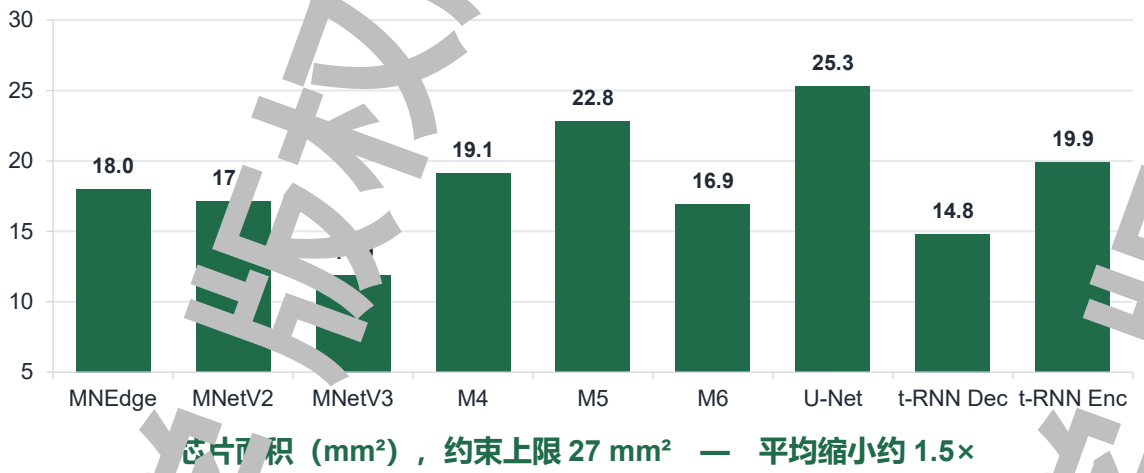
Google | ICLR 2022 | 从“用 AI 优化布局”上升到“用 AI 决定架构参数”

- **问题。**加速器架构有大量离散参数：PE 阵列规模、片上存储容量、带宽分配、数据流方式……组合空间巨大。
- **传统做法。**仿真驱动搜索：每评估一个配置就跑一次周期精确仿真，慢且每换一个应用就要重来。
- **PRIME 的做法。**只用历史积累的日志数据（含大量不可行配置），离线学一个保守的代理模型，再对代理模型做优化，全程不再调用仿真器。
- **“用不可行数据”是关键。**失败的设计同样携带信息，让代理模型知道哪里是悬崖。
- **代价大幅下降。**论文报告：相对最优仿真驱动方法，性能提升约 1.54×，同时总仿真时间减少 93%–99%。

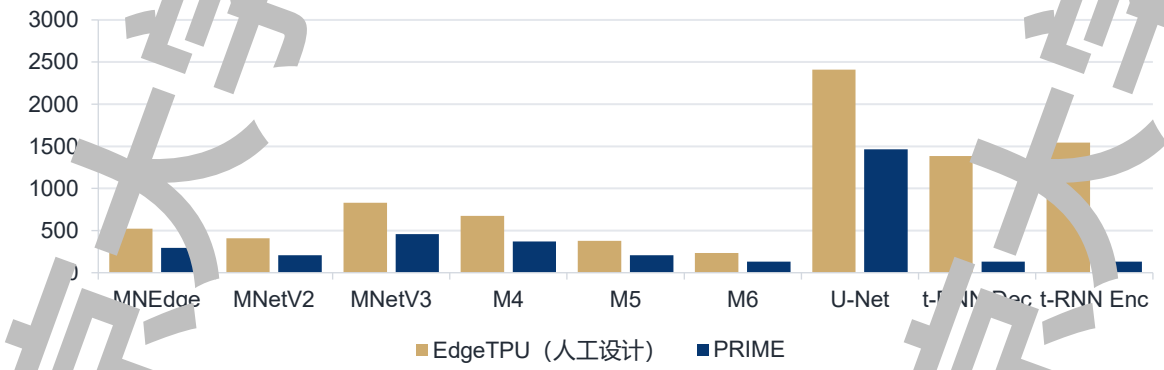
依据 Kumar et al., ICLR 2022 论文流程重绘

PRIME 结果：比人工定制的 EdgeTPU 更小、更快

在 27 mm² 面积约束下，针对九个模型分别做专用化设计



芯片面积（mm²），约束上限 27 mm² — 平均缩小约 1.5×



延迟（毫秒） — 平均加速约 2.69×，最高 11.84×（t-RNN Enc）

两个值得注意的细节

- 面积不是优化目标，却顺带下降了 —— 说明代理模型学到了参数之间的真实权衡。
- t-RNN 上的巨大加速来自结构专用化，而不是简单地堆资源。

- **对照要看清楚。** PRIME 是为每个应用单独定制，EdgeTPU 是一颗通用芯片，这个对比展示的是“专用化的潜力”，不能读成“AI 全面超越人类设计”。
- **方法论价值。** 它证明了历史日志数据可以直接变成设计能力 —— 数据本身就是资产。

数据来源：PRIME 论文附录 Table 14（27 mm² 面积约束，25 GBps DRAM 带宽）

自我加速的飞轮：AI既是设计对象，也是设计工具

- 设计AI芯片架构 -> 利用AI设计AI芯片架构

参数化硬件单元库　　针对某类任务的最优芯片设计

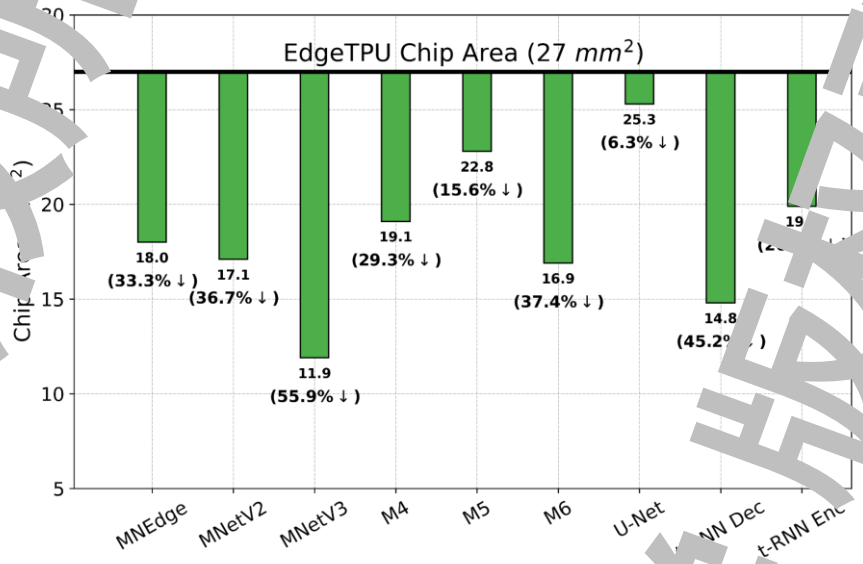
PRIME

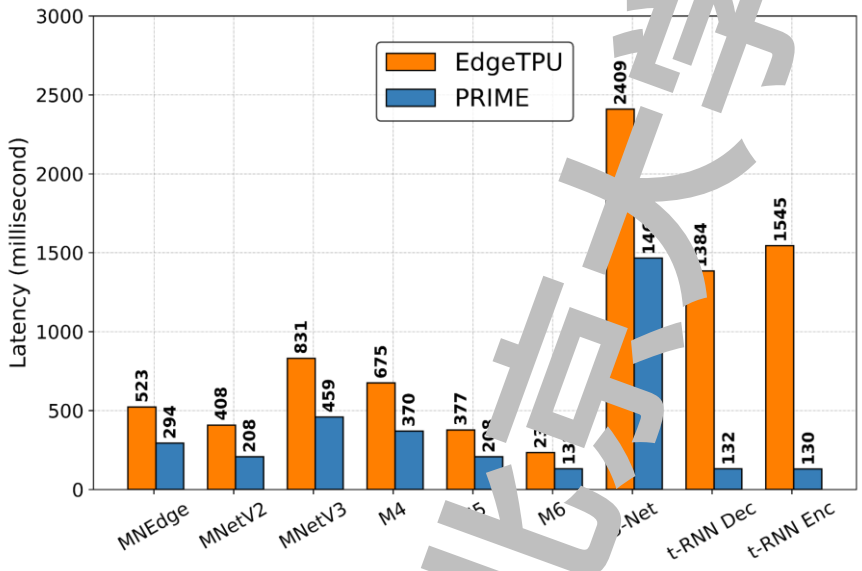
Optimizer

RL、大模型等方式

设计AI芯片的AI模型

AI芯片





思想自由 兼容并包

北京大学
PEKING UNIVERSITY

PART 03 | Case Studies in Depth

案例深度剖析

案例三 · ChipNeMo：芯片设计的领域大模型

NVIDIA | 2023 | 问题不是“大模型行不行”，而是“通用大模型为什么不够”

通用大模型在芯片设计场景的三个短板

- **专有语言与内部工具。**公司内部的脚本语言、工具命令、库函数在公开语料中几乎不存在，模型没见过。
- **术语被错误分词。**通用分词器会把电路术语、信号名切得支离破碎，浪费上下文且损失语义。
- **知识不在参数里。**设计规范、方法学文档是内部资产，模型无从知晓，只能靠检索补进来。
- **成本敏感。**内部部署要考虑推理成本，不可能对每个工程师都挂一个 70B 模型。

训练语料：全部来自内部资产

- 内部设计源码（RTL、C/C++、脚本）
- 验证代码与测试平台
- 架构文档与设计规范
- Bug 记录与工程师讨论
- 内部 Wiki、工具手册与 FAQ

按比例混入公开数据，抑制通用能力遗忘

语料类型依据 ChipNeMo 论文描述整理；具体规模以论文为准

ChipNeMo 方法：四步领域适配流水线

不是简单微调 —— 四项技术分别解决分词、知识、指令与检索四个问题

① 定制分词器

在通用分词器上扩充
芯片领域词表
复用原有嵌入初始化

② 领域自适应
继续预训练 DAPT

在 LLaMA2 7B/13B 上
用内部语料继续预训练

③ 领域指令微调
SFT

用领域指令数据 +
通用指令数据混合微调

④ 领域适配
检索模型 RAG

微调检索器
显著提升命中率

依据 ChipNeMo 论文（arXiv:2311.00176）方法章节重绘

	① 分词器	② DAPT	③ SFT	④ RAG
解决什么	术语分词效率	领域知识注入	任务对齐	实时知识接入
代价	需重训嵌入层	算力开销最大	需人工构造指令	需构建知识库
可迁移性	高，任何团队可做	需十亿级 token 语料	中	高，最容易先落地

务实顺序建议：先做 ④ RAG（成本最低、见效最快），再考虑 ②DAPT（成本最高、收益也最大）。

ChipNeMo 发现（一）：小而专，胜过大而全

领域适配后的 13B 模型，在多数设计任务上追平甚至超过 70B 通用模型

最多 5×

在效果相当或更优的前提下，模型规模可以缩小的倍数

- **部署门槛。**13B 模型无需量化即可装入单张 A100；70B 模型做不到，这直接决定了能不能给全公司工程师用。
- **成本结构。**论文引用的估计是：同等延迟目标下，8B 模型的推理成本比 62B 低 8–12 倍。
- **通用边界。**领域适配的收益，集中在公开语料稀缺的任务上，通用任务上通用大模型仍占优。

什么时候值得做领域适配？

语料规模

内部语料要达到十亿级 token 量级，否则 DAPT 收益有限

任务独特性

越是专有语言、内部库、私有方法学，收益越大

使用频次

高频、全员使用的场景才摊得平前期训练成本

评测能力

必须先有可信的领域评测集，否则无法判断是不是变好

四条都满足，才建议自建领域模型；否则先用 RAG。

ChipNeMo 发现（二）：检索器也需要领域适配

RAG 的瓶颈往往不在生成模型，而在“有没有把对的文档取回来”

工程师提问 → **检索器 召回相关文档** → 拼接上下文 → 模型生成回答

真正的瓶颈在这一步

- **通用检索器不懂芯路术语。**同义词、缩写、内部代号在通用嵌入空间里距离很远，召回率因此偏低。
- **用领域数据微调检索器。**论文报告：检索命中率显著提升，并直接带来生成质量的进一步改善。
- **检索质量决定回答上限。**取回错误文档时，模型会用一本正经流畅的语言给出错误答案 —— 这比不回答更危险。

课程实验里可以直接复用的做法

- 把设计规范、方法学文档、工具手册切块入库，先跑通一个最小 RAG。
- 构造 30–50 条带标准答案的问题，作为固定评测集。
- 对比“有无检索”“检索器是否微调”三种配置，量化每一步的增量。

结论：先把检索对，再考虑把模型做大。

ChipNeMo 三大应用与评测结果

三个场景覆盖了“问、写、读”——正是工程师日常最耗时的三件事

1 工程助手问答

回答设计规范、方法学与内部工具相关问题

专家评分 7.4 / 10

2 EDA 脚本生成

生成内部工具与专有语言的脚本，通用模型最吃力的场景

正确率 > 50%

3 Bug 摘要与分析

把冗长的缺陷记录与讨论压缩成可读摘要

专家评分 4.5 / 7

怎么读这三个数字？

- **7.4 / 10 是“可用但需核对”。**足以当查询入口，不足以当权威答案。
- **> 50% 正确率听起来不高，但它对应的是通用模型几乎做不了的任务 —— 参照系变了，结论就变了。**
- **论文自己也强调仍有改进空间。**把它读成“方向被验证”，而不是“问题已解决”。

案例四 · 商用 EDA 中的 AI：从论文走进产线

同样是强化学习，落点从“布局”扩展到“整条流程的参数与策略”

Synopsys DSO.ai

- 用强化学习在综合与布局布线的巨大参数空间中自主搜索 PPA 最优解
- 2021首个客户流片；2023公布“首个 100 例商业流片”里程碑
- 公开客户案例：生产率提升 3 倍以上、总功耗最多降低 25%、芯片面积可观缩小。

Cadence Cerebrus

- 机器学习驱动 RTL-to-signoff 全流程，官方宣称最多 3× 设计生产率与 20% PPA 改善。
- 客户案例：单名工程师 10 天内完成 3.5 GHz 移动 CPU 的实现收敛。
- 同类还有 Siemens 在物理验证与变差分析上的 AI 能力。

设定 PPA 目标 → **生成一组工具配置** → 并行跑综合与 P&R → 评估 QoR → **更新搜索策略**

商用工具的通用 AI 优化循环（重绘示意）：本质是把工程师的调参经验变成可搜索的策略

读厂商数据时要小心：“最多”“up to”通常是最好情况；缺少统一基准与第三方复现，横向对比几乎不可能。把它当作“方向已被产业验证”的证据，而不是精确的性能承诺。

案例五 · 大模型与 RTL：从能生成到能评测

没有可信评测集，“生成能力”就无法被讨论 —— 这是这一方向的真正起点

VerilogEval（NVIDIA，ICCAD 2023）

- **题库。**取自 HDLBits 教学网站，人工描述版 156 题，机器生成描述版 143 题，覆盖组合逻辑到复杂状态机。
- **判定。**在沙箱中用 iVerilog 仿真，把生成代码的波形与标准答案比对，功能等价才算通过。
- **指标。**pass@k：每题采样 20 次，只要有一次通过即算通过。

领域里的其他基准与方法

- **RTLLM。**50 个更接近真实模块的设计题，考察规模更大的生成任务。
- **RTLCoder / VeriGen 等。**用合成数据做监督微调，让开源小模型在 Verilog 上追赶闭源大模型。
- **合成数据自举。**VerilogEval 论文本身也验证了：用模型生成的“题目—代码”对做微调，可显著提升生成能力。

真正有工程价值的形态：把生成放进闭环

自然语言规格 → LLM 生成 RTL → 自动跑仿真 / 形式检查 → 失败则回灌错误信息 → 重新生成直到通过

关键不在“一次写对”，而在“能不能自动判定对错并迭代”——判定器才是这条链路的核心资产。

案例六 · AI 用于验证：最大的工作量，最保守的采用

验证占整体工时六成以上，却是 AI 渗透最慢的环节 —— 因为责任无法转移

已经在做的

- 用覆盖率反馈引导激励生成，减少无效随机用例
- 从设计文档与代码中挖掘候选断言，交人工确认
- 回归失败的自动聚类与根因摘要，缩短调试时间
- 在大规模回归中预测哪些用例最可能失败，优先跑

仍然做不到的

- 证明“没有遗漏的场景”——覆盖率高不等于验证充分
- 替代形式验证给出的数学保证
- 承担签核结论的责任：出了问题，模型不会被追责
- 在没有黄金参考模型时判断“什么才是对的”

为什么验证是飞轮上最慢的一环？

- 生成任务允许出错——错了重来一次成本很低；验证任务不允许漏判——漏一次就是一次流片失败。
- 所以 AI 在“提出候选”上很有价值，在“下最终结论”上短期内不会被采用。

六个案例的横向对比

看清每个案例站在“流程”的哪个位置、用了哪一类方法

案例	介入环节	方法类型	层次	最硬的证据	最大的软肋
AlphaChip	物理设计 · 宏单元布局	强化学习 + 图神经网络	L2	三代 TPU 量产采用	对照基准与可复现性争议
PRIME	架构探索 · 参数选择	离线数据驱动优化	L2	面积与延迟同时改善	对照的是通用[illegible]片
ChipNeMo	问答 / 脚本 / 摘要	领域自适应大模型 + RAG	L1	13B 追平 70B	评分仍未达生产级权威
DSO.ai / Cerebrus	综合与 P&R 全流程	强化学习参数搜索	L2	百例级商业流片	厂商数据，缺少第三方复现
VerilogEval 生态	RTL 生成	大模型 + 自动判定闭环	L1→L3	有了可复现的公开基准	真实模块规模仍偏小
AI 辅助验证	功能验证	覆盖率引导 / 断言挖掘	L1	确实减少无效用例	无法[illegible]担签核责任

共同规律：AI 目前最擅长“在明确目标下搜索”和“在海量文本中检索归纳”，最不擅长“对最终正确性负责”。

北京大学
PEKING UNIVERSITY

PART 04 | Tool Stack & Boundaries

工具栈与能力边界

当前 AI 工具栈：四层视图

自下而上依赖 —— 没有干净的数据与可复现的实验管理，上面三层都建在沙子上

L4	应用与智能体层	面向具体任务的助手与流程编排：代码助手、验证助手、多步骤自动化 Agent
L3	领域模型层	模型与评测基准：领域大模型、RAG 知识库、专用评测集
L2	通用基础模型层	提供底层推理与代码能力的通用大模型，通过提示与工具调用接入设计流程
L1	数据与基础设施层	设计数据、EDA 日志与报告、算力集群、版本与实验管理

课程建议：从 L1 与 L2 起步 —— 先把数据整理干净、把评测集建起来，再谈模型。

AI 在设计全流程上的落地地图

越靠近“生成”越成熟，越靠近“签核”越保守 —— 因为责任无法转移给模型

架构探索 中

设计空间搜索、性能建模与快速估算

代表案例：PRIME

RTL 设计 高

代码补全、模块生成、跨语言转换

代表案例：Verilog 生态

功能验证 中

激励生成、断言挖掘、失败聚类

代表案例：AI 辅助验证

逻辑综合 中

约束推荐、PPA 多目标参数寻优

代表案例：DSO.ai / Cerebrus

物理设计 中

布局规划、拥塞预测、时序收敛辅助

代表案例：AlphaChip

签核与 DFT 低

报告解读、缺陷定位、根因摘要

代表案例：尚无标志性案例

右上角为成熟度判断（高 / 中 / 低），仅代表 2025 年前后的工程可用程度

能力边界：AI 今天能做什么，不能做什么

生成不是瓶颈，验证才是

已经足够好用

- 写模板化、结构重复的 RTL 与 testbench 草稿
- 生成与解释 EDA 脚本、约束文件、构建流程
- 把长报告、日志、缺陷记录压缩成可读摘要
- 在明确目标函数下做参数寻优与方案排序
- 作为查询接口，快速定位方法学与规范条款

仍必须由人把关

- 功能正确性判定 —— 模型会给出看起来对的错误代码
- 架构级权衡与需求取舍，这是价值判断而非搜索问题
- 签核结论与流片决策，责任无法转移给模型
- 专有 IP 与设计数据的合规使用与外泄风险控制
- 结果可复现性：随机性与版本漂移必须被显式固定

核心判断：谁能低成本地把结果正确，谁才能真正把 AI 用起来。
所以这门课后半程的重点，是评测与验证，而不是生成。

工具栈明细：今天都能用到什么

课程实验从“开源流程 + 通用大模型”起步 —— 可复现、零成本，且足以走通完整闭环

类别	代表工具 / 项目	典型用途	接入门槛
商用 EDA 的 AI 能力	Synopsys DSO.ai、Cadence Cerebrus、Siemens	在既有流程内做 PPA 自动寻优与设计空间探索	高（商用授权）
开源设计流程	OpenROAD、OpenLane、Yosys、iEDA	端到端 RTL-to-GDS 实验平台，适合教学与研究	低（免费可跑）
RL 布局开源实现	Circuit Training（AlphaChip）、MacroPlacement 基准	复现与对比宏单元布局方法	中
通用大模型	Claude、GPT、Gemini 等及其编程助手	RTL 草稿、脚本、文档解读、报告摘要	低
领域模型与评测集	ChipNeMo、VerilogEval、RTLLM、RTLCoder	专有语言与方法学任务；客观评测模型好坏	中
智能体与编排	工具调用框架、MCP 类协议、CI 流水线集成	把“生成—仿真—修复”串成可重复闭环	中
高层次设计语言	Chisel、SpinalHDL、HLS（C/C++ 综合）	抬升描述抽象层次，与生成式方法天然互补	中

第 3 周起，我们统一使用 通用大模型接口完成全部实验。