INTRODUCTION TO RISC-V

RISC-V入门教程

ISA | 汇编指令 | 系统编程 | 组成原理 | 嵌入式应用

RISC-V简介

主讲 邢建国

中国开放指令生态（RISC-V）联盟 | 浙江中心
China RICS-V Alliance | Zhejiang Center

浙江图灵算力研究院
ZHEJIANG TURING INSTITUTE

RISC-V简介

- RISC-V历史
- RISC-V设计目标
- RISC-V设计原则
- RISC-V发展及未来

RISC-V的历史

RISC-V是一种基于精简指令集计算（RISC）原则的开源指令集架构（ISA），由加州大学伯克利分校于2010年启动，并在2015年成立了非盈利组织RISC-V基金会进行维护。

RISC-V的名称中的“V”代表第五代RISC架构，同时也象征着变化（Variation）和向量（Vectors）

RISC-V的历史

2010年加州大学伯克利分校的实验室项目需要一个易于实施的、高效的、可扩展的，且与他人分享时不受限制的指令集。

没有一个现成的指令集满足以上需求。

在 David Patterson教授的支持下, Krste Asanovic教授和Andrew Waterman、Yunsup Lee等创建了RISC-V指令集架构。

2014年该指令集架构一经公开，便迅速在全球范围内受到广泛欢迎。




RISC-V是什么

一款高质量，免许可证，开放的RISC指令集架构

一套由非营利的 RISC-V 基金会维护的标准: https://riscv.org/

适用于所有类型的计算系统：从微控制器到超级计算机

RISC-V 不是一家公司，也不是一 款 CPU 实现

设计目标

要适合设计各种规模的处理器
包括从最小的嵌入式控制器，到最快的高性能计算机。

要兼容各种流行的软件栈和编程语言。

要适用于所有实现技术
包括FPGA（Field-Programmable Gate Array，现场可编程逻辑门阵列），ASIC（Application-Specific Integrated Circuit，专用集成电路），全定制芯片，甚至未来的制造元件技术。

能用于高效实现所有微体系结构
包括微程序或硬连线控制、顺序、解耦或乱序流水线、单发射或超标量等。
支持高度定制化，成为定制加速器的基础，以应对摩尔定律的放缓。

要稳定，基础ISA不会改变。
不能像以往的公司专有ISA那样消亡，包括AMD的Am29000，Digital 的Alpha 和 VAX，Hewlett Packard2的PA-RISC，Intel 的 i860 和 i960，Motorola 的 88000，以及 Zilog 的 Z8000。

RISC-V特点

完全开源

采用宽松的BSD协议，允许企业自由使用并添加自有指令集，而不必开放共享。

模块化设计

RISC-V的设计模块化，支持通过标准或非标准扩展来增强其功能，例如整数运算、浮点运算、向量运算等。这种灵活性使得RISC-V能够适应各种不同的应用场景，从嵌入式系统到高性能计算。

开源与闭源

完全开源

采用宽松的BSD协议，允许企业自由使用并添加自有指令集，而不必开放共享。

领域	开放的标准	开源的实现	闭源的实现
操作系统	POSIX	Linux, FreeBSD, ……	Windows, ……
编译器	C	Gcc, LLVM, ……	Intel icc, ARMcc, … …
数据库	SQL	MySQL, ……	Oracle, DB2, ……
ISA	???	……	X86, ARM, ……

开源许可证

- 开源许可证一般包括：
  - （1）提及原作者
  - （2）法律免责
  - （3）共享方式，特别是衍生作品的共享方式

- 常用的开源许可证
  - GPL： 衍生作品也要使用GPL，并且开源。GCC、Linux内核
  - LGPL：对修改部分开源，自有代码不需要。Arduio
  - Apache：衍生作品可以以任何形式发布（包括闭源），可以商业应用，提供了明确的专利授权。
  - BSD：允许用户自由地使用、修改和分发软件，同时也允许在商业软件中使用BSD许可证下的代码，没有提供专利授权。OpenSSH
  - 知识共享（Creative common）
    - Creative Common Atribution (CC BY) : 提及原作者，衍生作品不需要。RISC-V 官方文档
    - Creative Common Atribution-ShareAlike(CC BY-SA)：提及原作者，衍生作品同样要遵循。

模块化设计与增量化设计

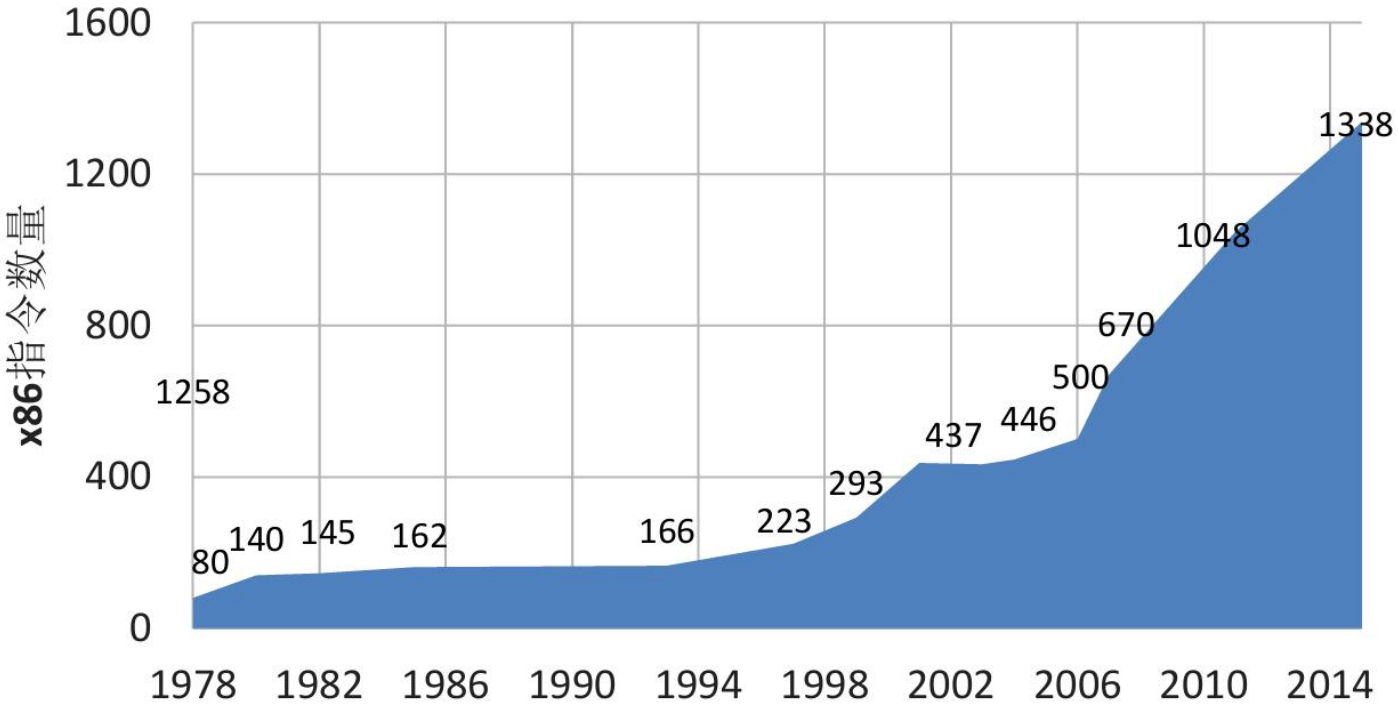
增量化设计

计算机体系结构的传统发展方式是增量型ISA，这意味着新处理器不仅需要实现新的ISA 扩展，还必须实现过去的所有扩展。其目的是保持向过去的二进制兼容性， 从而使数十年前的二进制程序仍可在最新处理器上正确运行。出于市场营销目的，新 一代处理器的发布通常伴随着新指令的发布。
这两点需求共同导致ISA的指令数量随时间大幅增长。

模块化设计

RISC-V的设计模块化，支持通过标准或非标准扩展来增强其功能。

这种灵活性使得RISC-V能够适应各种不同的应用场景，从嵌入式系统到高性能计算。



RISC-V设计原则

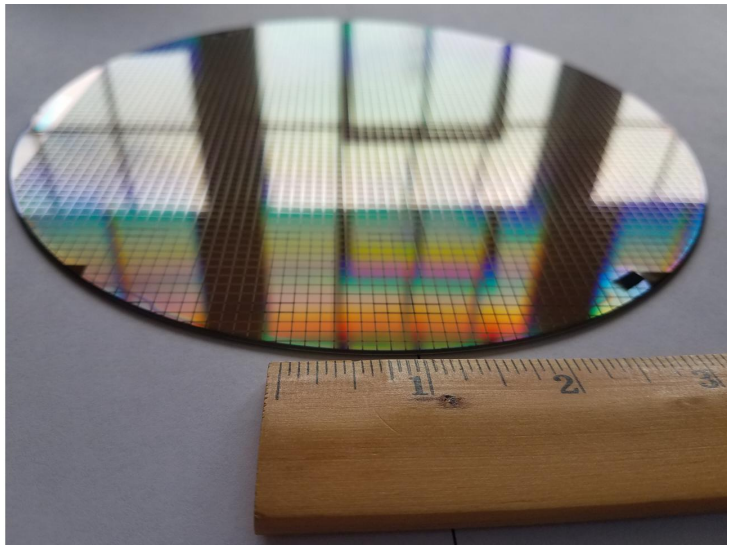
- 成本
- 简洁
- 性能
- 架构和实现分离
- 提升空间
- 代码大小
- 易于编程/编译/链接

RISC-V设计原则： 成本

处理器以集成电路的形式实现，通常称为芯片（chip）或晶粒（die）。成本对晶粒面积十分敏感。晶粒越小，有缺陷的晶粒比例越低。保持ISA 的简洁性，从而缩小相应处理器的尺寸。

将缓存 大小（16KiB）和制造工艺（TSMC40GPLUS）均相同的 RISC-V Rocket 处理器与 ARM-32 Cortex-A5 处理器进行对比。RISC-V 晶粒的大小是0.27mm^2，而 ARM-32 晶粒的大小是0.53mm^2。由于面积大将近一倍，ARM-32Cortex-A5 的晶粒成本大约 是RISC-V Rocket 的 4（22）倍。即使晶粒大小只减小10%，成本也会降低为 81% （0.92）。

$$\text{成本} \approx f(\text{晶粒面积}^2)$$



RISC-V设计原则：简洁

鉴于成本对复杂性十分敏感，架构师需要一款简洁的ISA来减小晶粒面积。

简洁的ISA也能节省芯片设计和验证的时间，而这可能占芯片开发成本的主要部分。

简单指令通常比复杂指令更常用。

ARM-32 ISA指令ldmiaeq：
该指令表示LoaD Multiple, Increment-Address, on EQual。它会在EQ 条件码置位时从内存读入5次数据，并写入6个寄存器。此外，它还将结果写入PC，从而执行条件分支。

```
ldmiaeq SP!, {R4-R7, PC}
```

RISC-V设计原则：性能

在每个程序中简洁ISA需要执行的指令比复杂ISA多，但前者能通过更高的时钟频率或更小的每指令平均周期数（Cycles Per Instruction，CPI）来弥补。

$$\frac{\text{指令数}}{\text{程序}} \times \frac{\text{平均时钟周期数}}{\text{指令}} \times \frac{\text{时间}}{\text{时钟周期}} = \frac{\text{时间}}{\text{程序}}$$

CPU性能铁律

运行 CoreMark 基准测试 [Gal-On and Levy 2012] （100000 次迭代）的性能

$$\frac{32.27\ B\ \text{指令数}}{\text{程序}} \times \frac{0.79\ \text{时钟周期数}}{\text{指令}} \times \frac{0.71\ ns}{\text{时钟周期}} = \frac{18.15\ s}{\text{程序}}$$

ARM-32 Cortex-A9

$$\frac{29.51\ B\ \text{指令数}}{\text{程序}} \times \frac{0.72\ \text{时钟周期数}}{\text{指令}} \times \frac{0.67\ ns}{\text{时钟周期}} = \frac{14.26\ s}{\text{程序}}$$

RISC-V BOOM

RISC-V设计原则：架构和实现分离

- 架构和实现之间的最初区别可追溯到1960年代，即：架构是机器语言程序员为了编写正确的程序所需了解的知识，而不是为了提升程序性能。
- 对于架构师，一项诱人的方案是在ISA中添加指令，来优化特定时期某个实现的性能和成本，但这会给其他不同或将来的实现带来负担。
- 架构师除了不应加入那些仅有助于一个实现的功能，也不应加入阻碍某些实现的功能。

MIPS-32 ISA 的延迟分支。

```
Loop:   LD      F0,0(R1)        ;F0=vector element
        ADDD    F4,F0,F2        ;add scalar from F2
        SD      0(R1),F4        ;store result
        SUBI    R1,R1,8         ;decrement pointer 8B (DW)
        BNEZ    R1,Loop         ;branch R1!=zero
        NOP                     ;delayed branch slot
```

RISC-V设计原则：提升空间

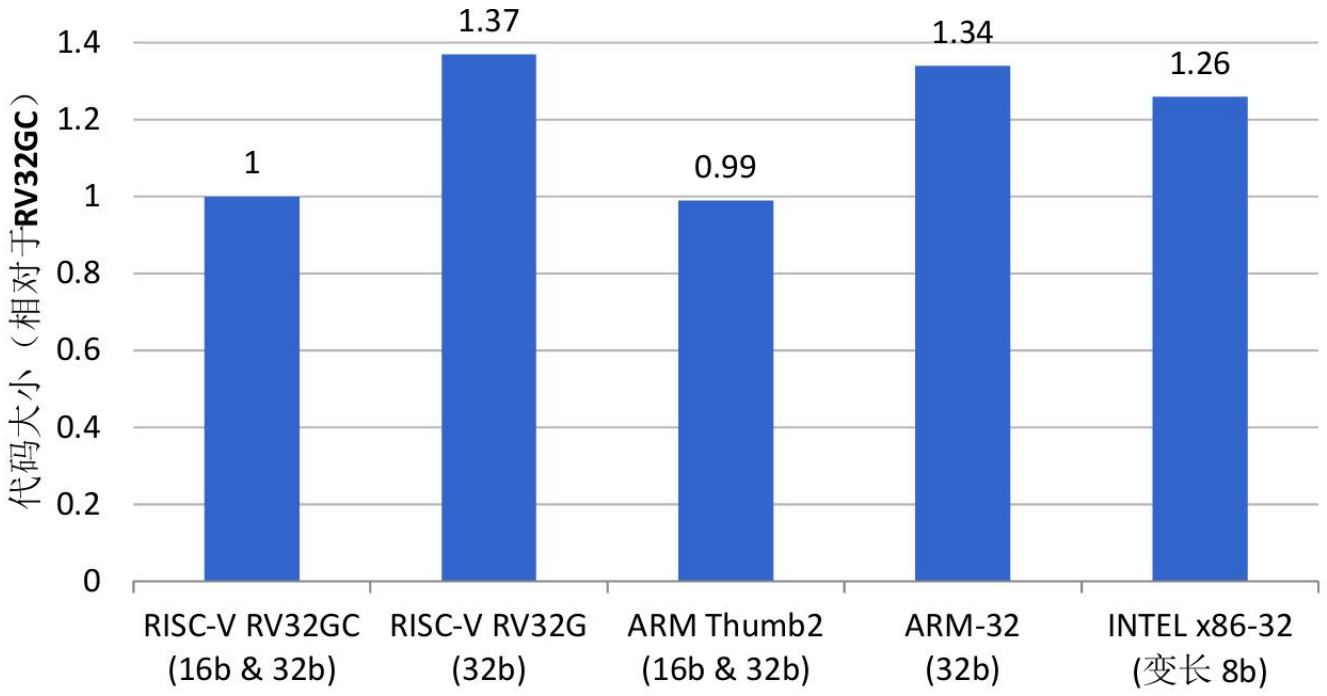
随着摩尔定律（Moore’slaw）终结，大幅提高性价比的唯一途径是为特定领域（如深度学习，增强现实，组合优化，图形等）添加自定义指令。

ISA必须为将来的扩展预留操作码空间。

ARM-32架构师后来试图通过向以前统一的32 位 ISA 中添加16位指令来缩减代码大小，但发现无可用空间。因此，唯一的解决方案是先设计一款16位指令的新ISA（Thumb），后来又设计了一款同时支持16位和32 位指令的新ISA（Thumb-2），并通过一个模式位在它们和 ARM ISA 之间切换。

RISC-V设计原则：代码大小

- 程序越小，程序存储器所需芯片面积越小，这对嵌入式设备是一项巨大的成本。ARM架构师在Thumb和Thumb-2 ISA 中追加一些更短的指令。
- 更小的程序还能减少指令缓存的缺失次数，从而降低功耗（访问片外DRAM 的能耗远高于访问片上SRAM）并提升性能。
- 让代码更短是ISA架构师的目标之一。



RISC-V设计原则：易于编程/编译/链接

- 由于访问寄存器比访问内存快得多，因此编译器必须做好寄存器分配工作，这在寄存器数量较多时更简单。

  ARM-32有16个寄存器，而x86-32 只有 8 个。大多数现代ISA（包括RISC-V）都有相对较多的32个整数寄存器。更多寄存器显然能让编译器和汇编语言程序员的工作更轻松。

- 编译器和汇编语言程序员面临的另一个问题是估计代码序列的执行速度。复杂指令和内存操作数使处理器设计者难以实现性能的可预测性。

  每条RISC-V 指令通常最多只需要一个时钟周期程序越小，ARM-32 和x86-32 有些指令的执行仍需多个时钟周期。此外，与ARM-32 和RISC-V 不同，x86-32 算术指令的操作数可位于内存中，而不要求都在寄存器中。

- ISA 支持位置无关代码（Position Independent Code，PIC）有助于支持动态链接，因为共享库代码在不同程序中可位于不同地址。PC相对分支和数据寻址有利于PIC。

  几乎所有ISA都提供PC相对寻址的分支指令，但x86-32和MIPS-32 缺少 PC 相对数据寻址。

RISC-V设计原则

	过去的错误设计			吸取的经验教训
	ARM-32（1986）	MIPS-32（1986）	x86-32（1978）	RV32I（2010）
成本	整数乘法必选	整数乘除法必选	8 位和 16 位操作。整数乘除法必选	无 8 位和 16 位操作。整数乘除法可选（RV32M）
简洁	无零寄存器。条件执行指令。寻址模式复杂。栈指令（push/pop）、算术/逻辑指令中可移位	立即数有零扩展和符号扩展。部分算术指令会触发溢出自陷	无零寄存器。过程调用/返回指令（enter/leave）复杂。栈指令（push/pop）。寻址模式复杂。循环指令	零寄存器 x0。立即数仅符号扩展。寻址模式唯一。无条件执行。无复杂调用/返回指令和栈指令。算术溢出无自陷。独立的移位指令
性能	分支指令条件码。指令格式中源/目的寄存器位置不固定。多字读数。立即数需计算。PC 作为通用寄存器	指令格式中源/目的寄存器位置不固定	分支指令条件码。二操作数指令	比较-跳转指令（无条件码）。三操作数指令。无多字读取。指令格式中源/目的寄存器位置固定、立即数为常数。PC 不是通用寄存器
架构和实现分离	像通用寄存器般写入 PC 暴露流水线长度	延迟分支。延迟取数。乘除法专用的 HI 和 LO 寄存器	部分寄存器不通用（AX、CX、DX、DI、SI 有特殊用途）	无延迟分支。无延迟读数。通用寄存器
提升空间	可用操作码空间有限	可用操作码空间有限		可用操作码空间丰富
程序大小	仅 32 位指令（Thumb-2 为独立 ISA）	仅 32 位指令（microMIPS 为独立 ISA）	指令长度可变，但选择很少	32 位指令 +16 位 RV32C 扩展
易于编程/编译/链接	仅 15 个寄存器。内存数据必须对齐。寻址模式不规则。性能计数器不一致	内存数据必须对齐。性能计数器不一致	仅 8 个寄存器。无 PC 相对数据寻址模式。性能计数器不一致	31 个寄存器。数据不必对齐。PC 相对数据寻址模式。数据寻址模式对称。性能计数器在架构中定义

RISC-V的发展

- RISC-V 成员已超过 700 个，分布在全球 50 个国家
- Premier成员（25）、Strategic成员 (170)、Community (199)
- **https://riscv.org/members/**

RISC-V厂商及产品

SiFive

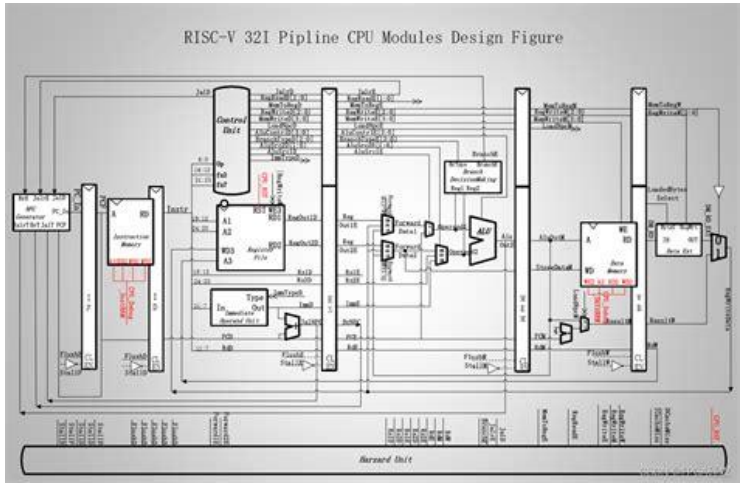
SiFive是全球最早生产基于RISC-V架构芯片的公司之一，与多家全球领先的半导体制造商合作，提供高性能、低功耗的计算解决方案，应用于可穿戴设备、数据中心、边缘计算、汽车和航空航天等领域。

Codasip

Codasip：Codasip提供定制化的RISC-V CPU IP核心，通过其独特的架构描述语言CodAL进行设计，支持标准扩展和自定义指令，以满足不同客户的需求。

西部数据

西部数据（Western Digital Corporation）：西部数据与SiFive合作，每年向市场供应约10亿个RISC-V核心，用于其存储解决方案



RISC-V厂商及产品

兆易创新

兆易创新是中国知名的半导体公司，也在RISC-V领域进行了布局，推出了多款基于RISC-V架构的芯片产品

阿里平头哥

T-Head（阿里巴巴集团）致力于开发基于RISC-V的系统级芯片，例如T-head RVB-ICE开发板，其产品主要面向工业和汽车市场

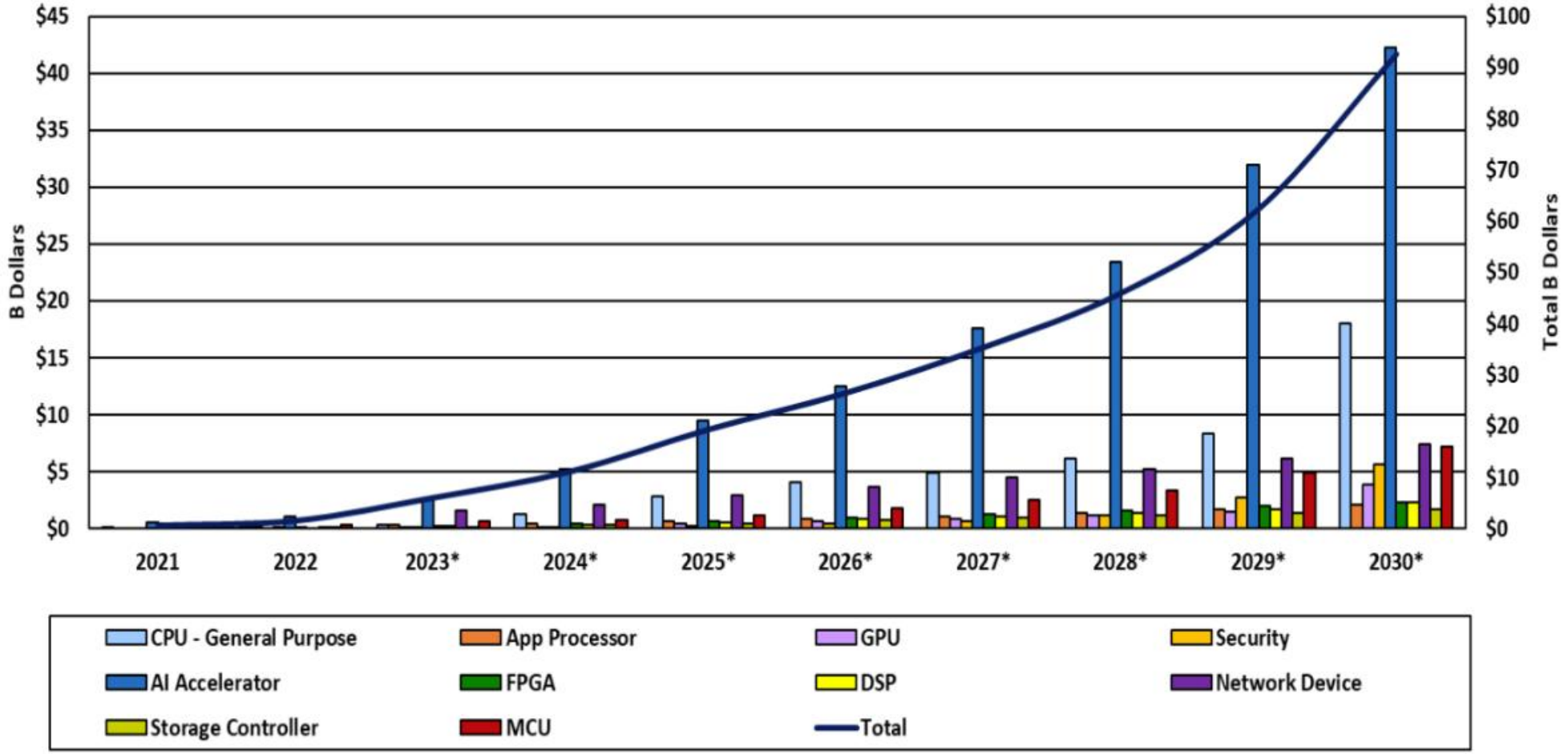
芯来科技

芯来科技：作为国内RISC-V IP企业之一，芯来科技提供多款RISC-V IP产品，并支持开源软件的开发和推广

RISC-V的发展

- RISC-V市场预计将在2030年达到927亿美元
- RISC-V SoC的年度出货量预计在2030年达到162亿个
- 到2030年，RISC-V处理器在全球市场的份额预计将超过四分之一
- 预计到2030年，中国RISC-V芯片市场规模将达到250亿美元。
- 由于对人工智能的需求增加，RISC-V技术在AI领域的应用预计将在未来几年内实现爆炸式增长。

RISC-V的发展



*Forecast

Source: The SHD Group, January 2024

RISC-V简介

- RISC-V历史
- RISC-V设计目标
- RISC-V设计原则
- RISC-V发展及未来