2024 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION JAPAN

RISC-Vの歩き方

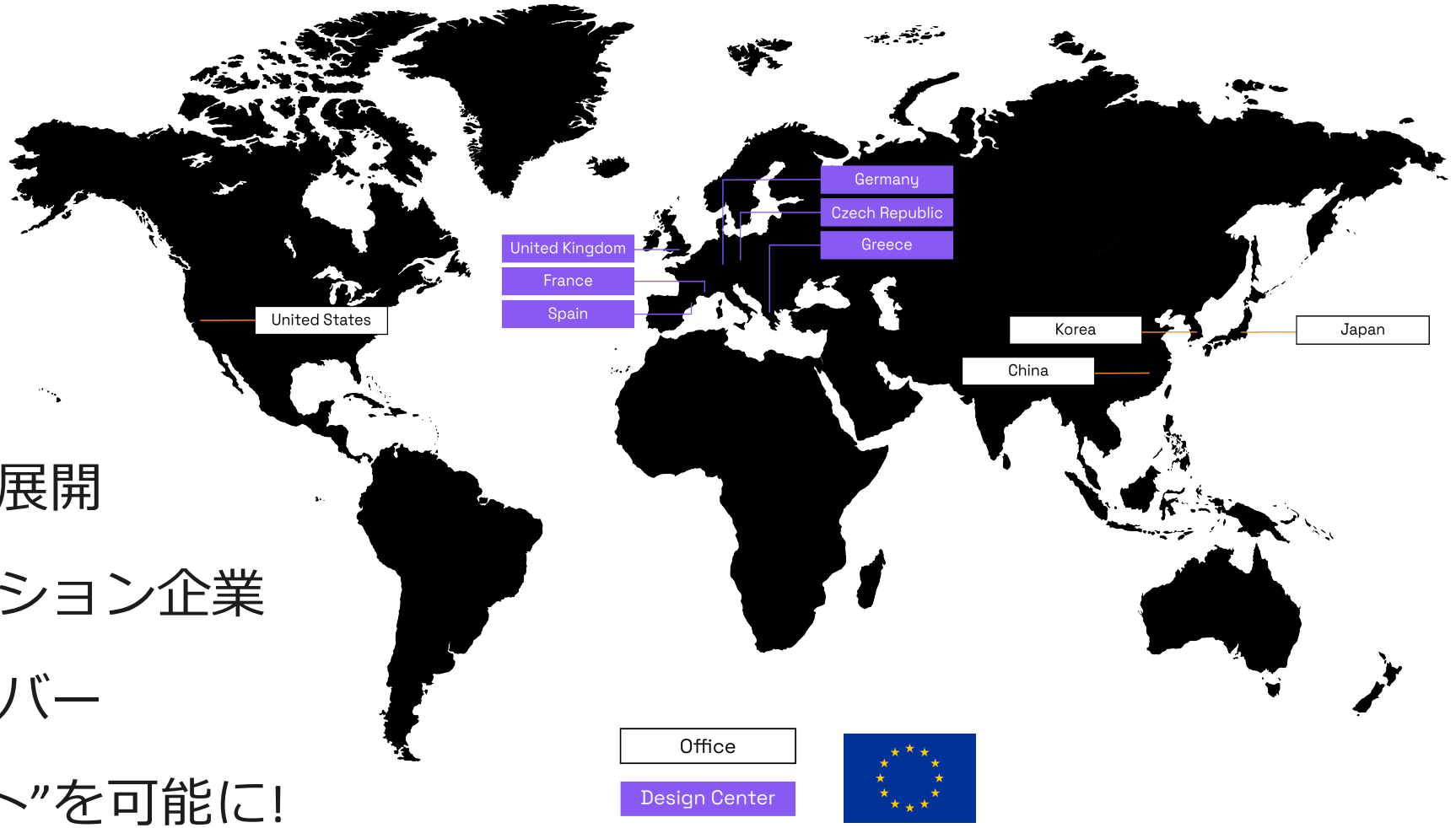
Takaaki Akashi, Country Manager - Japan

Codasip

accellera SYSTEMS INITIATIVE

→ コダシップ概要

- 2014年設立
- 独ミュンヘン本社
- 約250名の社員
- ヨーロッパで開発
- グローバルなオフィス展開
- プロセッサ・ソリューション企業
- RISC-V創設メンバー
- “カスタム コンピュート”を可能に!



→ なぜRISC-V？

Codasip

オープンソースIPがあるから

ARMより安価で導入できそう

カスタマイズできるから

命令セットがオープンだから

タダだから

複数のIPベンダから選べるから

オープンソースだから

なんとなく

自作できるから

Codasip

↘

RISC-Vのおさらい

→ ウェビナー: RISC-Vに関する誤解のトップ10

1. Linuxがオープンソースのオペレーティングシステムであるように, RISC-Vはオープンソースのプロセッサである
2. すべてのRISC命令セット・アーキテクチャ(ISA)は同等です
3. 新しいオープンなRISC-Vを選ぶより, 確立されたベンダ固有のISAを選ぶ方が安全なビジネス判断です
4. RISC-V ISAは, 安いから人気が出ているだけです
5. ベンダ固有ISAのソフトウェアのエコシステムは, 断片化しない
6. モジュール化されたRISC-Vは断片的なソフトウェア・エコシステムとなる
7. RISC-Vは組み込み用途にしか向かない
8. RISC-Vは, ベンダ固有ISAほど安全ではない
9. RISC-Vは常にベンダ固有ISAの性能とエコシステム堅牢性を追いかけるだけ
10. 以上の点から, RISC-Vが支配的なISAになることはありえない

見逃した方へ: 録画があります
https://attendee.gotowebinar.com/recording/6855805075368559965

→ 1. Linuxがオープンソースのオペレーティングシステムであるように, RISC-Vはオープンソースのプロセッサである

RISC-V is an open-source processor, like Linux is an open-source operating system.

Linux has a single master open-source code base you can download, while RISC-V is an open of the hardware/software interface for which there are many different im*specification*plementations. A better analogy than Linux is Ethernet, since both Ethernet and RISC-V are free and open specifications.

Before the Ethernet standard, companies had their own proprietary local area networks (LANs): Apple AppleTalk (1985), Datapoint ARCNET (1977), Digital Equipment Corporation DECnet (1975), IBM Token Ring (1984), Xerox Ethernet (1974), and so on. In 1980 Digital Equipment Corporation, Intel, and Xerox ("DIX") joined forces to create a local network standard based on Ethernet. They also created an organization—IEEE 802.3 working group—that has advanced the Ethernet standard over the past four decades.

Ethernet made rapid advances in cost and performance because many companies could build network products that ran the same software stack on top of the Ethernet standard. While you can design your own Ethernet switches and there could be open hardware designs to download, many simply buy switches that meet the Ethernet standard. A decade after the creation of the "DIX" standard, Ethernet became the dominant networking technology. Today, proprietary LANs are practically extinct. Does anyone miss them?

The popular Universal Serial Bus (USB) also followed the Ethernet game plan by providing a free and open standard for peripheral interconnect that is embraced by many companies plus an organization to evolve it.

Like Ethernet and USB, RISC-V is an open standard that lets *many* organizations design hardware, which fosters competition to improve its cost-performance and develop a rich shared software ecosystem that offers RISC-V products in many markets. Like Ethernet and USB, RISC-V also has a foundation that evolves the standard over time to meet new demands. Like Ethernet and USB, you can buy RISC-V hardware, build it yourself, license designs, or download open-source designs.

Linuxには、ダウンロードできる単一のマスターオープンソースコードベースがありますが、**RISC-Vは、さまざまな実装があるハードウェア/ソフトウェアインターフェイスのオープン仕様**です。RISC-VはEthernetのように自由でオープンな仕様であるため、LinuxよりもEthernetの歴史が参考になります。

Ethernet規格が登場するまでは、Apple社AppleTalk(1985年)、Datapoint社ARCNET(1977年)、Digital Equipment Corporation社DECnet(1975年)、IBM社Token Ring(1984年)、Xerox社Ethernet(1974年)など、企業独自のローカルエリアネットワーク(LAN)が存在していました。1980年、Digital Equipment Corporation社、Intel社、およびXerox社(「DIX」)が協力して、Ethernetに基づくローカルネットワーク標準を作成しました。また、彼らはIEEE802.3ワーキンググループを創設し、現在まで過去40年間にわたりEthernet規格を発展させてきました。

Ethernetは、多くの企業がEthernet標準の上で同じソフトウェアスタックを実行するネットワーク製品を構築できるため、コストとパフォーマンスの面で急速な進歩を遂げました。Ethernetスイッチは自身で独自の設計、開発することもできますし、オープンなハードウェアデザインをダウンロードして活用することもできますが、多くの人は単にEthernet標準に適合するスイッチを購入しています。「DIX」標準の策定から10年後、Ethernetは主要なネットワーキングテクノロジーになりました。今日、独自のLANは事実上消滅しています。独自LANを懐かしいと思う人はいらっしゃいますか？

非常にポピュラーなUSB（Universal Serial Bus）も、Ethernetに倣って、周辺機器の相互接続のためのフリーでオープンな標準を提供し、多くの企業に受け入れられ、それを進化させる組織も活動しています。

EthernetやUSBと同様に、**RISC-Vは多くの組織がハードウェアを設計できるようにするオープン規格であり**、**コストパフォーマンスを向上**させ、多くの市場でRISC-V製品を提供する**豊富な共有ソフトウェアエコシステム**を開発するための**競争を促進**します。EthernetやUSBと同様に、RISC-Vもまた、新しい需要に対応するために、時とともに規格を進化させる基盤を持っています。EthernetやUSBと同様に、**RISC-Vハードウェアの購入**、**自作**、**デザインのライセンス供与**、**オープンソースのデザインのダウンロード**が可能です。

→ 1. Linuxはオープンソースのオペレーティングシステムですが, RISC-V はオープンスタンダートISAである

Linux is an open-source operating system, but not like it, RISC-V is an open-standard ISA.

オープンソースは、ソフトウェアやプログラムのソースコードが誰にでも公開され、
自由に使用・改変・再配布できる仕組みです

Linux

C++

APACHE

MySQL

LLVM COMPILER INFRASTRUCTURE

openstack™

git

TensorFlow™

※ オープンソースライセンスは数千種類ある

オープンスタンダードは、誰でもアクセス可能な仕様であり、競争やイノベーションを促進し、相互運用性と持続可能性を確保します

- オープンスタンダードのEthernetで出来ること
  - オープンソースIPをダウンロードしてChip化
  - 商用IPを購入してChip化
  - 自身で独自の設計、Chip化
  - ベンダが開発したChip/Boardを購入
  - 市販スイッチを購入

IEEE 802

IEEE 802.1: ネットワーク機能およびプロトコルのアーキテクチャ
IEEE 802.3: イーサネット（有線）
IEEE 802.11: Wi-Fi（無線）
IEEE 802.11a: 5GHz帯域を使用する高速無線
IEEE 802.11b: 2.4GHz帯域を使用する低速無線
IEEE 802.11g: 2.4GHz帯域を使用する高速無線
IEEE 802.11n: MIMO（Multiple-Input Multiple-Output）
IEEE 802.11ac: 5GHz帯域を使用する高速無線（Wi-Fi5）
IEEE 802.11ax: OFDMA（Wi-Fi6）
IEEE 802.11ay: 60GHz帯域超高速無線
IEEE 802.11ad: 60GHz帯域高速無線（WiGig）
IEEE 802.15: WPAN
IEEE 802.16: WiMAX
IEEE 802.22: WRAN
IEEE 802.24: スマートグリッド通信
IEEE 802.3ae: 10GB
IEEE 802.3ba: 40GB/100GB
IEEE 802.1X: ネットワーク認証およびポート制御

SUPERSPEED+ USB CERTIFIED

Bluetooth®

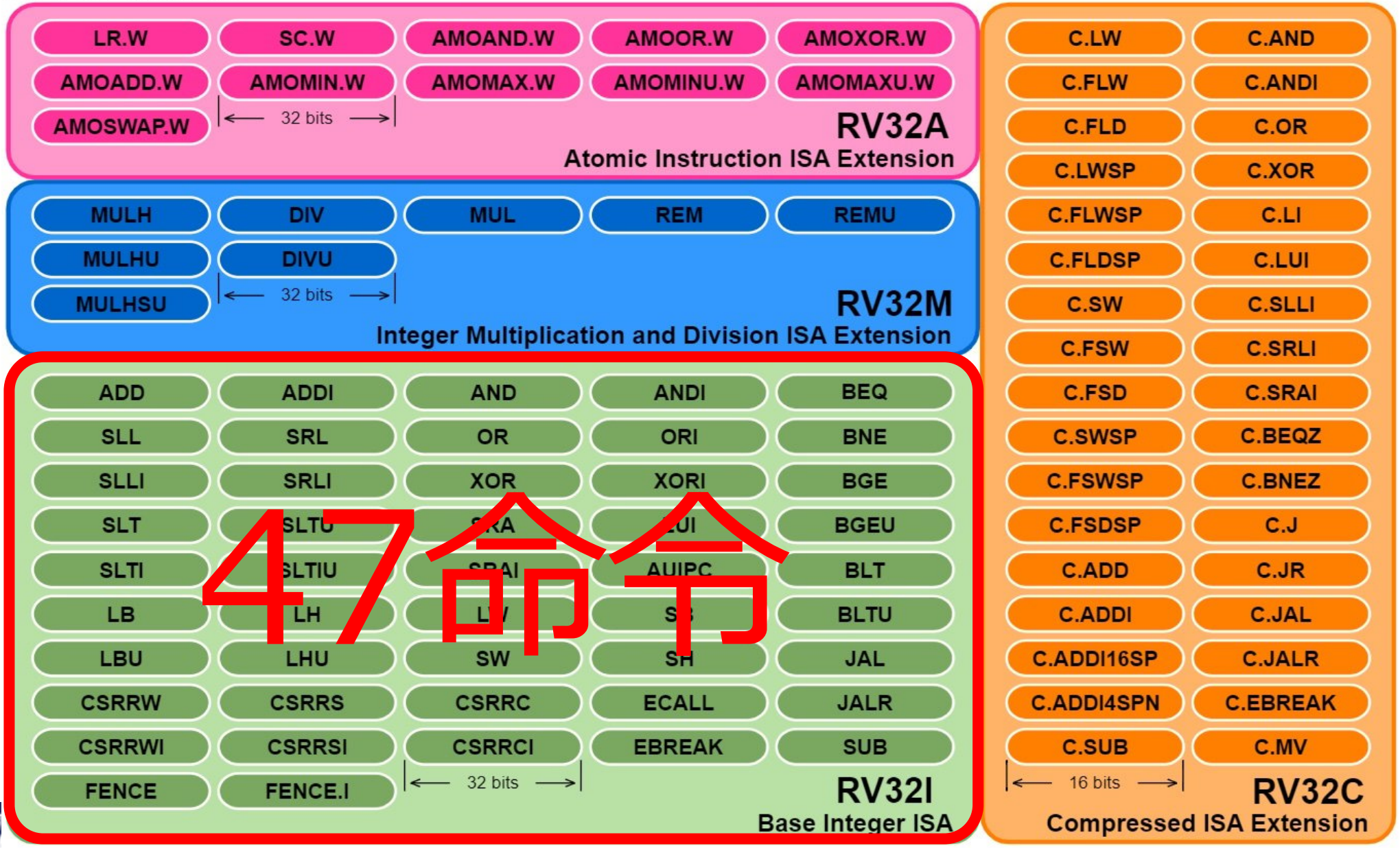
HTTP

SystemVerilog

RV32IMAC

https://en.wikipedia.org/wiki/RISC-V

Codasip

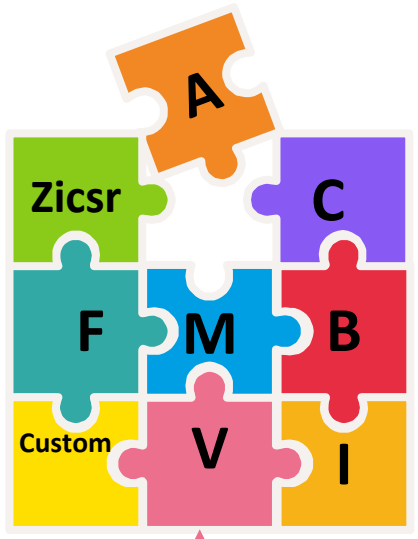


→ RISC-V 標準拡張

互換性を担保したい場合近年、
RVA23S64等のプロファイルが
準備されている

https://en.wikipedia.org/wiki/RISC-V

Extension				
M	Standard Extension for Integer Multiplication and Division	2.0	Ratified	8 (RV32) 13 (RV64)
A	Standard Extension for Atomic Instructions	2.1	Ratified	11 (RV32) 22 (RV64)
F	Standard Extension for Single-Precision Floating-Point	2.2	Ratified	26 (RV32) 30 (RV64)
D	Standard Extension for Double-Precision Floating-Point	2.2	Ratified	26 (RV32) 32 (RV64)
Zicsr	Control and Status Register (CSR) Instructions	2.0	Ratified	6
Zifencei	Instruction-Fetch Fence	2.0	Ratified	1
G	Shorthand for the IMAFDZicsr_Zifencei base and extensions[1]:129	—	—	
Q	Standard Extension for Quad-Precision Floating-Point	2.2	Ratified	28 (RV32) 32 (RV64)
L	Standard Extension for Decimal Floating-Point	0.0	Open	
C	Standard Extension for Compressed Instructions	2.0	Ratified	40
B	Standard Extension for Bit Manipulation	1.0	Ratified	43[28]
J	Standard Extension for Dynamically Translated Languages	0.0	Open	
T	Standard Extension for Transactional Memory	0.0	Open	
P	Standard Extension for Packed-SIMD Instructions	0.9.10	Open	
V	Standard Extension for Vector Operations	1.0	Ratified	187[29]
Zk	Standard Extension for Scalar Cryptography	1.0.1	Ratified	49[30]
H	Standard Extension for Hypervisor	1.0	Ratified	15
S	Standard Extension for Supervisor-level Instructions	1.12	Ratified	4
Zam	Misaligned Atomics	0.1	Open	
Zihintpause	Pause Hint	2.0	Ratified	
Zihintntl	Non-Temporal Locality Hints	0.2	Open	
Zfa	Additional Floating-Point Instructions	0.1	Open	
Zfh	Half-Precision Floating-Point	1.0	Ratified	
Zfhmin	Minimal Half-Precision Floating-Point	1.0	Ratified	
Zfinx	Single-Precision Floating-Point in Integer Register	1.0	Ratified	
Zdinx	Double-Precision Floating-Point in Integer Register	1.0	Ratified	
Zhinx	Half-Precision Floating-Point in Integer Register	1.0	Ratified	
Zhinxmin	Minimal Half-Precision Floating-Point in Integer Register	1.0	Ratified	
Zmmul	Multiplication Subset of the M Extension	1.0	Ratified	
Ztso	Total Store Ordering	1.0	Ratified	



V拡張**417**命令

https://wiki.riscv.org/display/HOME/Recently+Ratified+Extensions

Specification name	Ratification date	New extension(s) or Profile(s)
"Zihintntl" Non-Temporal Locality Hints	May 2023	Zihintntl
RISC-V Code Size Reduction	April 2023	Zca, Zcb, Zcd, Zce, Zcf, Zcmp, Zcmt
RISC-V Profiles	March 2023	RVA20, RVI20, RVA22
"Zicntr" and "Zihpm" Counters	March 2023	Zicntr, Zihpm
RV32E and RV64E Base Integer Instruction Sets	January 2023	RV32E/RV64E
"Ztso" Standard Extension for Total Store Ordering	January 2023	Ztso
RISC-V Wait-on-Reservation-Set (Zawrs) extension	November 2022	Zawrs
Zmmul Extension	June 2022	Zmmul
PMP Enhancements for memory access and execution prevention on Machine mode (Smepmp)	November 2021	Smepmp
RISC-V Base Cache Management Operation ISA Extensions	November 2021	Zicbom, Zicbop, Zicboz
RISC-V Bit-Manipulation ISA-extensions	November 2021	Zba, Zbb, Zbc, Zbs
RISC-V Count Overflow and Mode-Based Filtering Extension	November 2021	Sscofpmf
RISC-V Cryptography Extensions Volume I: Scalar & Entropy Source Instructions	November 2021	Zbkb, Zbkc, Zbkx, Zknd, Zkne, Zknh, Zksed, Zksh, Zkn, Zks, Zkt, Zk, Zkr
RISC-V State Enable Extension	November 2021	Smstateen
RISC-V "stimecmp / vstimecmp" Extension	November 2021	Sstc
RISC-V Vector Extension	November 2021	Zve32x, Zve32f, Zve64x, Zve64f, Zve64d, Zve, Zvl32b, Zvl64b, Zvl128b, Zvl256b, Zvl512b, Zvl1024b, Zvl, Zv
The RISC-V Instruction Set Manual Volume II: Privileged Architecture	November 2021	Sm1p12, Ss1p12, Sv57, Hypervisor, Svinval, Svnapot, Svpbmt
"Zfh" and "Zfhmin" Standard Extensions for Half-Precision Floating-Point	November 2021	Zfh, Zfhmin
"Zfinx", "Zdinx", "Zhinx", "Zhinxmin": Standard Extensions for Floating-Point in Integer Registers	November 2021	Zfinx, Zdinx, Zhinx, Zhinxmin
"Zihintpause" Pause Hint	February 2021	Zihintpause

Codasip

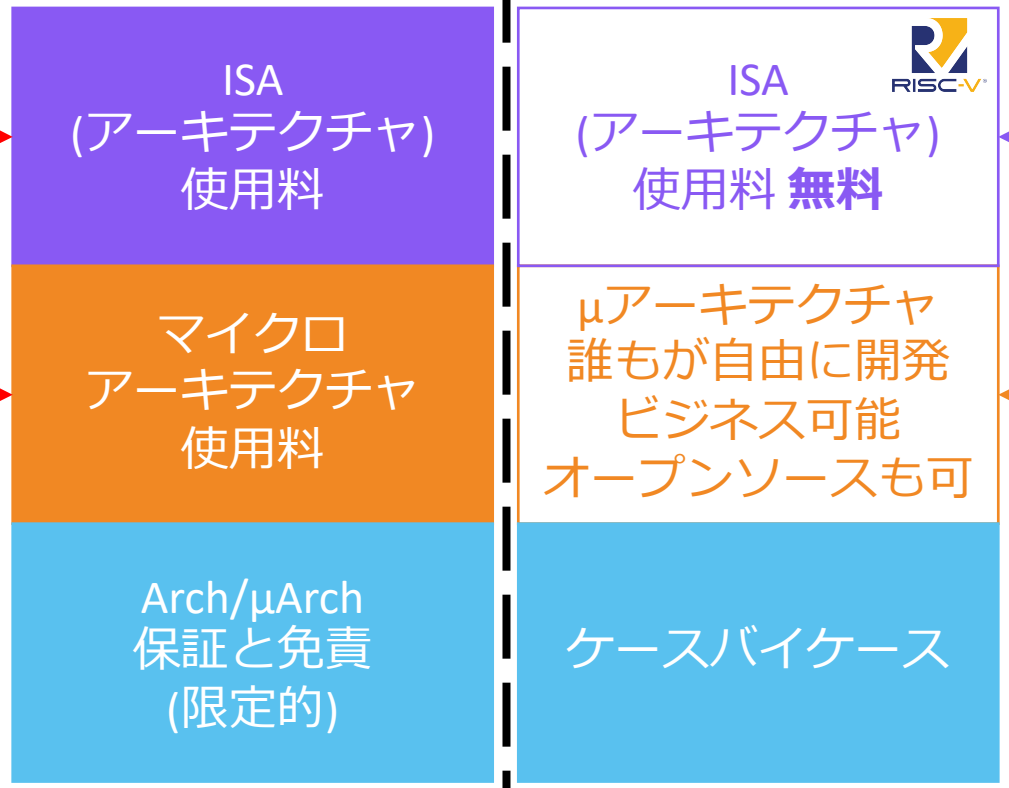
RISC-Vの大きな価値は
- 特定ベンダにロックインされない
- 多くの自由がある
- 競争原理が働き、選択肢が増え、コストが下がる

→ プロセッサ ライセンスモデルの違い

Codasip

特定企業が提供する
ISAが主で
オープン化が
進んでいなかった
固定ISA

特定企業が提供する
ほぼ固定のuArch
を使用する



RISC-V ISAによって自由化

RV32E/32I/64I/128I等
基本定義はあるが、ほぼ自由
様々なμアーキテクチャ実装が
行われ、競争原理が働く

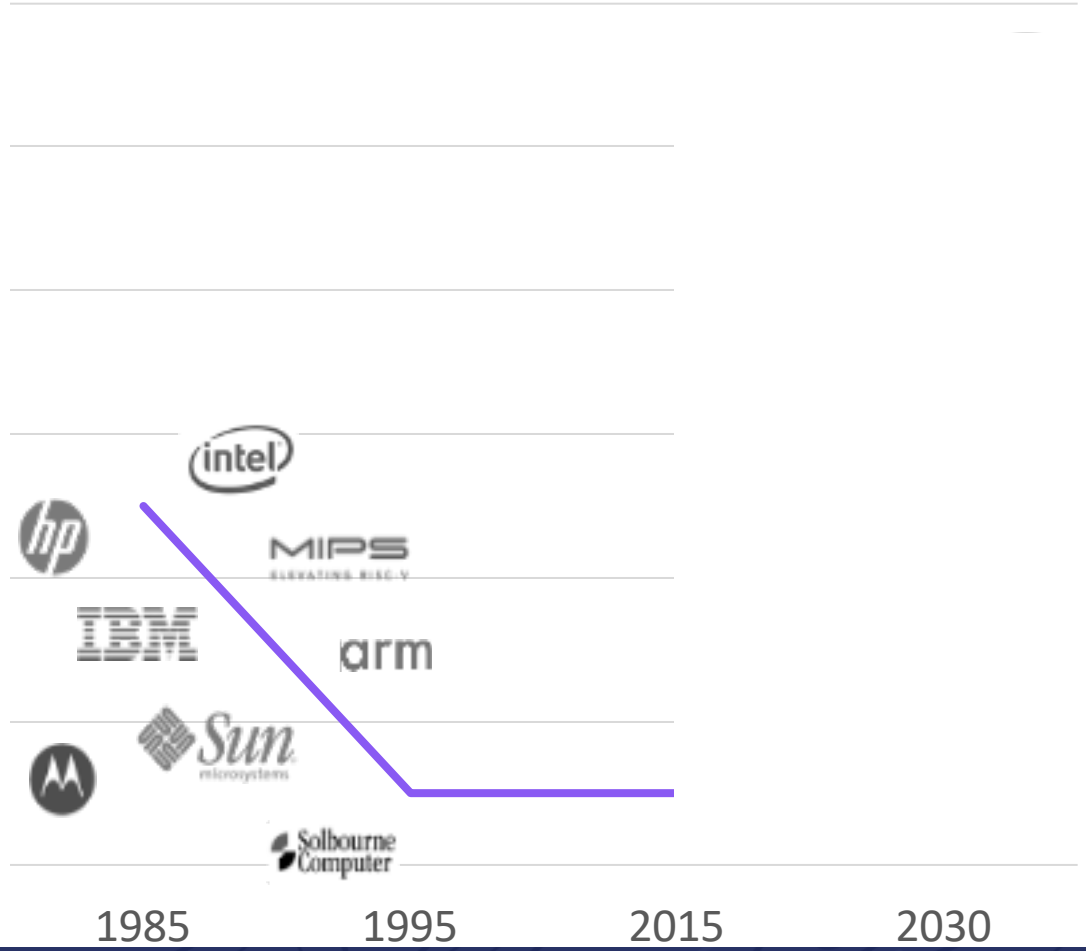
商用RISC-Vも
オープンソースのRISC-Vもある

→ IPライセンスモデルの違い

Codasip

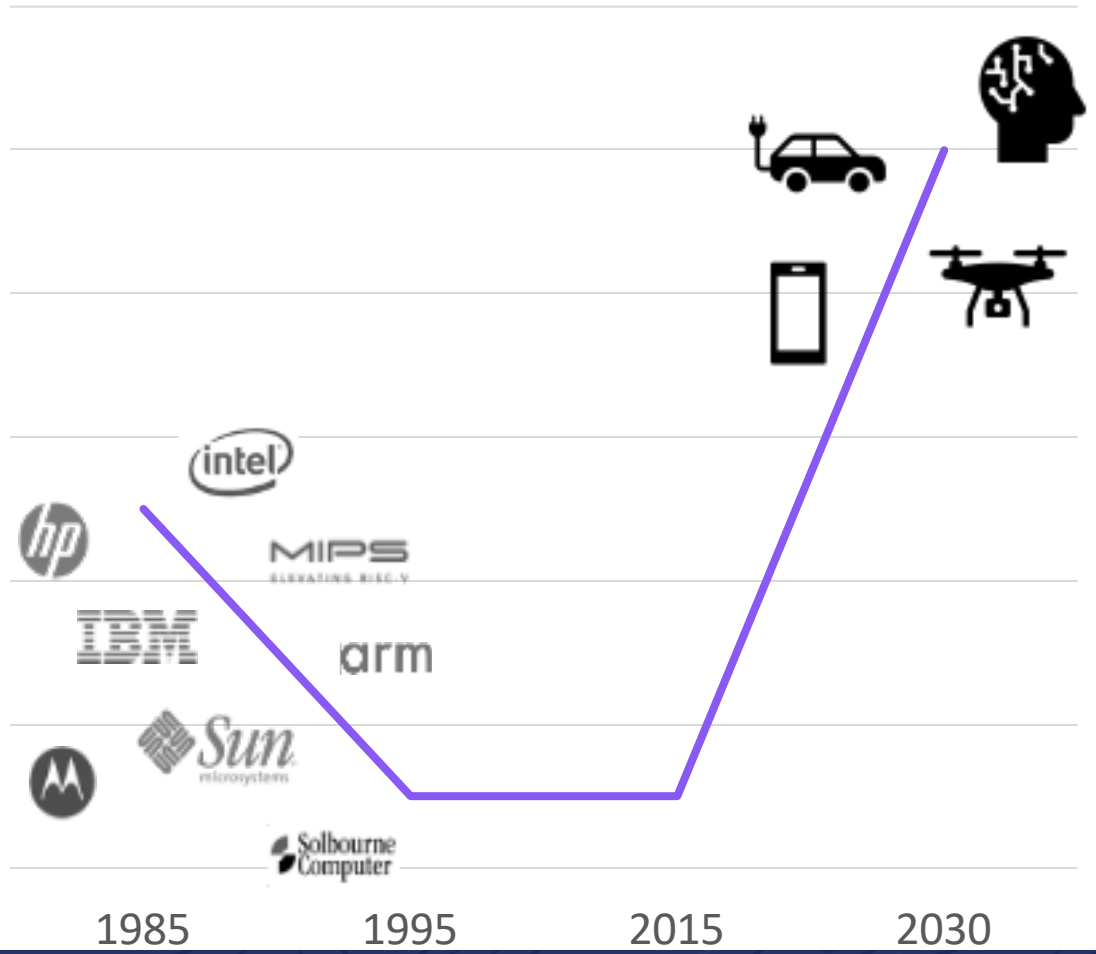
ISA (アーキテクチャ) 使用料	ISA (アーキテクチャ) 使用料 **無料**	ISA (アーキテクチャ) 使用料 **無料**
マイクロ アーキテクチャ 使用料	マイクロ アーキテクチャ 使用料	マイクロ アーキテクチャ 使用料 **無料**
Arch/uArch 保証と免責 (限定的)	uArch 保証と免責 (限定的)	**無保証・無補償**
古典的 商用IPライセンス	RISC-V 商用IPライセンス	RISC-V オープンソース ライセンス

→ プロセッサの歩み



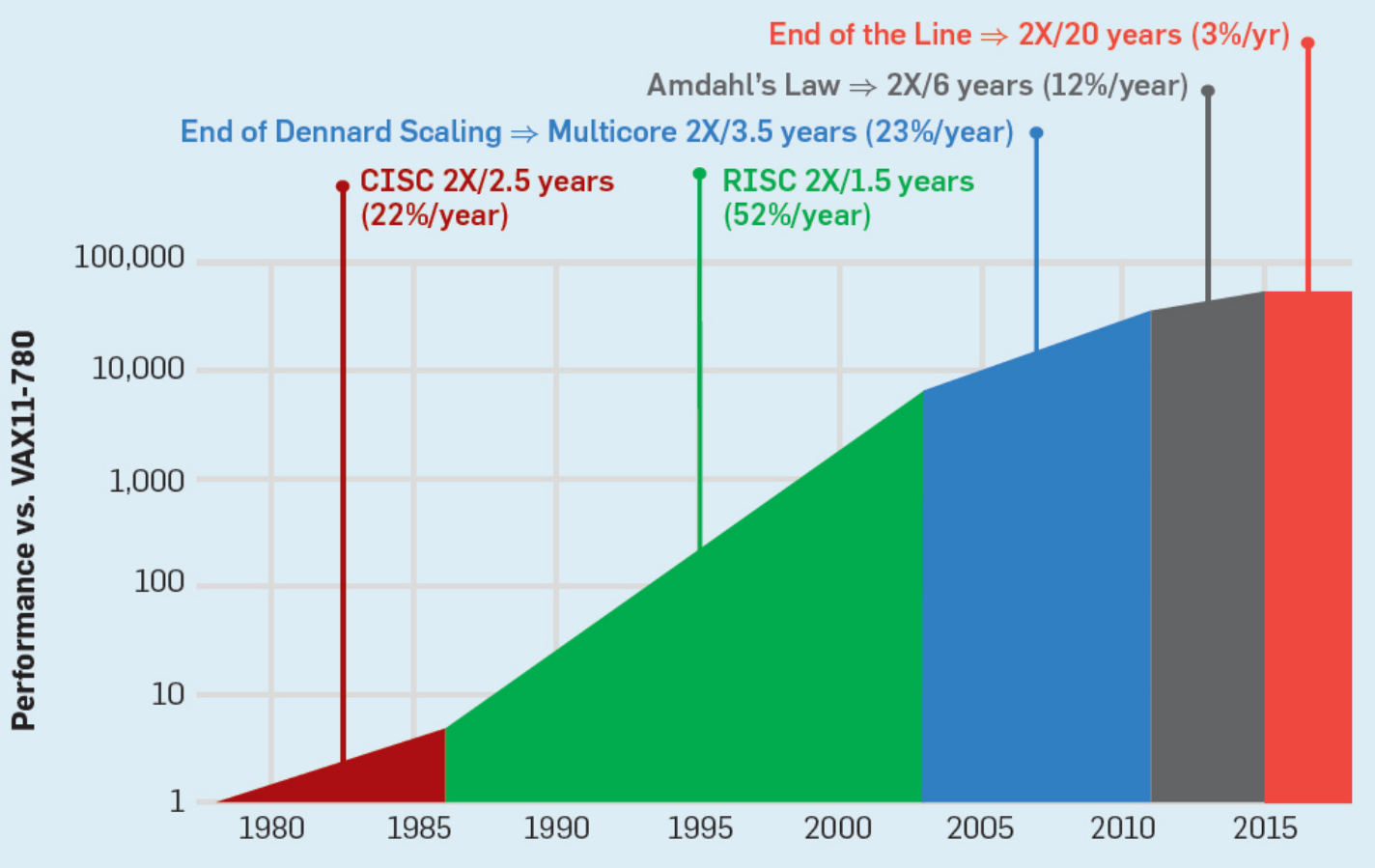
- 従来から差別化したプロセッサ・アーキテクチャが求められる傾向はある
- 多くのプロセッサ・ベンダが独自のアーキテクチャを維持し差別化を謀っていた過去にも、このような傾向はあった
- しかしプロセッサ開発にはハードウェアのみならずコンパイラ等の専門知識が必要であり、優秀な人材確保など、コストがかかり過ぎたため淘汰が起きた
- 約20年、この市場は活性化されなかった

→ 近年の多様なデマンド



- スマートフォン、自動運転、EV、AI、ドローン等、多様なPPAデマンド
- パフォーマンス
  - 汎用プロセッサ＋ソフトウェア処理では、コンピューティング能力の限界があり、ドメイン特化型のHWアクセラレータの需要が高まっている
- 低消費電力化
  - バッテリ駆動製品やAIによる大量の電力消費
- 半導体のコストダウン
  - 微細化に伴うコスト上昇を抑えたい

→ 汎用プロセッサの停滞
= Custom Computeのデマンド



出典: Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson,
"A Domain-Specific Architecture for Deep Neural Networks"
https://cacm.acm.org/magazines/2018/9/230571-a-domain-specific-architecture-for-deep-neural-networks/fulltext

成長の余地: ムーアの法則が通用しなくなったときにコスト・パフォーマンスを大きく向上させる唯一の道は、**特定のドメイン向け命令を追加**することである。例えばディープ・ラーニング、拡張現実、組み合せ最適化、グラフィックスなどのドメインが考えられる。

出典: RISC-V原典（日本語版）P9

*In our current research, we are **especially interested** in the **move towards specialized and heterogeneous accelerators**, driven by the power constraints imposed by the **end of conventional transistor scaling**. We wanted a **highly flexible and extensible base ISA** around which to build our research effort.*

出典: RISC-V Spec Volume I: Unprivileged ISA 20191213, Section 28.1
https://github.com/riscv/riscv-isa-manual/releases/download/Ratified-IMAFDQC/riscv-spec-20191213.pdf

31 26	25 15	14 12	11 7	6 0	
funct6	*custom*	funct3	*custom*	opcode	Recommended Purpose
6	11	3	5	7	
100011	*custom*	0	*custom*	SYSTEM	Unprivileged or User-Level
110011	*custom*	0	*custom*	SYSTEM	Unprivileged or User-Level
100111	*custom*	0	*custom*	SYSTEM	Supervisor-Level
110111	*custom*	0	*custom*	SYSTEM	Supervisor-Level
101011	*custom*	0	*custom*	SYSTEM	Hypervisor-Level
111011	*custom*	0	*custom*	SYSTEM	Hypervisor-Level
101111	*custom*	0	*custom*	SYSTEM	Machine-Level
111111	*custom*	0	*custom*	SYSTEM	Machine-Level

Figure 3.30: SYSTEM instruction encodings designated for custom use.

出典: RISC-V Spec Vol II: Privileged Architecture 20211203, Section 3.3.4
https://github.com/riscv/riscv-isa-manual/releases/download/Priv-v1.12/riscv-privileged-20211203.pdf

- **RV32I** – 最も基本的なRISC-Vの実装
- **RV32IMAC**
  – 整数 + 乗算 + アトミック命令 + 圧縮命令
- **RV32IMAC_X[ext]**
  – MAC + **非標準ユーザー拡張**

考慮されている「非標準ユーザー拡張」

Codasip

RISC-Vの高い自由度は
多様化するデマンドにフィットする

→ RISC-Vには大きく3つの道があります

Codasip

商用IP

オープンソースIP

“Homebrew”

→ それぞれの特徴

Codasip

商用IP

- アーキテクチャ: 固定
- マイクロアーキテクチャ: 固定
- PPA: 固定
- カスタマイズ: 不可
  (一部命令追加APIがある場合有)

- ライセンス費: 有償
- ロイヤリティ費: 有償
- サポート費: 有償
- 保証・補償: **有**

オープンソースIP

- アーキテクチャ: 固定
- マイクロアーキテクチャ: 固定
- PPA: 固定
- カスタマイズ: 各OSSライセンスによる

- ライセンス費: **不要**
- ロイヤリティ費: **不要**
- サポート費: **不要**
- 保証・補償: 無

“Homebrew”

- アーキテクチャ: **自由**
- マイクロアーキテクチャ: **自由**
  PPA: **自在**
- カスタマイズ: **制限無し**

- ライセンス費: **不要**
- ロイヤリティ費: **不要**
- サポート費: **不要**
- 保証・補償: 無

→ RISC-Vの多くのメリットを享受できる”Homebrew”

Codasip

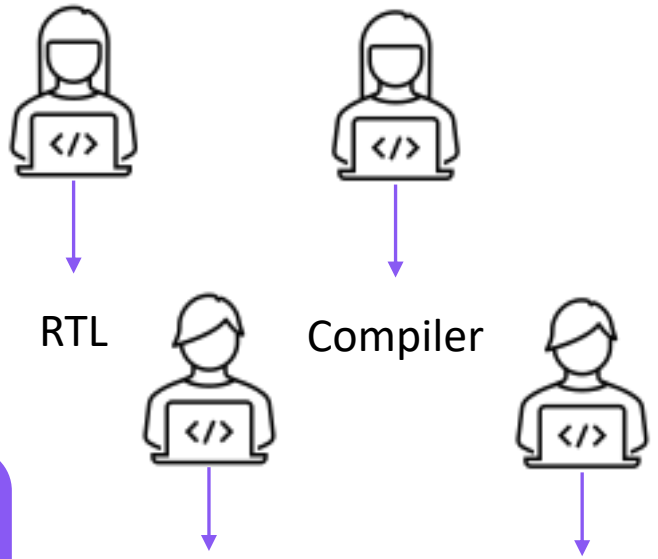
しかし…

“Homebrew”

複数のエキスパートが必要

「コンパイラ/SDKはオープンソースを使えばタダ!」

「RISC-Vの自由度を活用したい場合はOSSコンパイラのカスタマイズが必要です…」



- アーキテクチャ: **自由**
- マイクロアーキテクチャ: **自由**
  PPA: **自在**
- カスタマイズ: **制限無し**

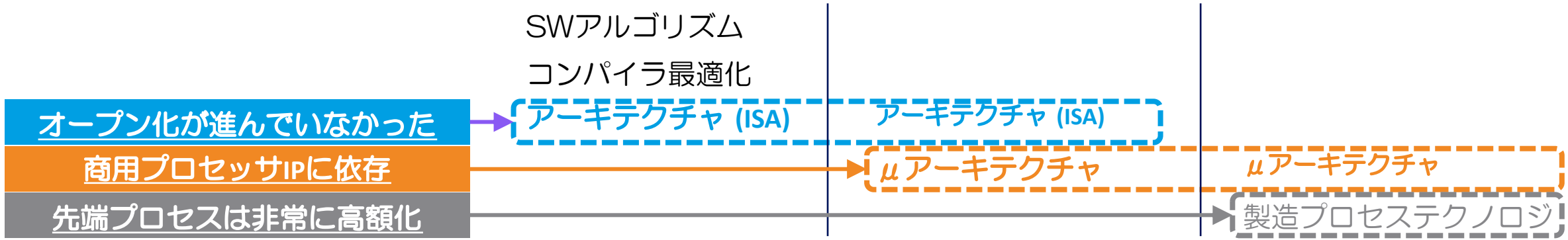
- ライセンス費: **不要**
- ロイヤリティ費: **不要**
- サポート費: **不要**
- 保証・補償: 無

→ プロセッサパフォーマンスの鉄則 (Iron law of processor performance)

Codasip

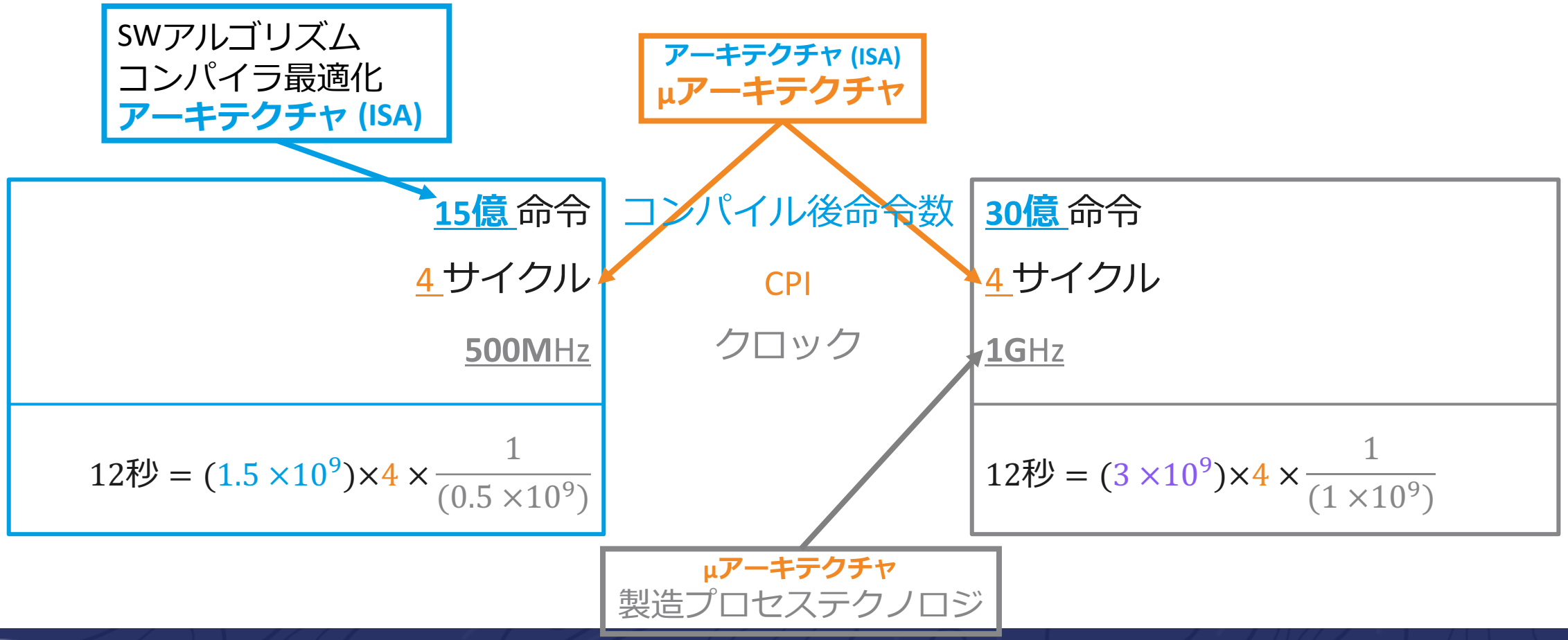
$$\text{パフォーマンス} = \frac{\text{時間}}{\text{プログラム}} = \frac{\text{命令数}}{\text{プログラム}} \times \frac{\text{ClockCycles}}{\text{命令数}} \times \frac{\text{時間}}{\text{ClockCycles}}$$

(プログラム->命令数) (CPI) (Cycle Time)



出典: A Characterization of Processor Performance in the VAX-11/780, Joel S. Emer, Douglas W. Clark, 1984, IEEE

→ Iron law を用いた考察



→ RISC-Vならできる
アーキテクチャ(IA)とμアーキテクチャ(CA)
のカスタマイズ（最適化）

$$\text{パフォーマンス} = \frac{\text{時間}}{\text{プログラム}} = \frac{\text{命令数}}{\text{プログラム}} \times \frac{\text{ClockCycles}}{\text{命令数}} \times \frac{\text{時間}}{\text{ClockCycles}}$$

(プログラム->命令数)　(CPI)　(Cycle Time)

出典: A Characterization of Processor Performance in the VAX-11/780, Joel S. Emer, Douglas W. Clark, 1984, IEEE

→ デザイン/カスタマイズできること

Codasip

アーキテクチャ(命令精度)レベル

- スカラー命令
- SIMD/ベクトル命令
- 単精度/倍精度浮動小数点
- 複数の入出力オペランドを持つ命令
- DSP命令
- ゼロ・オーバーヘッド・ループ
- プリ/ポストインクリメント付きロード/ストア
- 複数のレジスタファイル
- 等々

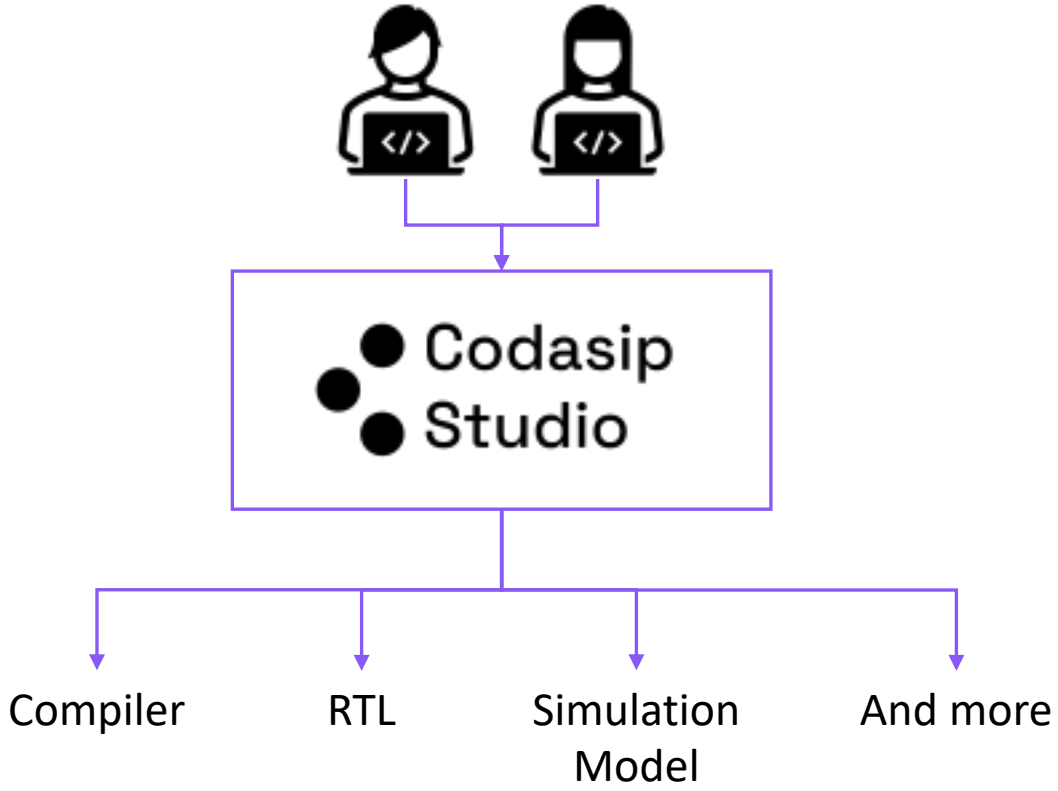
マイクロアーキテクチャ レベル

- パイプラインステージ数の定義
  - ステージの難読化
- メモリ・サブシステムへの接続の定義
  - プロトコル
  - ビット幅
- データ／制御／構造的ハザード処理
- リソース共有
- MMU物理メモリ属性／保護
- 等々

→ カスタムRISC-V “Homebrew”を極める

Codasip

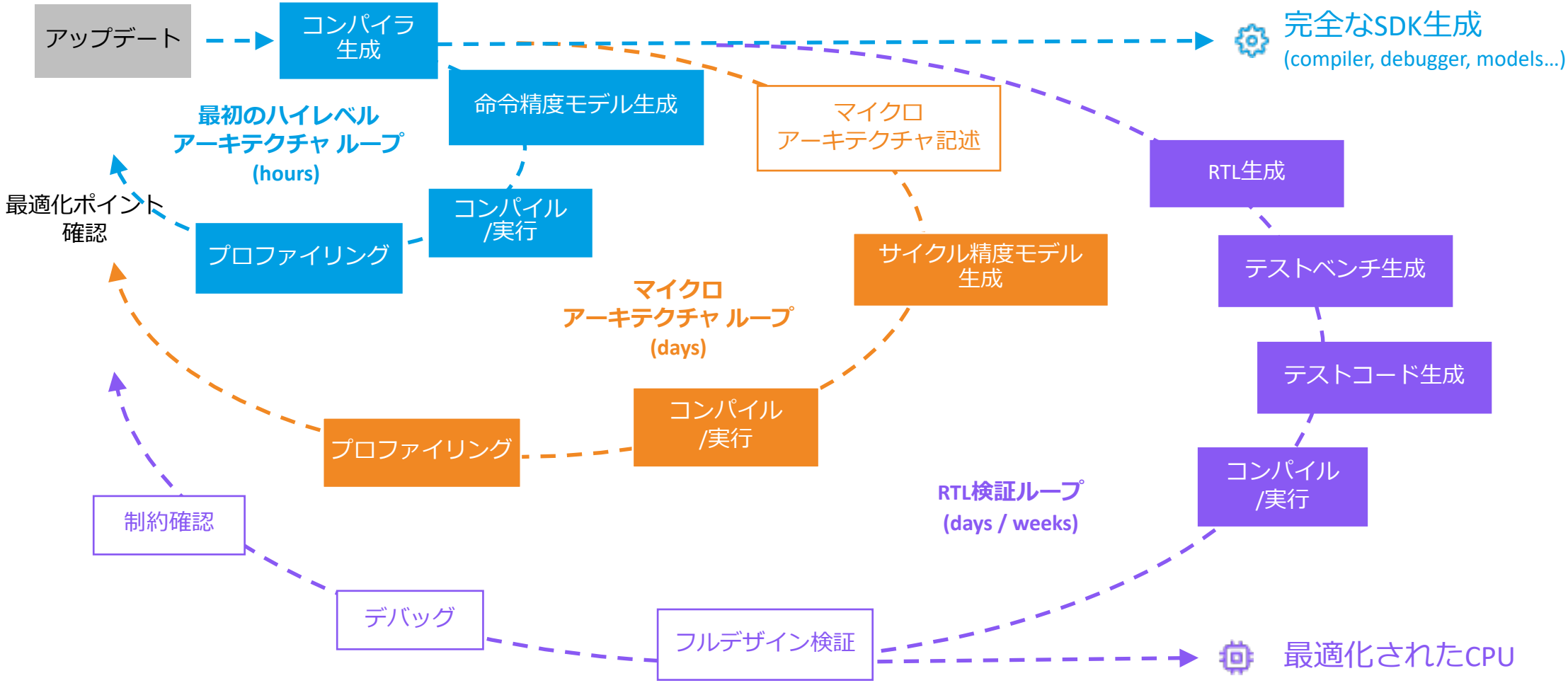
1. ツールを活用する



2. ゼロ・スタートでなく テンプレートを活用

Codasip uRISC	Minimal RISC architecture
Codasip uVLIW	Minimal architecture implementing very long instruction words
Codasip uRISC-V	Minimal implementation of the RISC-V architecture

→ 最短で最適な結果を得る開発フロー
自動化された反復的デザインプロセス



→ SDKとHDKを自動生成

Codasip

Software Development Kit (SDK)

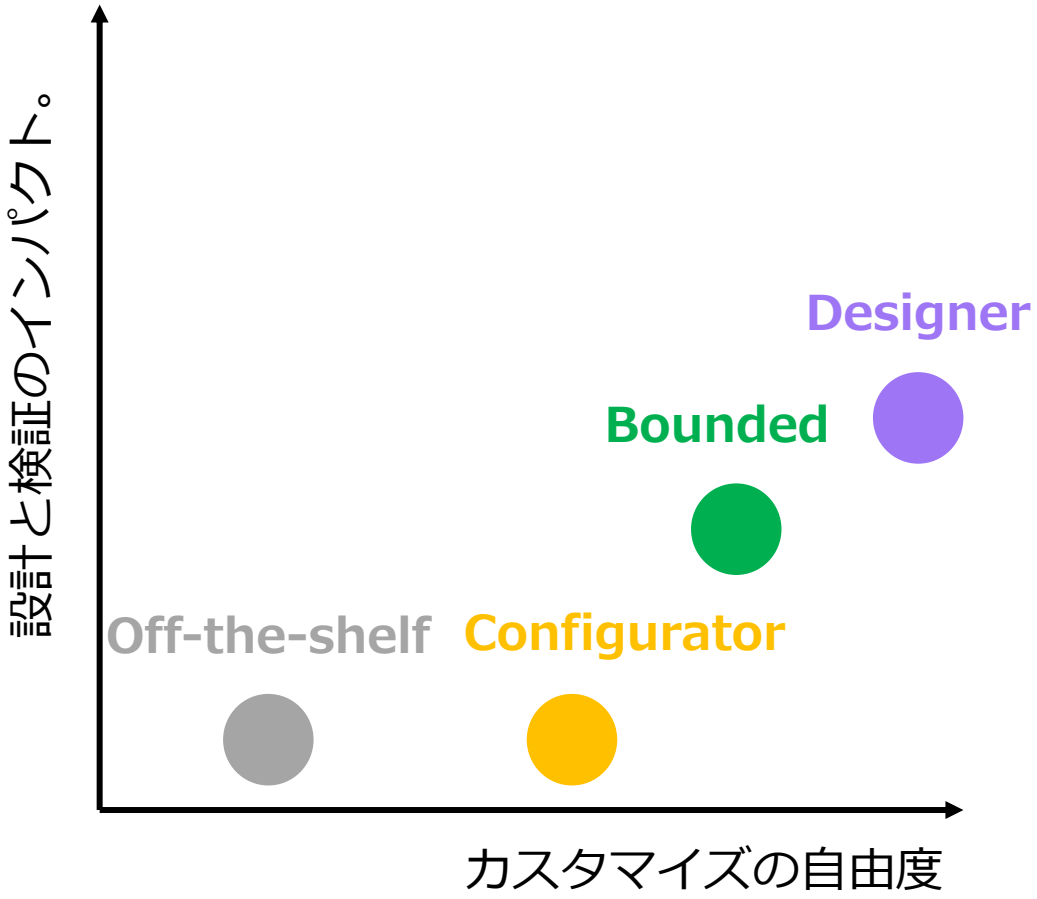
- C/C++ LLVM **コンパイラ**
  (コダシップ改良版)
- C/C++ ライブラリ (newlib)
- アセンブラ、逆アセンブラ、リンカー
- **命令精度(IA)シミュレータ**
- **デバッガとプロファイラー**
- ドキュメントとISAの可視化
- 検証時に使用するランダムプログラム

Hardware Development Kit (HDK)

- **RTL (Verilog/VHDL/SystemVerilog)**
  - 高い可読性
  - CodALソースへのリンク
- **サイクル精度(CA)シミュレータ**
- SystemVerilog UVM検証環境
- 統合テストベンチ
- 標準EDAツールのサンプルスクリプト
- SystemCコシミュレーション モデル
- デバッグツール (OpenOCD, LLDB)

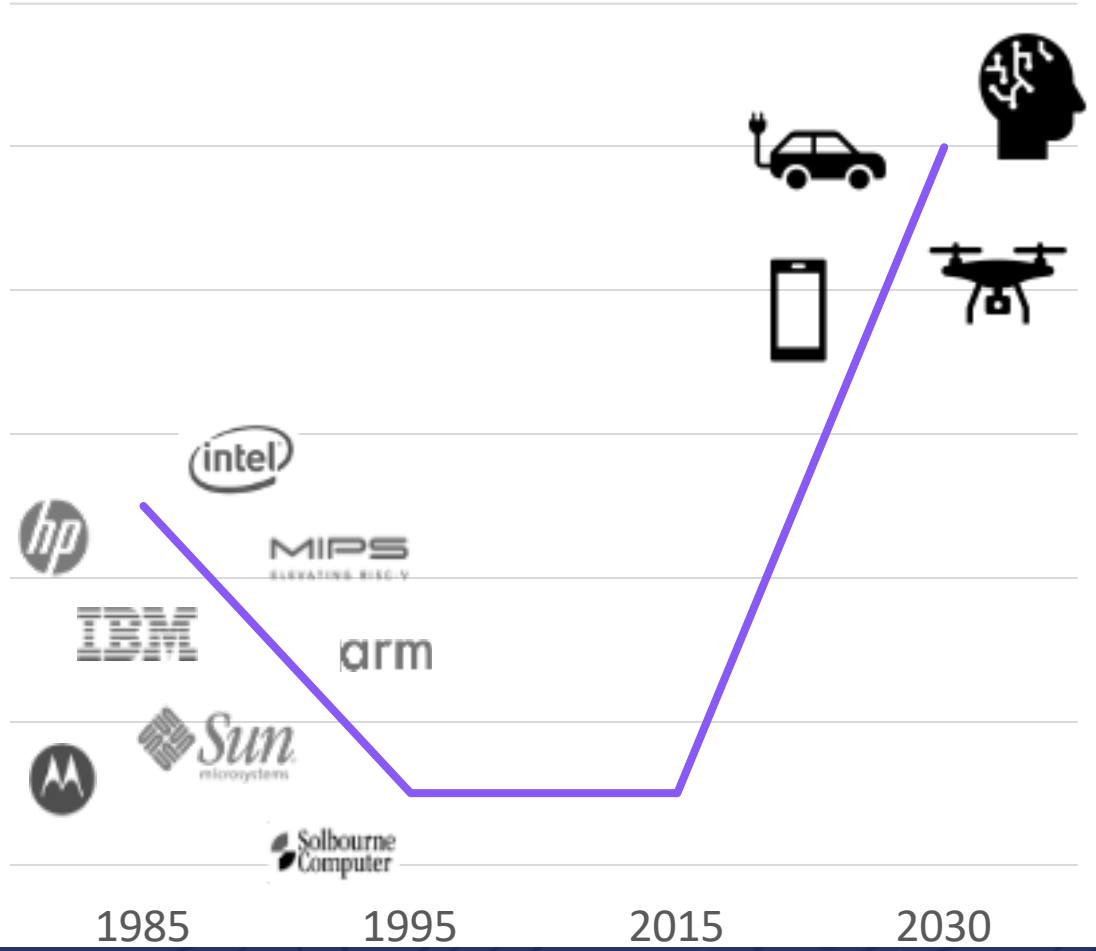
→ コダシップが提供する4つの選択肢

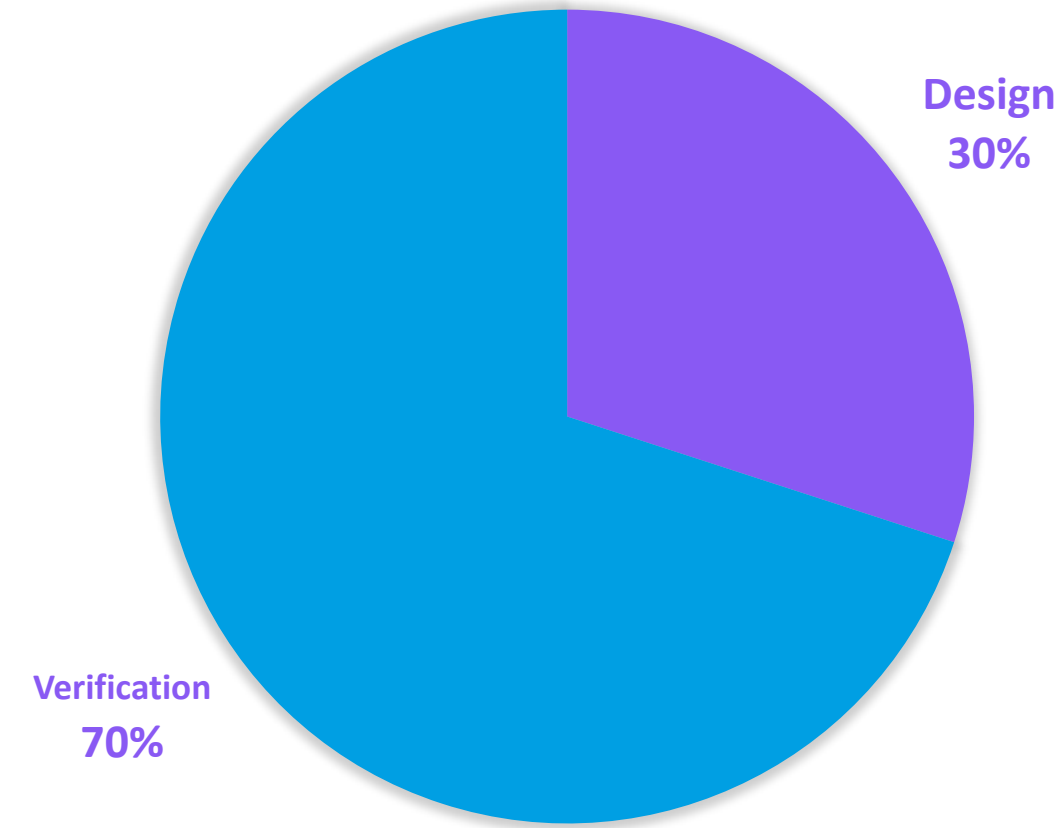
- **既製商用IP (Off-the-shelf)**
  - お手軽、検証済
- **既製商用IP ＋ Configuration**
  - お手軽、検証済、PPAの調整幅有
- **Bounded Custom (CBI)**
  - カスタマイズ部分を限定
  - ユーザは限定部分に注力
  - カスタマイズ部分以外は検証済
  - 検証労力小
- **フルカスタム (Designer)**
  - 全面的にカスタマイズ可能
  - 検証労力大



→ プロセッサ開発の大きなボトルネック = 検証

Codasip





→ カスタムRISC-V “Homebrew”を極める

Codasip

3. CBI (Bounded Customization) を活用する

- カスタマイズ可能な商用品質IPをベースに、ガイドラインに沿ったカスタマイズ
  - プロセッサ・コアの構造を侵食しないガイドライン
    - 高品質を維持しながら検証時間を大幅に削減できる
  - シングルサイクル、マルチサイクル対応
  - 僅かな面積のトレードオフでアプリケーションの性能を大幅に向上

Arithmetic	Averaging Sequences of joint additions
Multi-cycle	CRC Square root
Filter (DSP and AI)	FIR Median
Trigonometry (motor control)	CORDIC
Error correction	LDPC
Decoding	Viterbi
Cryptography	SHA512

Codasip

↘

RISC-Vの自由度をサポートするカスタマイズ可能な高品質ベースライン

RISC-V IP

→ Codasip RISC-V プロセッサ ロードマップ

Codasip

アプリケーション

組込み
リアルタイム

1200 High End RISC-V CPUs

1100 High End RISC-V CPUs

900 Family RISC-V CPUs

Series 7 RISC-V CPUs

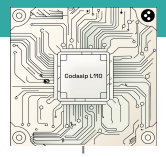
700 Family RISC-V CPUs

300 Family RISC-V CPUs

Series 3 RISC-V CPUs

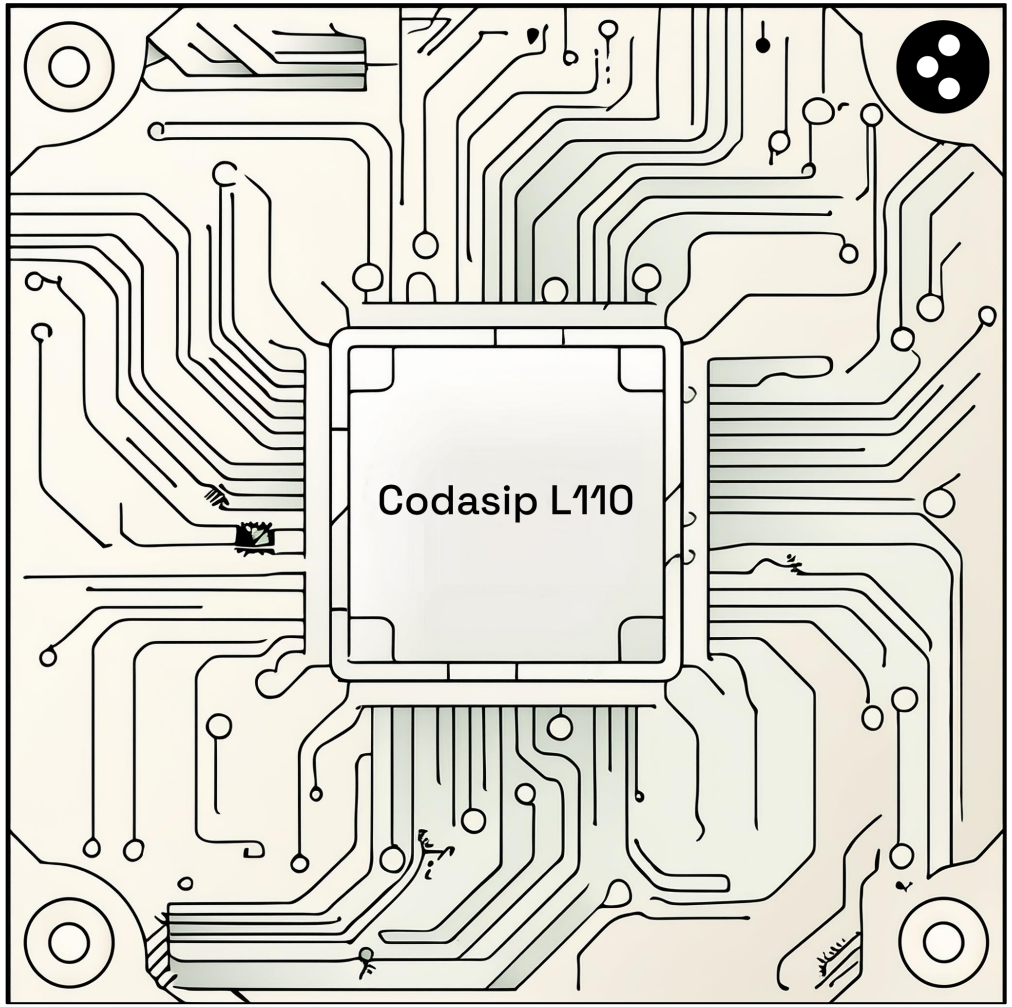
Series 1 RISC-V CPUs

100 Family Small
RISC-V CPUs



→ 低消費電力 小型RISC-VベースラインIP : L110

- ステートマシンの置き換え、
  センサーコントローラ、IoTエッジ等に最適
- 広範なコンフィギュレーションが可能で、
  さまざまな面積/性能のトレードオフ・レベルを実現

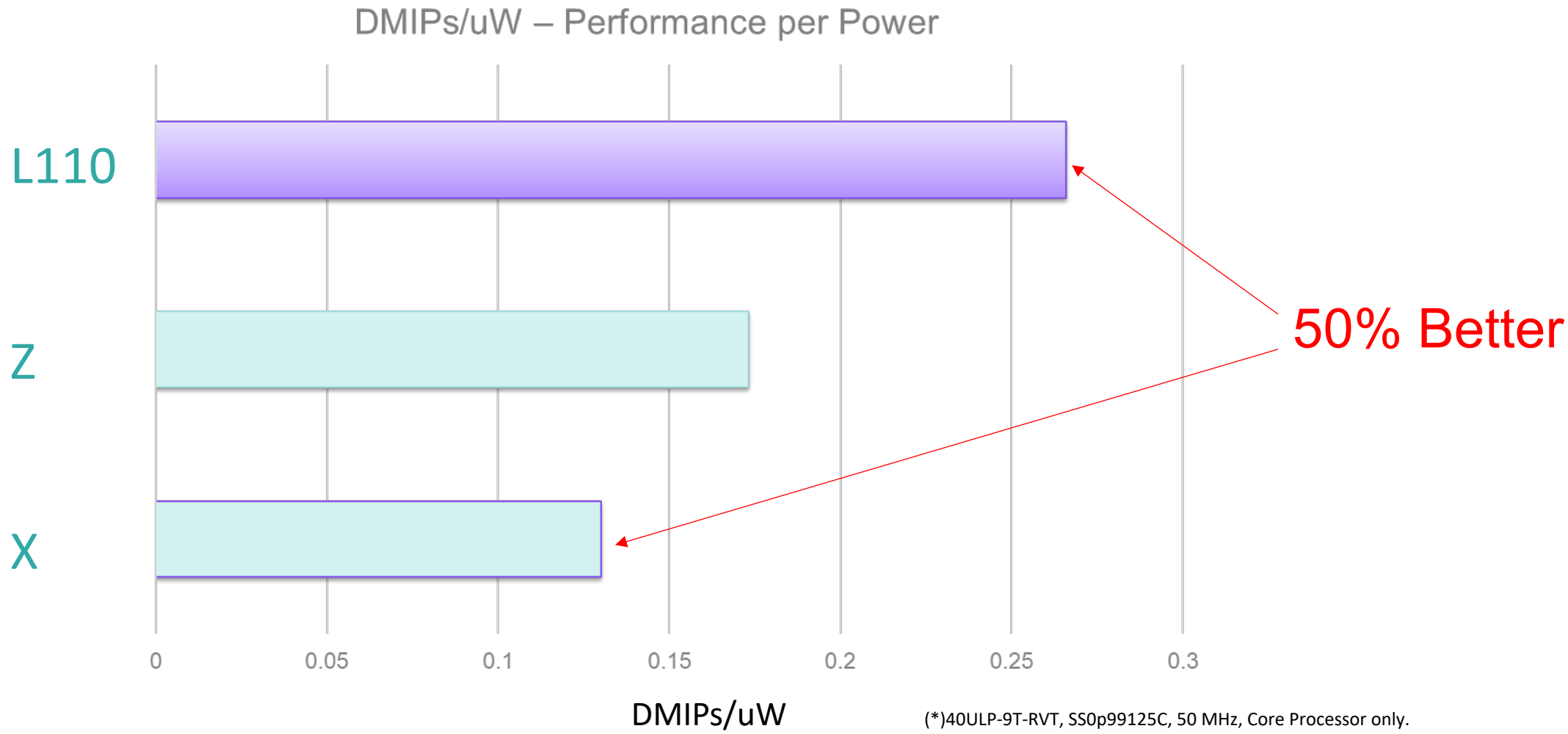


同クラスの競合コアと比較して
ワットあたりのパフォーマンスが**50%向上**
コードサイズは**20%小さい**

→ クラス最高レベルの電力効率

Codasip

競合の同等コアより50%高い電力効率 (カスタマイズ無)

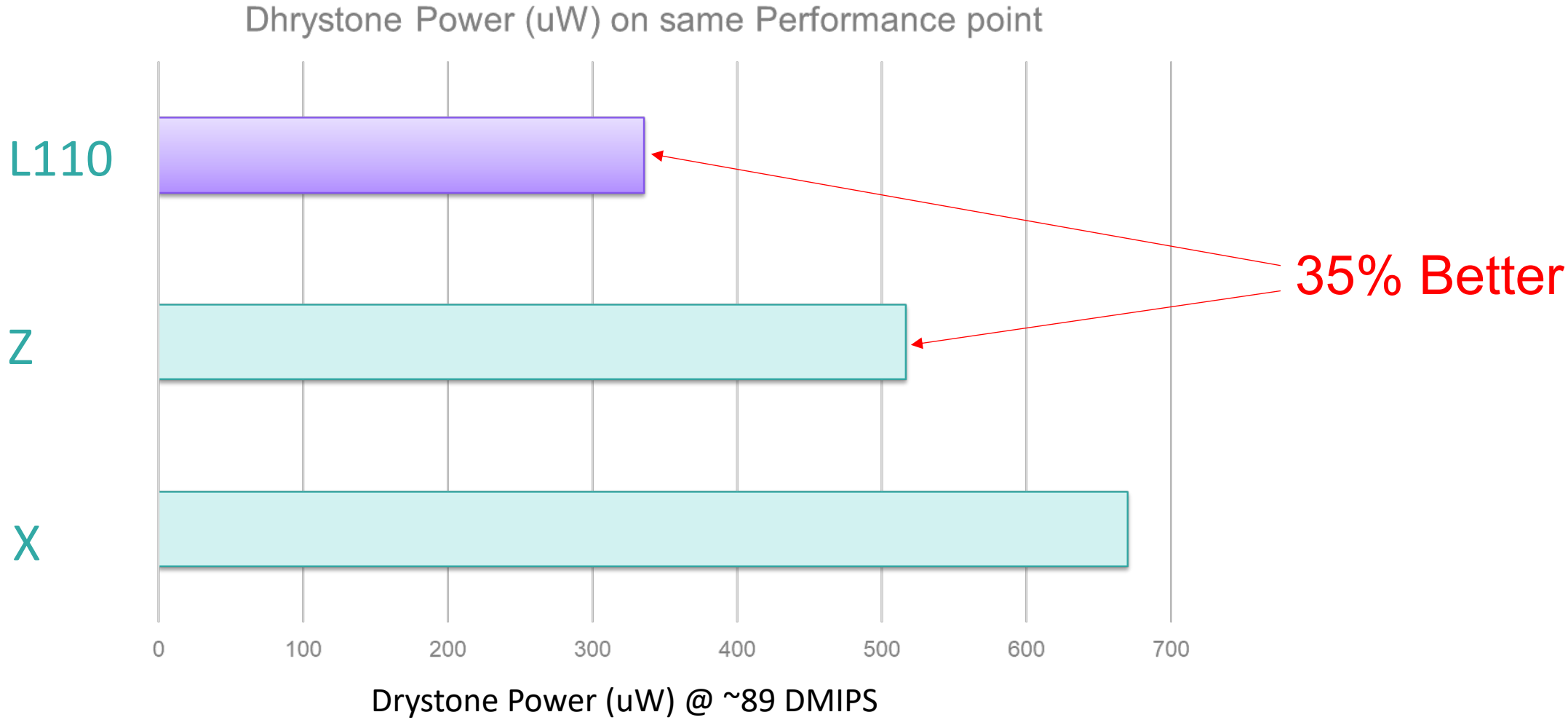


(*)40ULP-9T-RVT, SS0p99125C, 50 MHz, Core Processor only.
Data obtained from ARM M0 and M0+ datasheets

→ クラス最高レベルの電力削減

Codasip

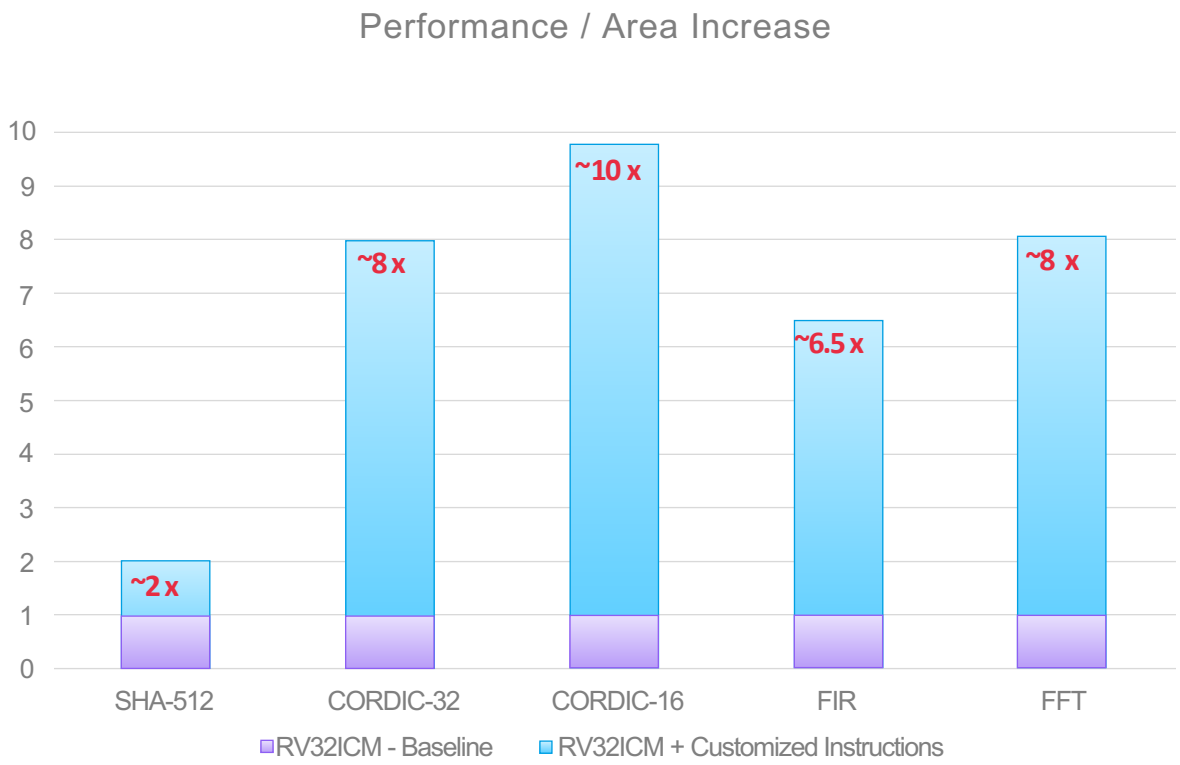
Drystoneの同等スコア値で消費電力が35%良い (カスタマイズ無)

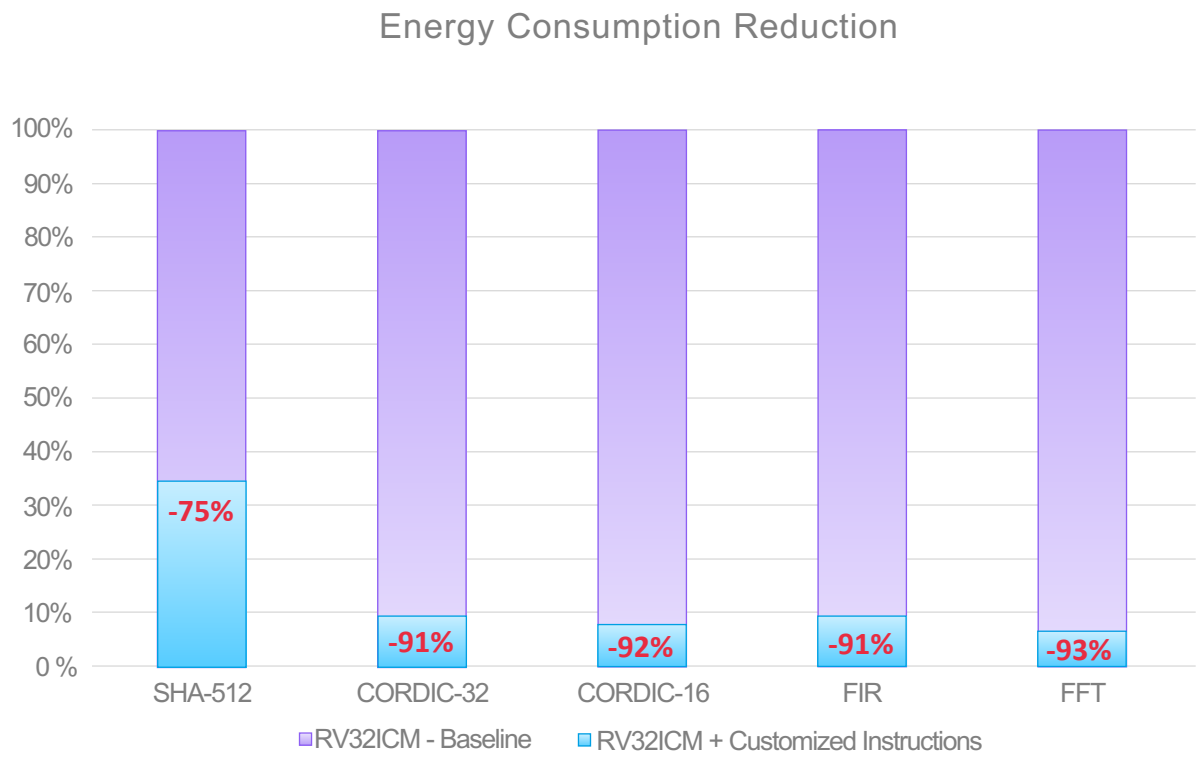


(*)40ULP-9T-RVT, SS0p99125C, 50 MHz, Core Processor only.
Data obtained from ARM M0 and M0+ datasheets

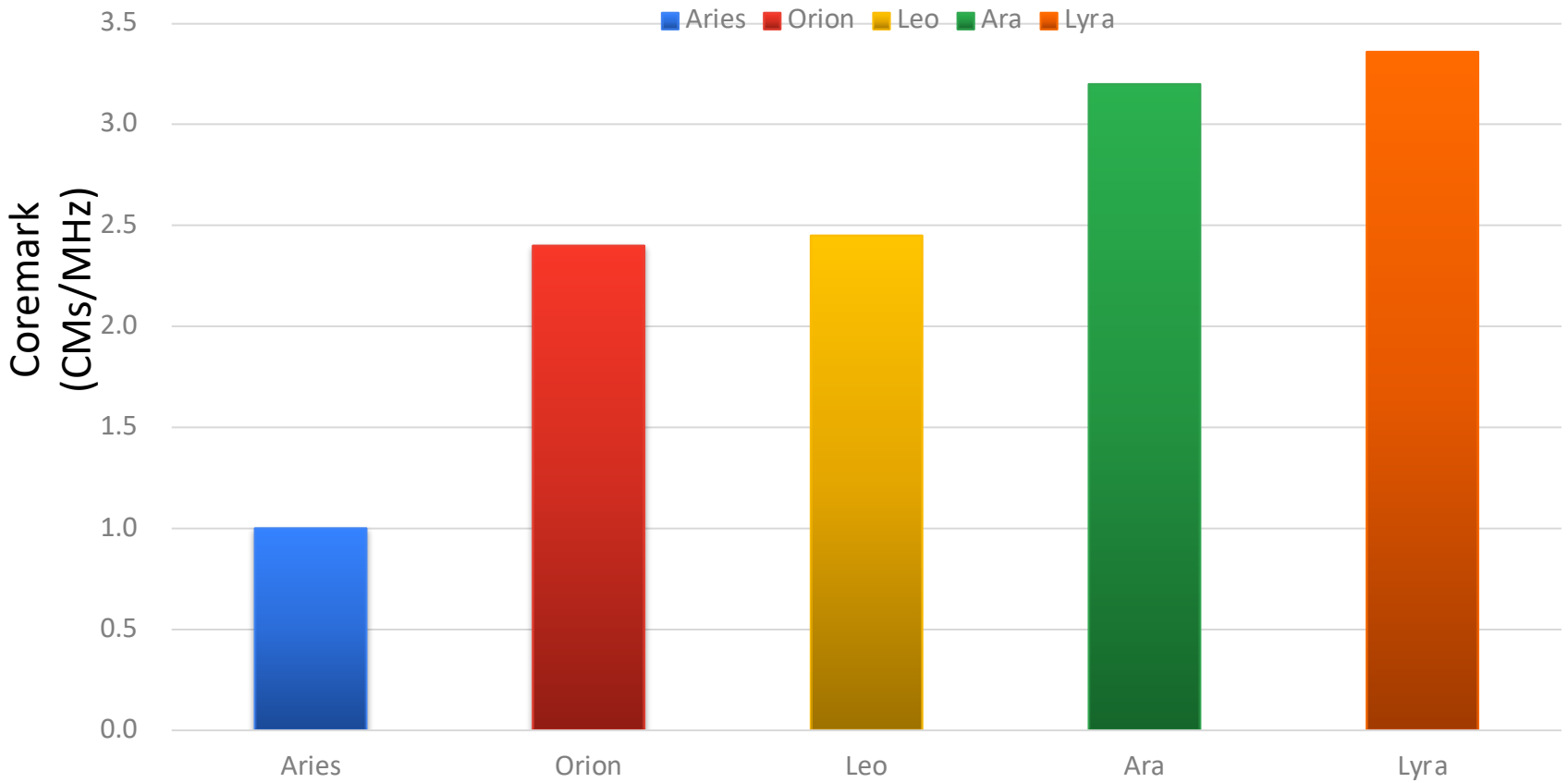
→ CBI (Bounded Customization)の効果

Codasip





→ PPAデマンドに合わせて
高度にコンフィギュラブルできるL110



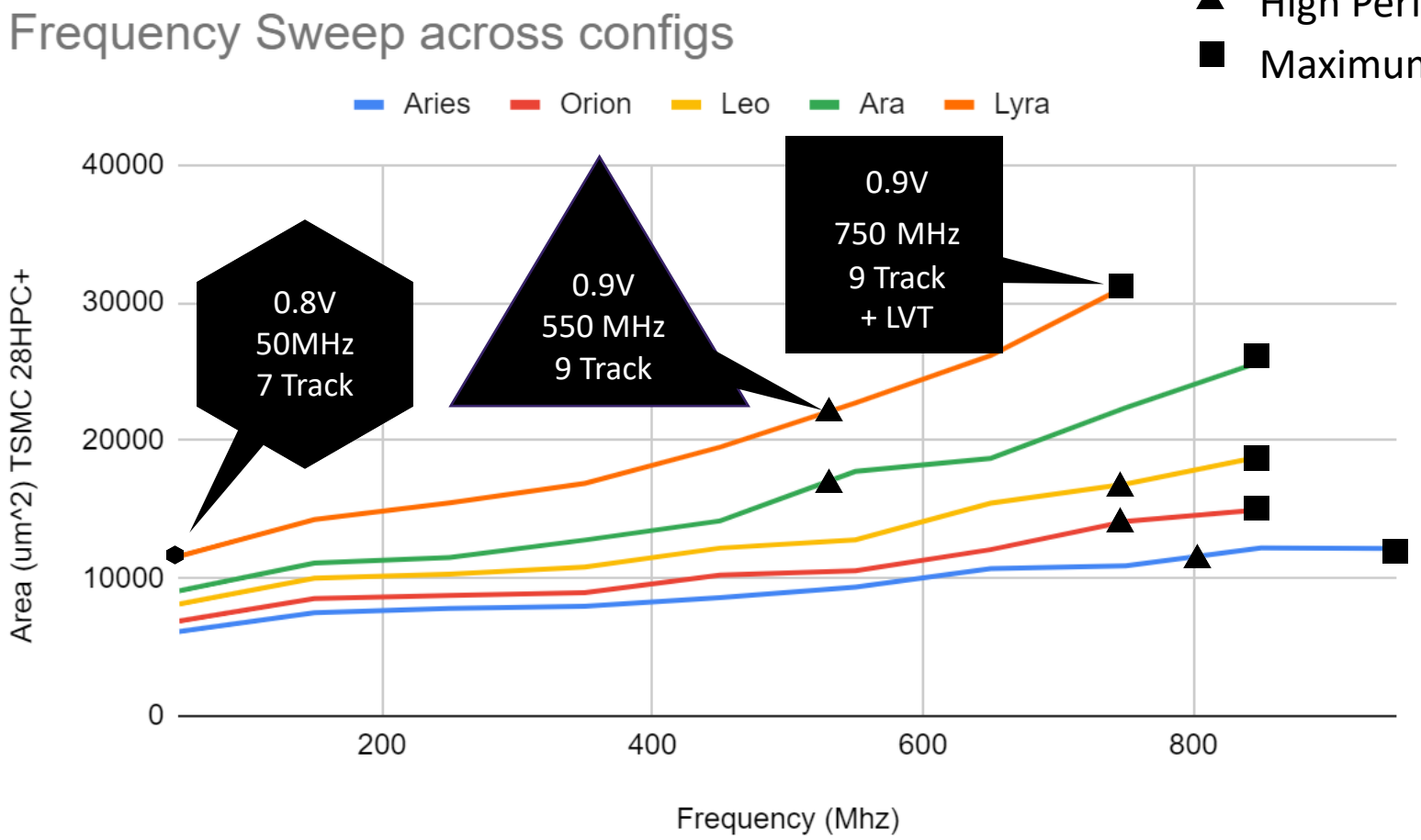
What is your performance requirement?

→ 最適ポイントで製品化

TSMC 28HPC+

Codasip

- Low Area/Power sweet spot
- High Performance sweet spot
- Maximum Frequency



Codasip

↘

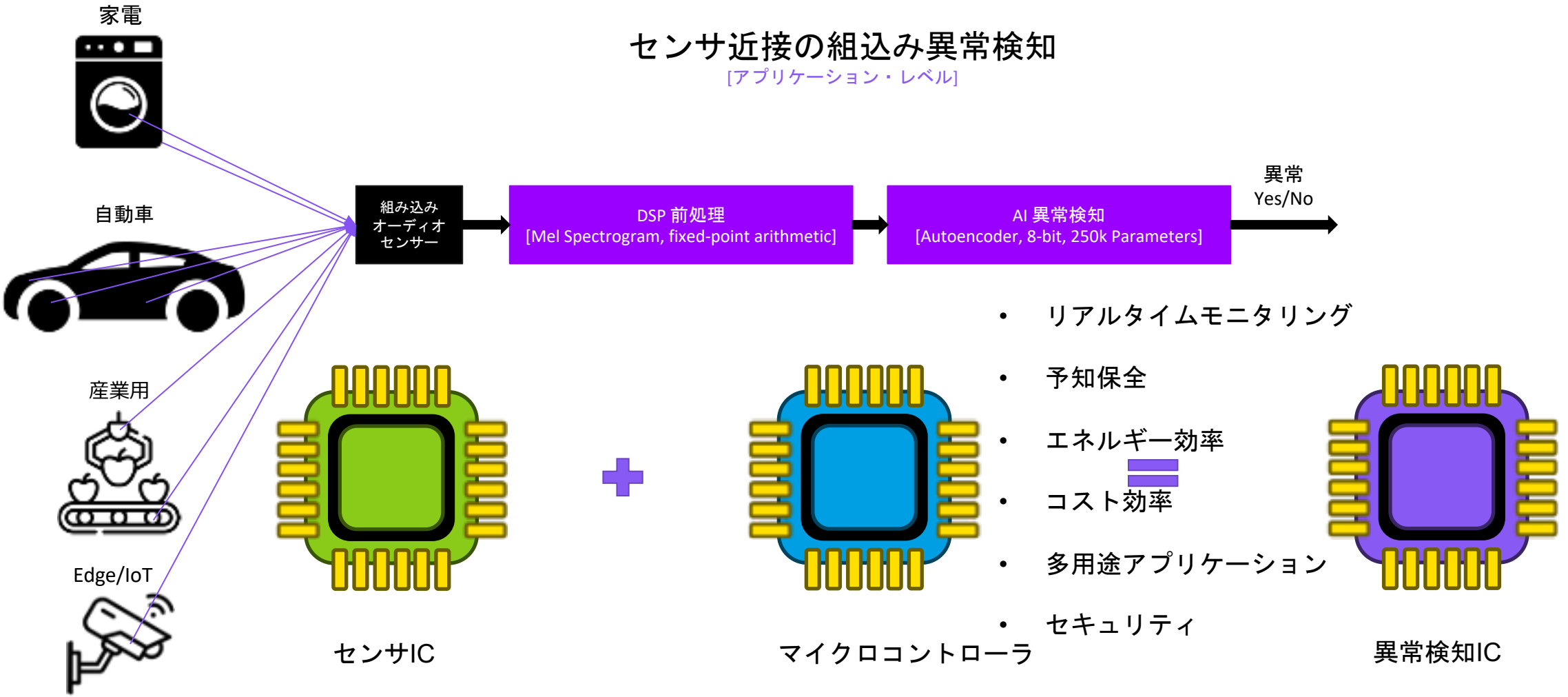
L110+CBI demo

→ 致命的な障害を防ぐ予知保全
異常検知

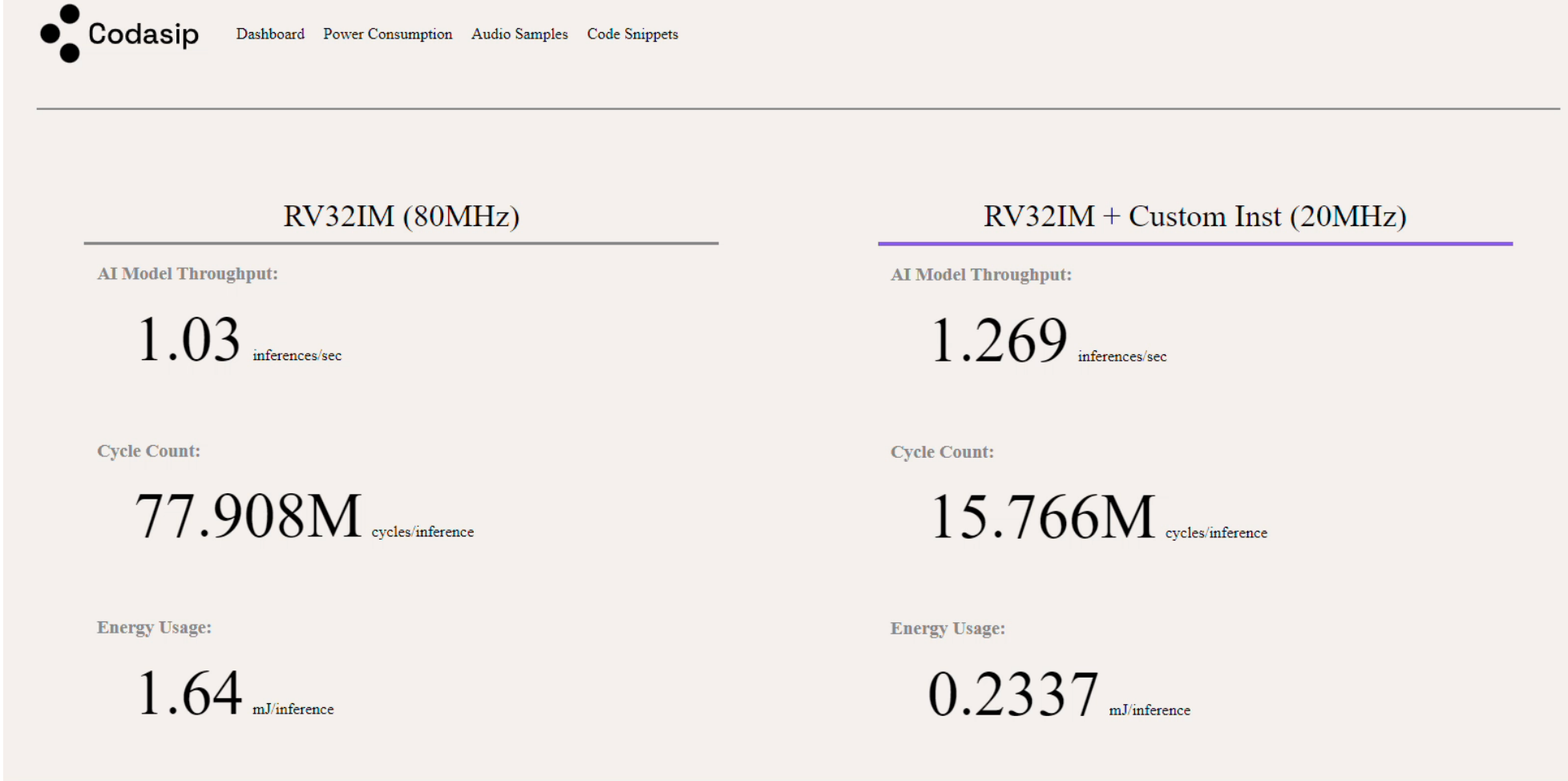
- 自動車:
  - 各車両は数十～数百の機器を搭載
- 製造:
  - 数千台のデバイスが使用されている工場
- 潜在的な故障点 の予知情報を提供する

→ 異常検知IC開発による製品差別化企画

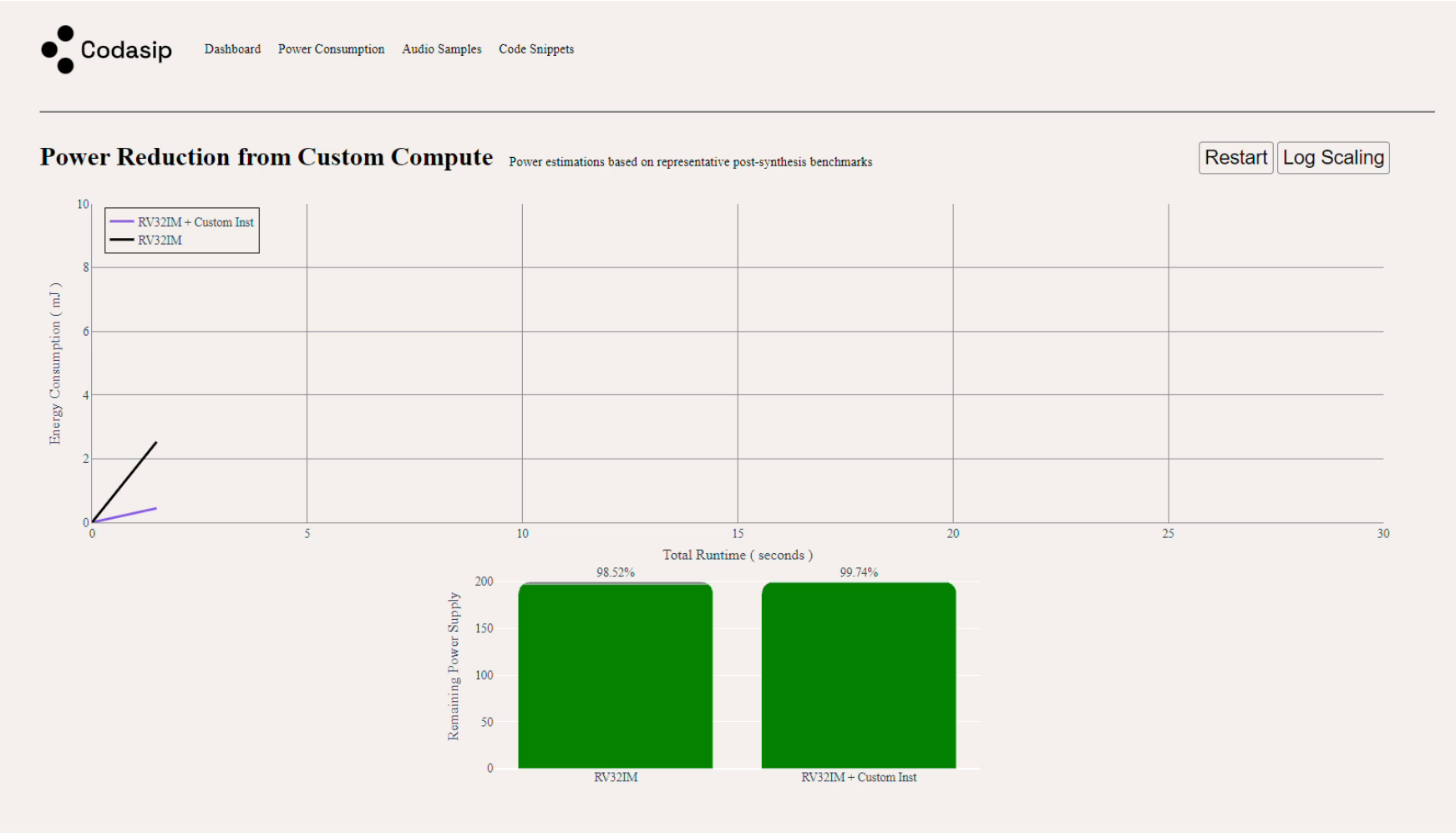
Codasip



→ クロックを下げ、消費電力を著しく改善



→ バッテリの持ちは7倍に



Codasip

↘

新製品
Codasip Studio Fusion

→ 設計とカスタマイズをシンプルにできる自動化技術
機能のパイプライン自動分割

Codasip

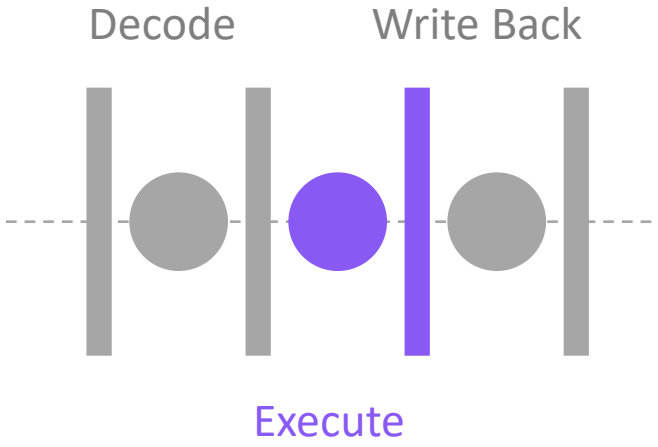
- 様々なパイプラインの設計を高速で探索
  - パイプラインの深さを変えて実験する場合に有用
- パイプライン・レジスタと制御信号の自動生成

```
stage pipe.DECODE_EXECUTE {
  alu_s1 = rf_gpr.r0[rs1];
  ex.alu.compute(alu_s1,…);
stage pipe.WRITE_BACK {
 …
```

→

```
stage pipe.DECODE {
  alu_s1 = rf_gpr.r0[rs1];
stage pipe.EXECUTE {
  ex.alu.compute(alu_s1,…);
stage pipe.WRITE_BACK {
 …
```

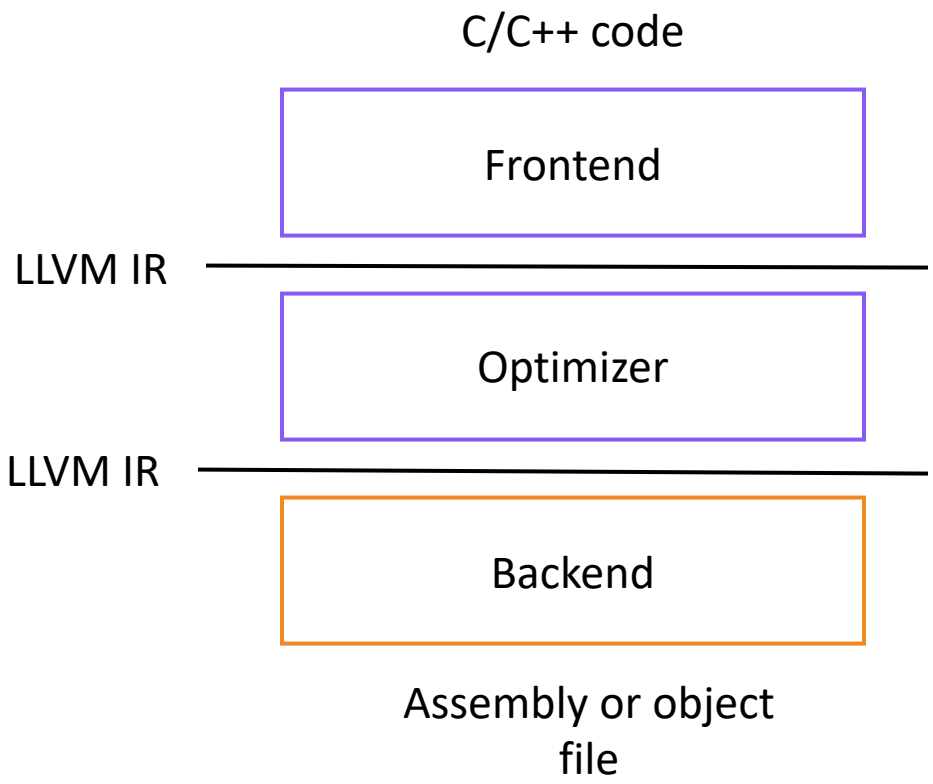
→



→ カスタム命令に追従し 自動生成されるLLVMコンパイラ

- CodasipはLLVMをベースラインとして使用
- カスタムC/C++コンパイラは、CodALプロセッサの記述から自動生成
- 他に以下のものも同時に生成
  - LLVM assembler
  - LLVM disassembler
  - LLVM linker
  - LLVM binutils
    - E.g., objdump
  - LLVM debugger (LLDB)
  - …
- 完全なSDK/ツールチェーンを生成

LLVM 構造



→ LLVMをエンハンスして使用

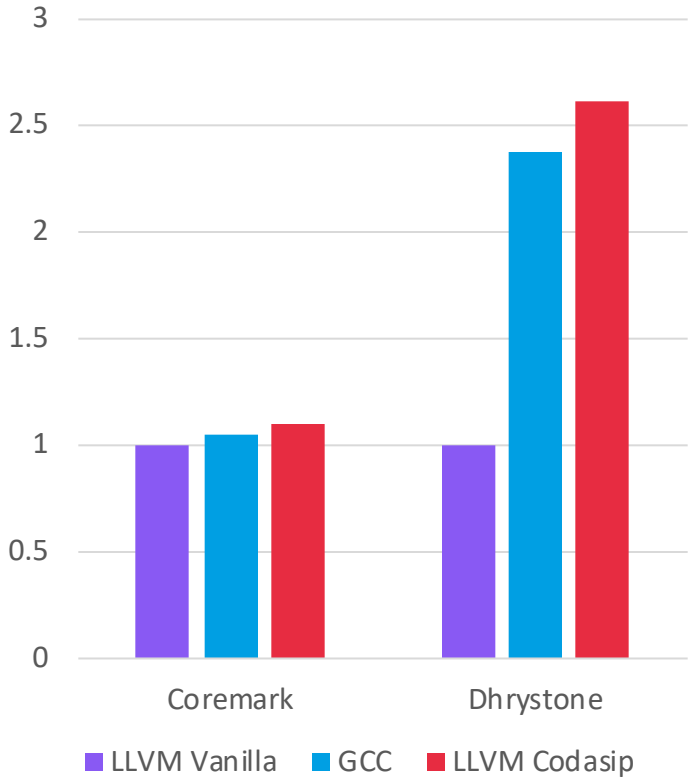
Codasip

エンハンス済コンパイラ

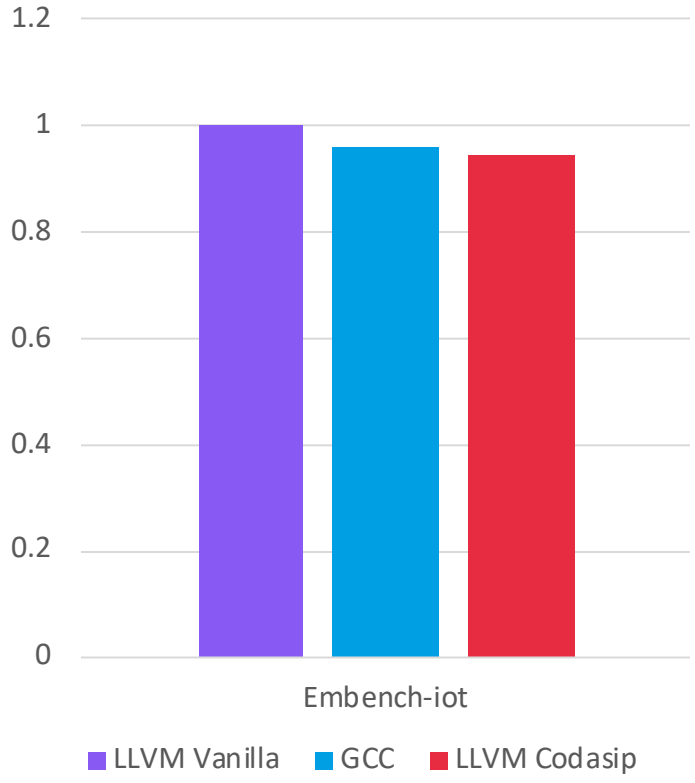
- より良いパフォーマンス
- コードサイズ縮小
- Vanilla LLVMに欠けているDSP機能
- 等

- LLVMはカスタムLLVMファイルを使ってチューニング可能

パフォーマンス改善



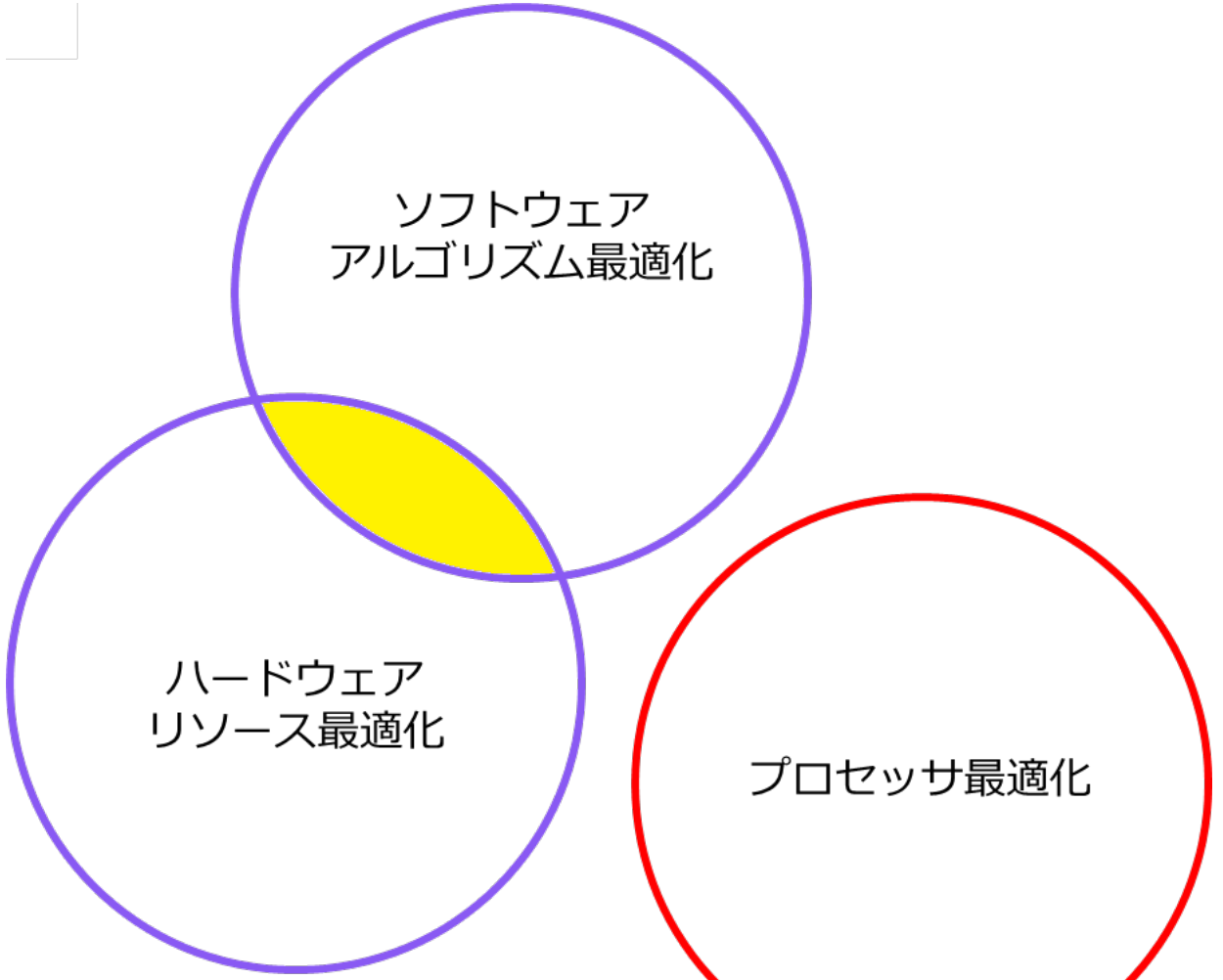
コードサイズ改善



→ 従来の開発における壁

- 従来の開発では、プロセッサ・アーキテクチャが固定されていることが多く、ソフトウェア・アルゴリズムとハードウェア・リソースの個別最適化に限定されていたことが多かった

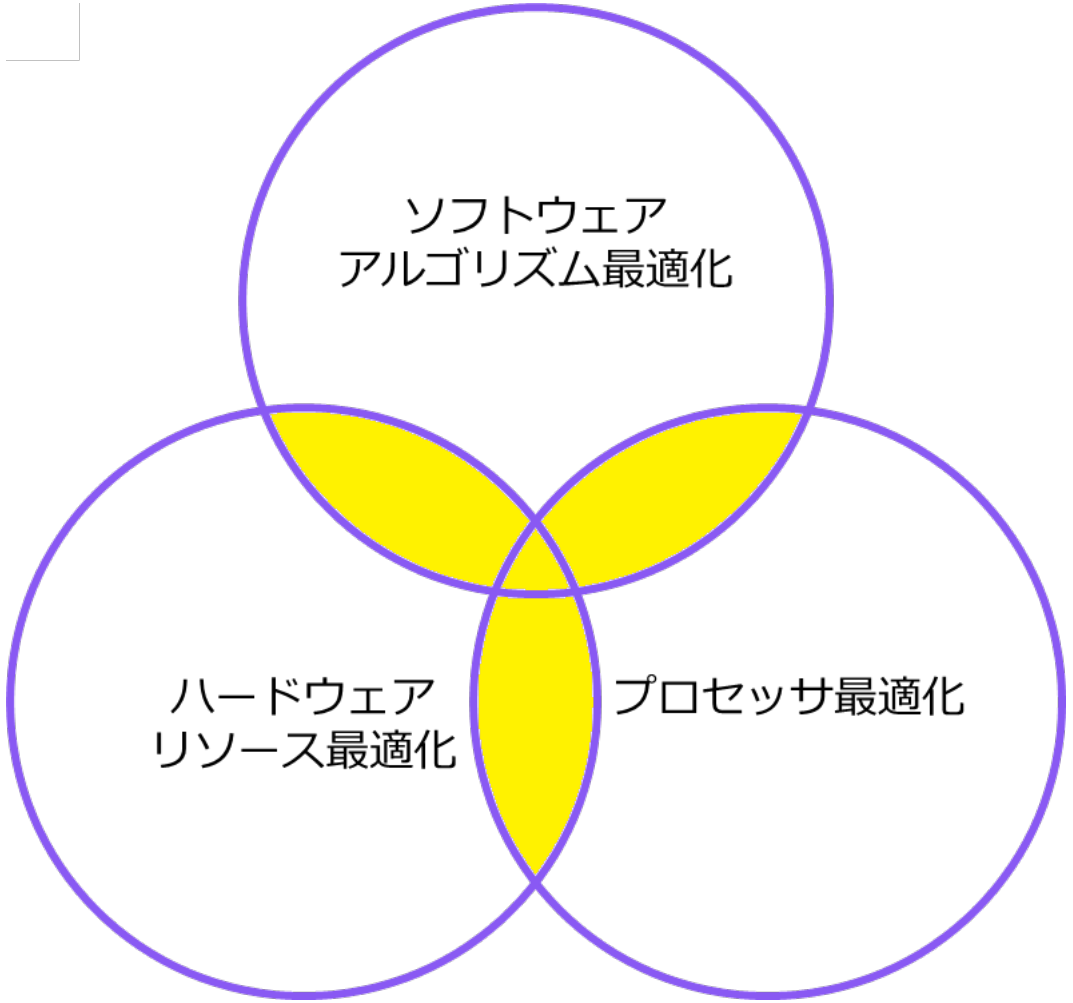
- 重要な要素であるプロセッサの最適化が考慮されていない場合も多い



→ 従来の開発における壁を突破する

Codasip

- RISC-V IPとCodasip Studioの組み合わせで、開発の壁を突破
- ソフトウェア・アルゴリズムを高速化するための新しい命令や、セキュリティ・プロトコルなどのハードウェア・リソースを緊密に結合させるなど、プロセッサに最適化のためのリソースを取り入れることが可能



Codasip

→

Thank you!

お問い合せは、contact_japan@codasip.com

→ SW/プロセッサ(RISC-V)協調設計ワークショップ

「Accelerating Neural Network Inference with SW/Processor Co-Design」

対象: RISC-V、その他プロセッサのカスタマイズに興味がある方

人数: 32名 (無料)

日時: 9月10日(火) 13:00-17:30

場所: 東京大学 目白台インターナショナル・ビレッジ ラーニングコモンズ

住所: 〒112-0015 東京都文京区目白台3-28-6

アジェンダ(予定):

- カスタムRISC-Vがもたらす価値
- カスタマイズ・メソドロジ
- SW/プロセッサ協調設計によるニューラルネットワーク推論の高速化
- Cloud版Codasip Studioを使用した演習も準備予定 (PCをご持参ください)

講師: Zdenek Prikryl (CTO, Codasip), Keith Graham (VP Univ. Program and Customer Experience, Codasip)

共催: 東京大学工学系研究科システムデザイン研究センター / Codasip GmbH

お申し込み/お問い合せ: contact_japan@codasip.com までご所属とご連絡先をメールお願いします.