→ カスタム コンピュート - 要求毎に最適化

Codasip

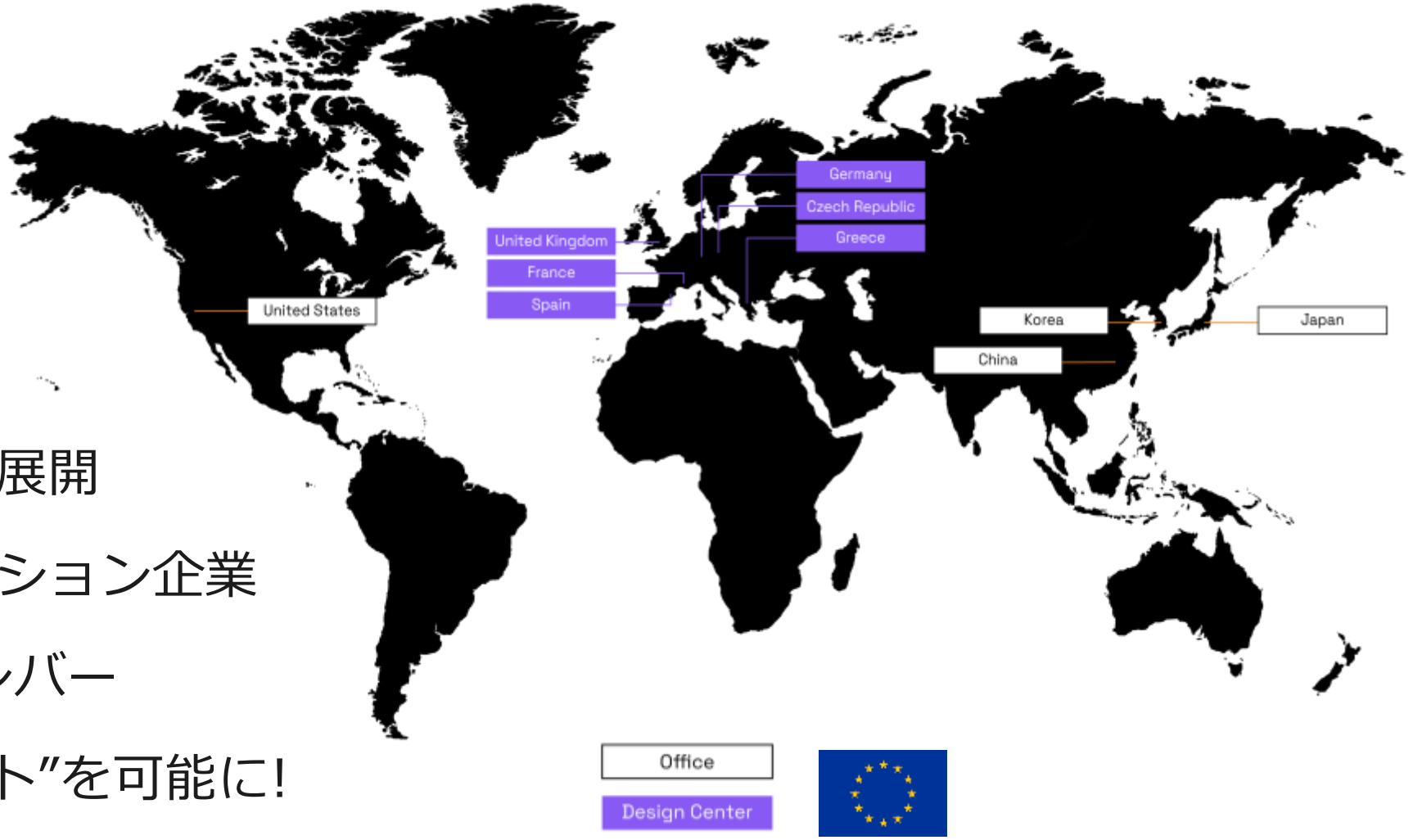
Architect your ambition

Takaaki Akashi, Country Manager - Japan

Jun 26, 2023

→ コダシップ概要

- 2014年設立
- 独ミュンヘン本社
- 約200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!



© 2023 Codasip. All rights reserved.

→ 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

© 2023 Codasip. All rights reserved.

→ 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。

© 2023 Codasip. All rights reserved.

→ 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な ISAである

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

Codasip

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

APACHE

MySQL

GCC

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 10Gbps

Bluetooth®

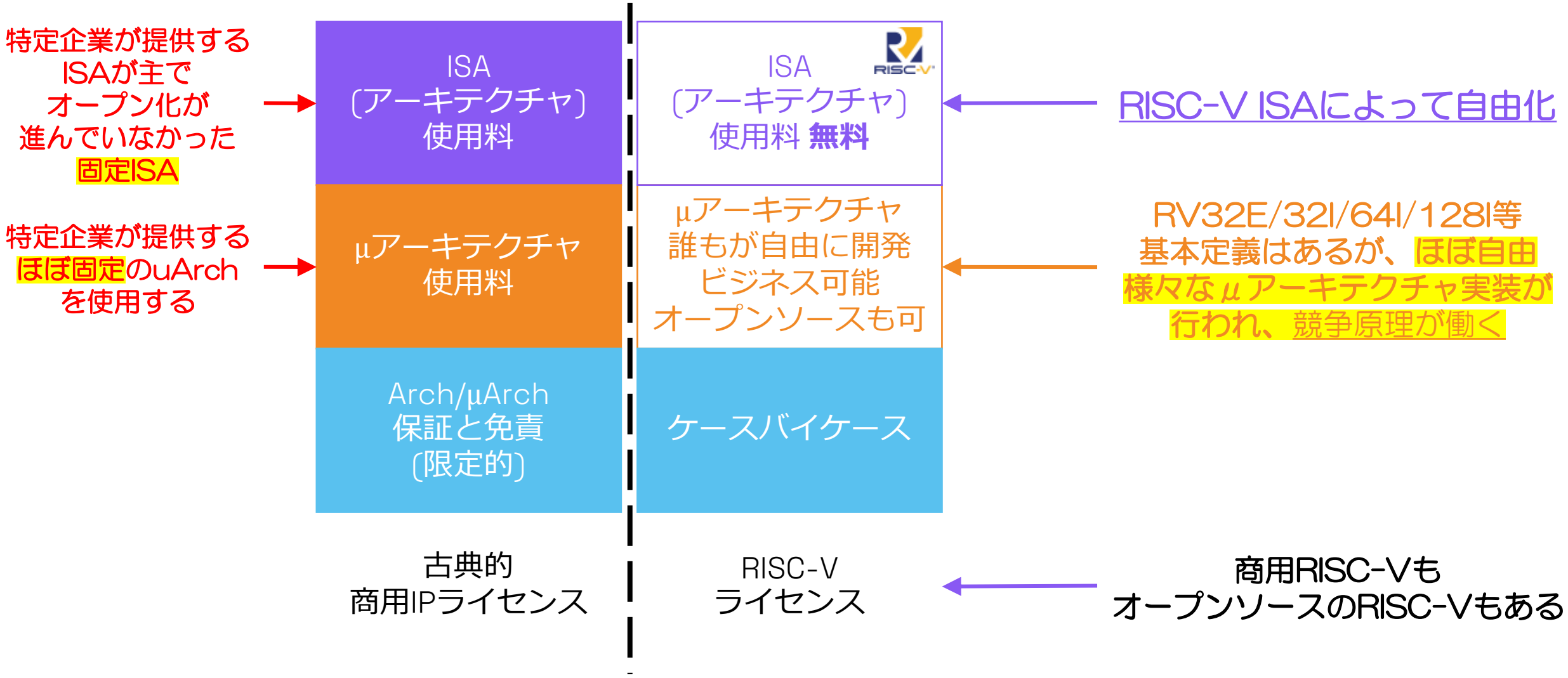
HTTP

SystemVerilog

© 2023 Codasip. All rights reserved.

→ プロセッサ ライセンスモデルの違い

Codasip



© 2023 Codasip. All rights reserved.

Codasip

↘
カスタム
コンピュートは未来

なぜCustom Computeが重要なのか
リーダーとしての地位を維持するために
それが必要な理由

© 2023 Codasip. All rights reserved.

→ 世界のシステムはプロセッサ上で稼働している

Codasip

全てのソフトウェアは、プロセッサの上で動いている

それにも関わらず、世界には僅かな種類の汎用プロセッサしか選択肢がなかった...

貴方の製品は競合に負けないスペックを手にしていますか？

あらゆる製品・サービスの進化とともに市場の競争は日々激化してきています

これまで高かったプロセッサ開発のハードルを下げました

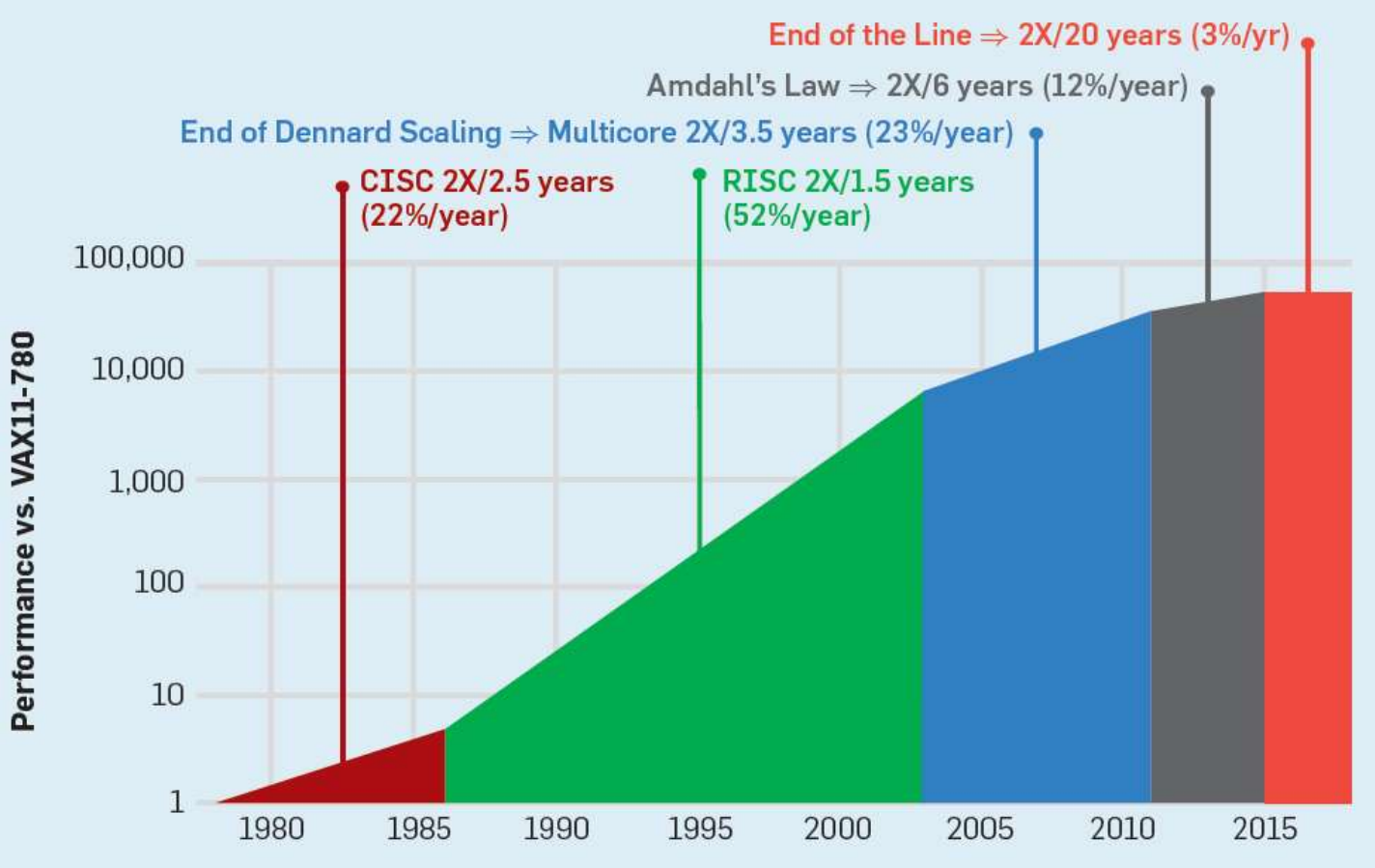
貴方だけの専用プロセッサを用いて、差別化開発しませんか？

Architect Your Ambition

© 2023 Codasip. All rights reserved.

Confidential

→ 汎用プロセッサ性能の飽和



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.

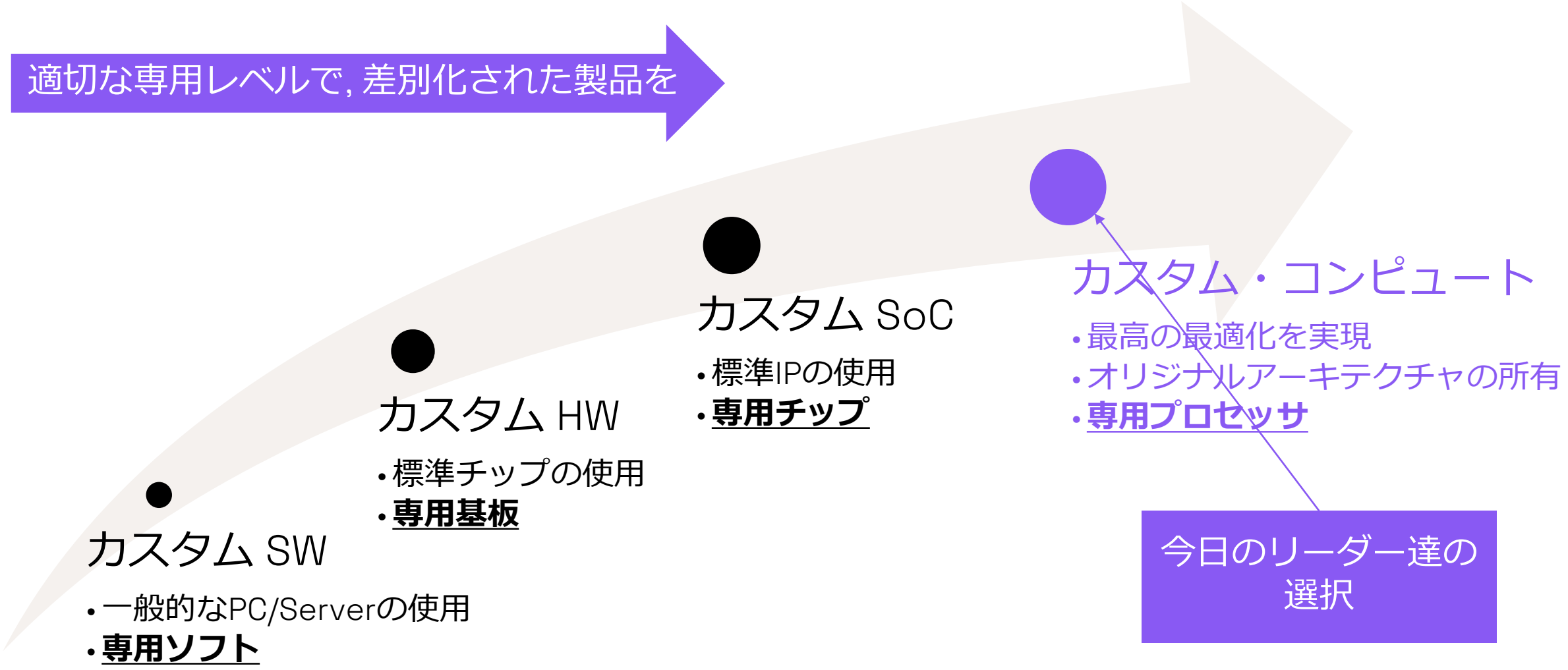
出典: RISC-V Spec Vol II: Privileged Architecture 20211203, Section 3.3.4
https://github.com/riscv/riscv-isa-manual/releases/download/Priv-v1.12/riscv-privileged-20211203.pdf

- 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」

© 2023 Codasip. All rights reserved.

→ 価値あるものに投資、所有する!



© 2023 Codasip. All rights reserved.

Codasip

コダシップは、お客様のカスタム・コンピュート実現を可能にします。
これを表しているのが当社のタグライン
“Architect your ambition” です

私たちは、お客様が真に差別化された製品を簡単に開発できるようにします

© 2023 Codasip. All rights reserved.

→ 命令セット(ISA: Instruction Set Architecture)って？

Codasip

- ソフトウェアとプロセッサの**インターフェース**
- ソフトウェアとプロセッサの**共通言語**

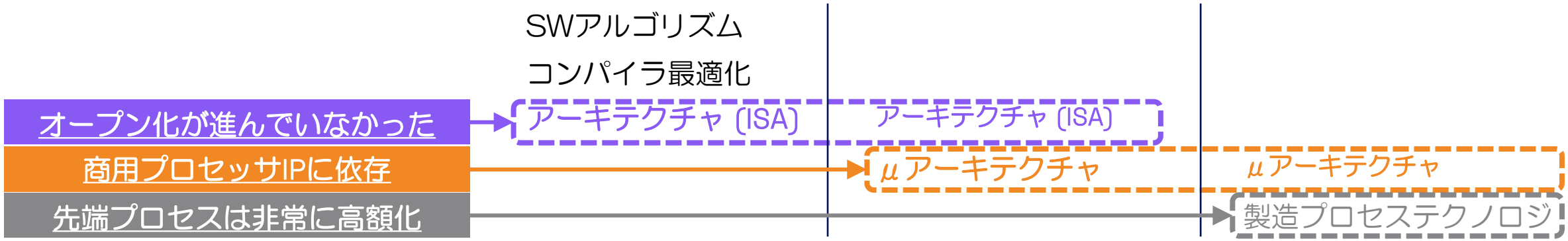
- 一般的に知られているものは:
  - x86, x86_64, ARMv7, ARMv8, mips, ppc, ~~alpha~~, ~~sparc~~, ~~IA64~~
  - **オープン化**が進んでいなかった

© 2023 Codasip. All rights reserved.

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

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

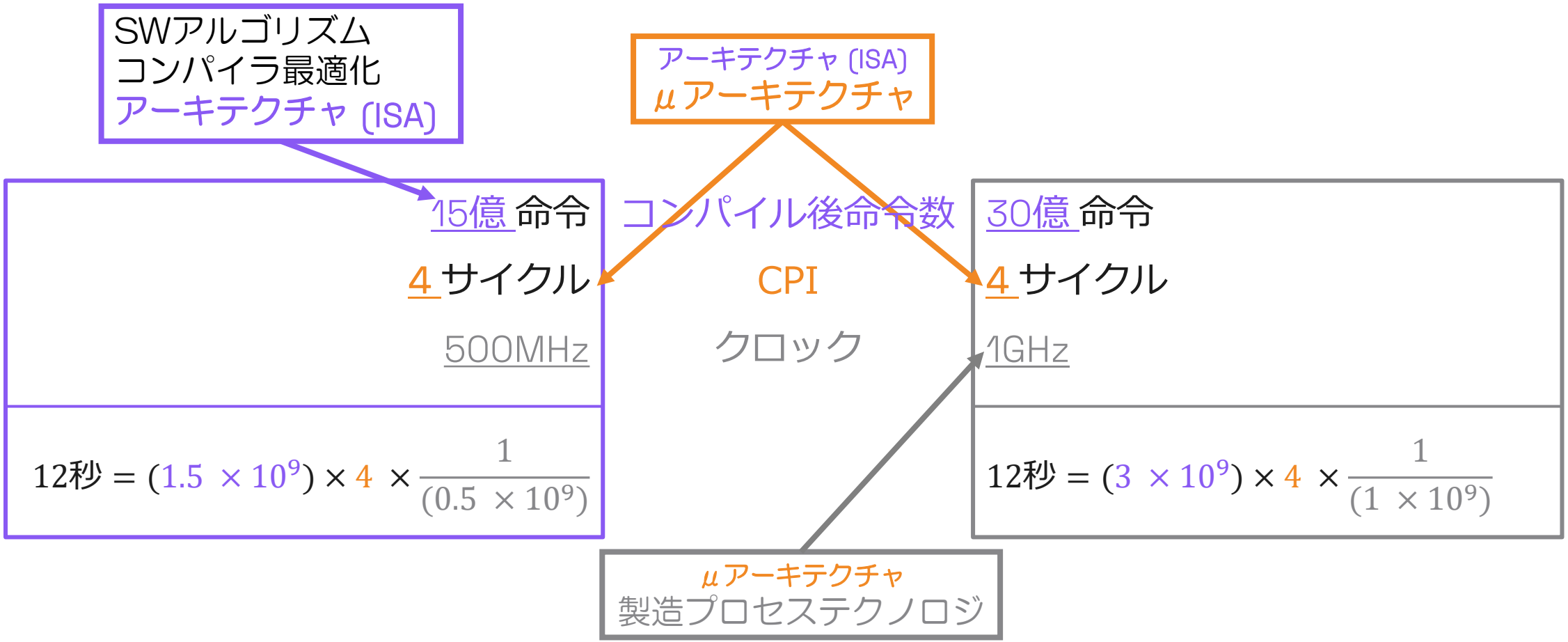
(プログラム->命令数) (CPI) (Cycle Time)



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

© 2023 Codasip. All rights reserved.

→ Iron law を用いた考察

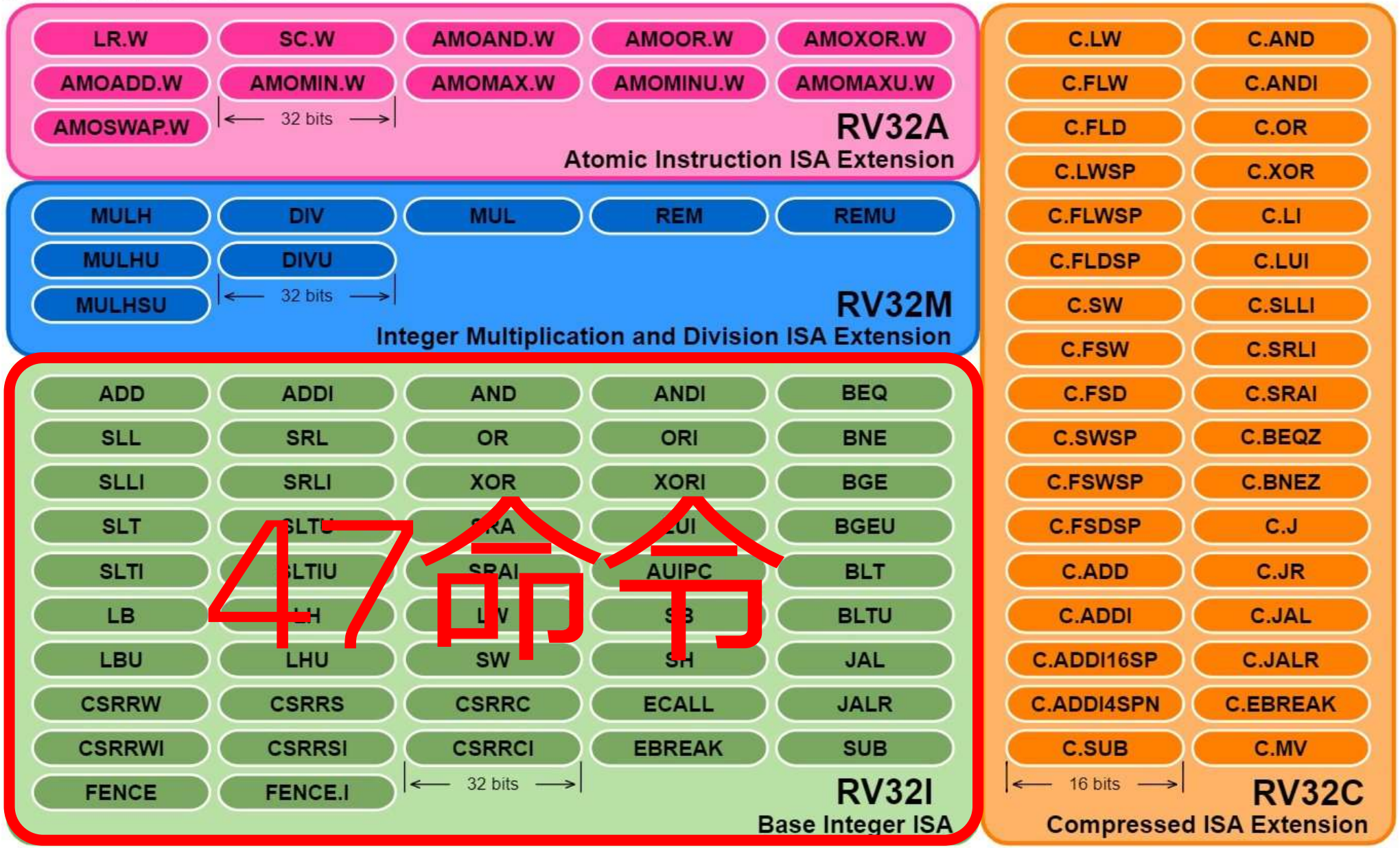


© 2023 Codasip. All rights reserved.

RV32IMAC

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

Codasip



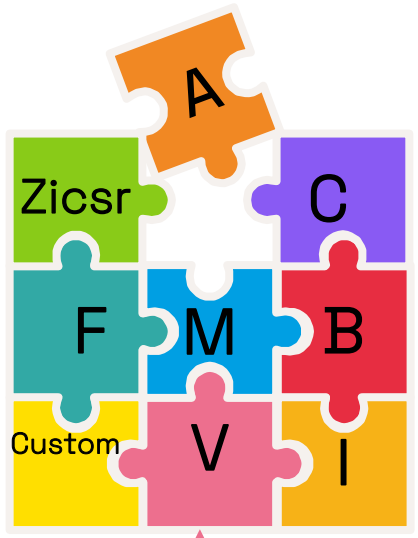
© 2023 Codasip. All rights reserved.

→ RISC-V 標準拡張

Codasip

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]:128	–	–	
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 "Hypervisor" 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

© 2023 Codasip. All rights reserved.

→ コンフィギュレーション と カスタマイズ

Codasip

Blog:
- プロセッサのコンフィグレーションとプロセッサのカスタマイズの違い
- プロセッサをカスタマイズする理由とその方法

コンフィギュレーション

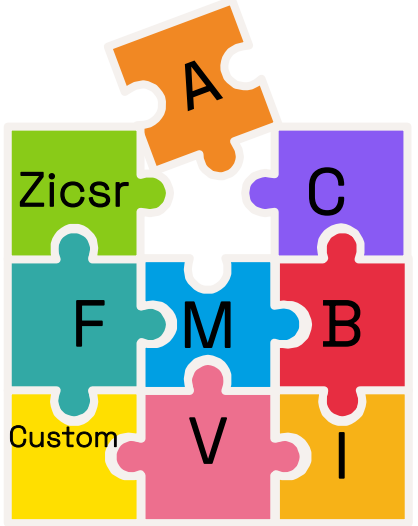
- キャッシュ、密結合メモリ、オンチップ デバッグ等

μアーキテクチャ

- **プロセッサのHW構造**

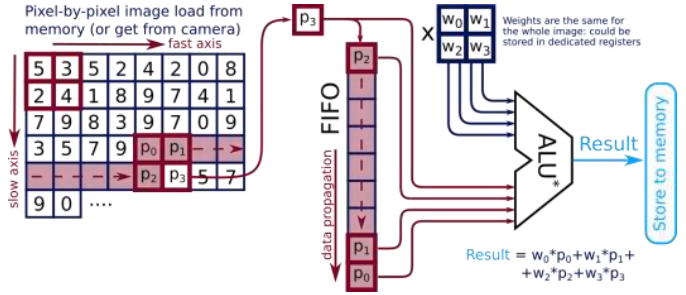
アーキテクチャ

- **モジュラーISA**



アーキテクチャ

- **専用命令追加や削除**



© 2023 Codasip. All rights reserved.

Codasip

Codasip Studio

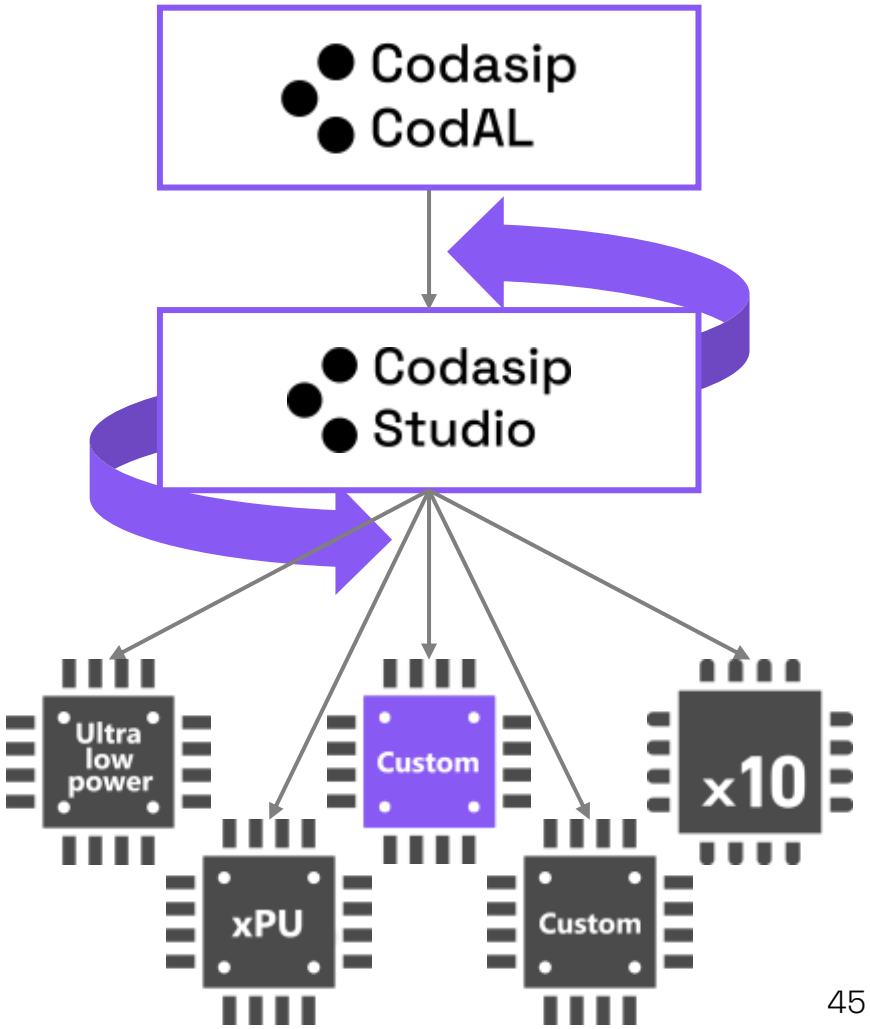
ADLレベルのユニークなEDAツール

© 2023 Codasip. All rights reserved.

→ ユニークなプロセッサ開発・カスタマイズツール

Codasip

- 命令セット・アーキテクチャ(ISA)探索・開発
  - IA(Instruction Accurate)レベル
- マイクロ・アーキテクチャ探索・開発
  - CA(Cycle Accurate)レベル
- RISC-V標準拡張の**コンフィギュレーション探索**
  - M, A, F, D, Q, C, etc
- **アーキテクチャ(ISA)探求**
- 既製Codasip RISC-Vプロセッサの**カスタマイズ**
- 新しいプロセッサをゼロから設計



© 2023 Codasip. All rights reserved.

→ コダシップはプロセッサ開発を自動化
– 速く, 安く, そして正確に

Codasip

古典的プロセッサ開発 (Codasip無)

GCC
手作業開発

Tool chain based on open source GNU

ISS
手作業開発

Own ISS

Verification
手作業開発

Verification environment and stimuli creation

RTL
手作業開発

Traditional RTL design

Codasipによる自動化

Codasip CodAL
手作業開発
もしくは
既製IP
C-based Processor Description Language

Codasip Studio
自動

自動生成 → LLVM 生成物

自動生成 → ISS 生成物

自動生成 → UVM 生成物

自動生成 → RTL 生成物

© 2023 Codasip. All rights reserved.

→ 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)

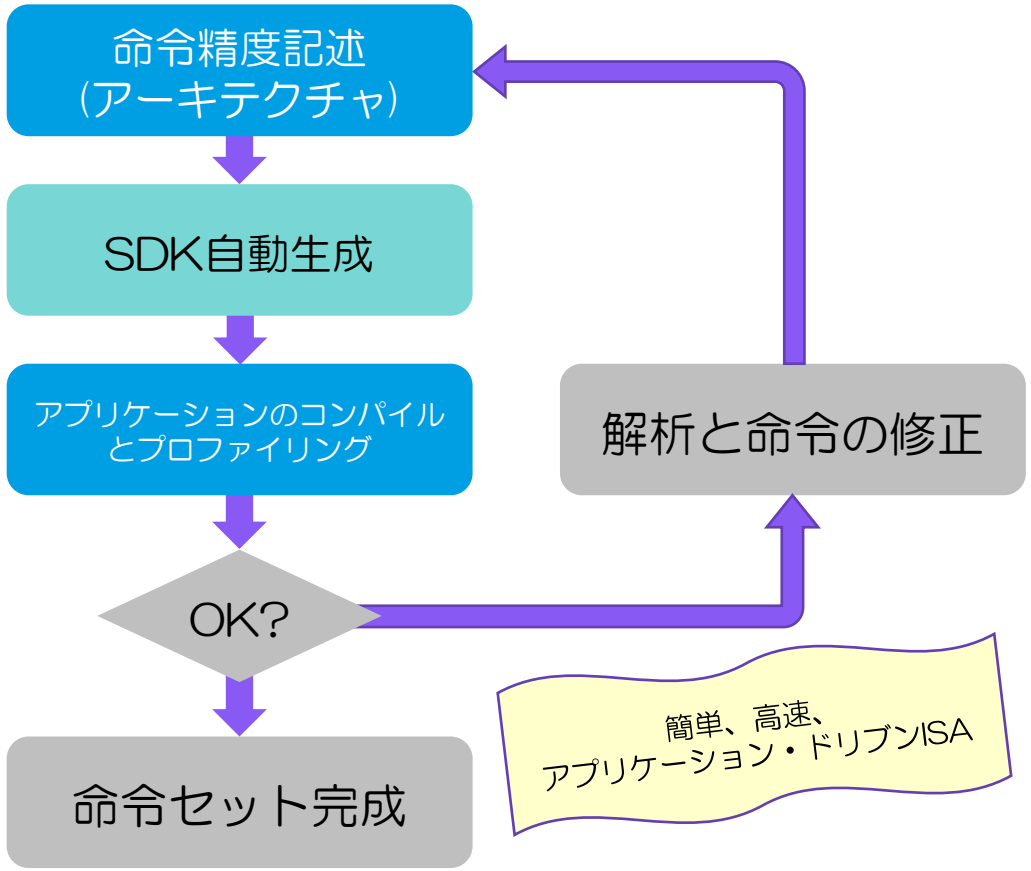
© 2023 Codasip. All rights reserved.

→ 命令セット(ISA)探求

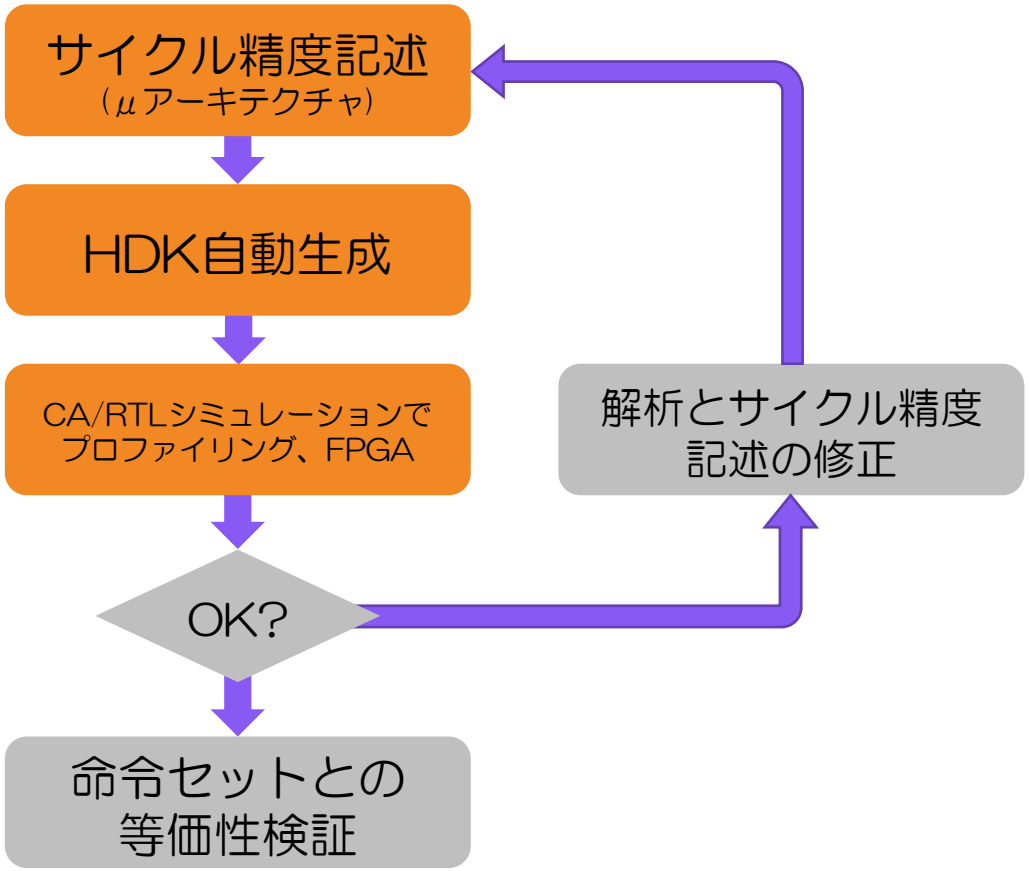
ハードウェア開発

Codasip

SDK in the loop (IAレベル)



HDK in the loop (CAレベル)



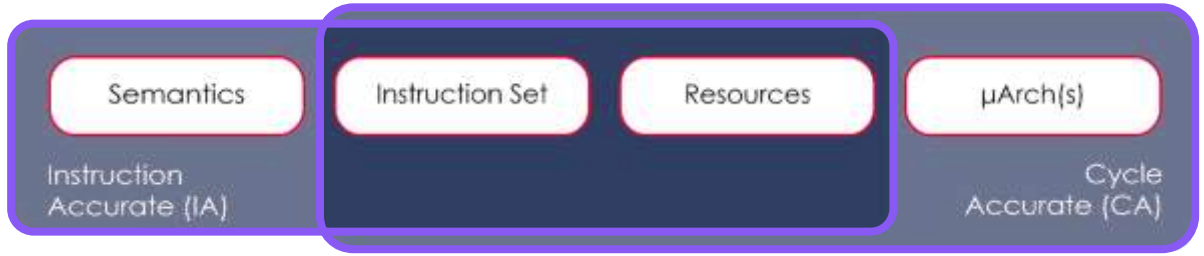
© 2023 Codasip. All rights reserved.

→ Codasip Architectural Language

Codasip

- プロセッサ開発に特化した言語
- 1つのCodALモデルからSDKとHDKを自動生成
  - CodALは、HDLのような汎用的設計に使われる言語とは異なります
- 一つのISAに対して、複数のμアーキテクチャの開発が可能

Codasip
CodAL




```
/*    Multiply and accumulate: semantics
                        dst += src1 * src2
*/

element i_mac {
            use reg as dst, src1, src2;
            assembly { “mac” dst “,” src1 “,” src2 };
            binary { OP_MAC dst src1 src2 0:bit[9] };
            semantics {
                        rf[dst] += rf[src1] * rf[src2];
            };
};
```

© 2023 Codasip. All rights reserved.

→ コンパクトで表現力豊かな言語

Codasip

SystemVerilog

```
module rf_gpr #(parameter xlen = 64, parameter size = 32,
  parameter resetval = 32'b0, localparam aw = $clog2(size))
( input wire clk, input wire rst, input wire w0_we,
  input wire [aw-1:0] w0_wa, input wire [xlen-1:0] w0_d,
  input wire r0_re, input wire [aw-1:0] r0_ra,
  output wire [xlen-1:0] r0_q, input wire r1_re,
  input wire [aw-1:0] r1_ra, output wire [xlen-1:0] r1_q );

reg [xlen-1:0] mem[size-1:0];
integer i;

always @(posedge clk or negedge rst)
if (~rst) begin
  for (i = 0; i < size; i = i + 1)
    mem[i] <= resetval;
end else if (w0_we) begin
  mem[w0_wa] <= w0_d;
end

assign r0_q = r0_re ? mem[r0_ra] : (xlen)'(0);
assign r1_q = r1_re ? mem[r1_ra] : (xlen)'(0);

endmodule
```

Codasip CodAL

```
arch register_file bit[32] rf_gpr
{
    dataport r0, r1 {flag = R;};
    dataport w0 {flag = W;};
    size = 32;
    reset = true;
    default = 0;
};
```

© 2023 Codasip. All rights reserved.

→ 可読性の高いHDLを生成

- CodAL記述から生成されるRTLは読みやすい
  - 前ページの手書きのコードと、右の自動生成コードを比べてみてください
- 生成されるRTLには、デバッグに有用な情報が含まれます
  - 生成されるRTLには、ソースであるCodALコードへのリンク

```
module rf_gpr(
    input  wire CLK,
    input  wire RST,
    input  wire r0_RE,
    input  wire [4:0] r0_RA,
    input  wire r1_RE,
    input  wire [4:0] r1_RA,
    input  wire w0_WE,
    input  wire [4:0] w0_WA,
    input  wire [31:0] w0_D,
    output wire [31:0] r0_Q,
    output wire [31:0] r1_Q
);
    localparam integer SIZE = 32'h00000020;
    localparam [31:0] DEFAULT_VALUE = 32'h00000000;
    // memory storage
    reg [31:0] RAM[0:SIZE-1];

    generate
        genvar ii;
        for ( ii = 32'sd0; ii < SIZE; ii = ii + 32'sd1 ) begin : WRITE_PROC
            always @( posedge CLK or negedge RST ) begin
                if ( RST == 1'b0 ) begin
                    RAM[ii] <= DEFAULT_VALUE;
                end else if ( (w0_WE == 1'b1) && (w0_WA == ii) ) begin
                    RAM[ii] <= w0_D;
                end
            end
        end
    endgenerate

    assign r0_Q = (r0_RE == 1'b1) ? RAM[r0_RA] : 32'h00000000;
    assign r1_Q = (r1_RE == 1'b1) ? RAM[r1_RA] : 32'h00000000;
endmodule
```

© 2023 Codasip. All rights reserved.

Codasip

Codasip RISC-V Processors

検証済み高品質 商用RISC-V IPコア

© 2023 Codasip. All rights reserved.

→ 既製プロセッサのポートフォリオ

- 直ぐに利用可能
- 検証済み、テープアウト品質のIP
  - お客様によるIPの検証は不要
- すべてのコダシップ プロセッサは RISC-Vに
  準拠しています
  - RISC-V特権仕様を実装
  - RISC-Vデバッグ仕様の実装
- 業界標準インターフェース
  - 命令バス、データバス用AMBA
  - デバッグ用JTAG (4ピン/2ピン)

© 2023 Codasip. All rights reserved.

1 Series

電力効率と面積効率の良い**3段パイプライン**

圧縮命令で最小のコードサイズを実現

16レジスタ内蔵命令セット (RV32E)

最小面積の**シーケンシャル乗算器**

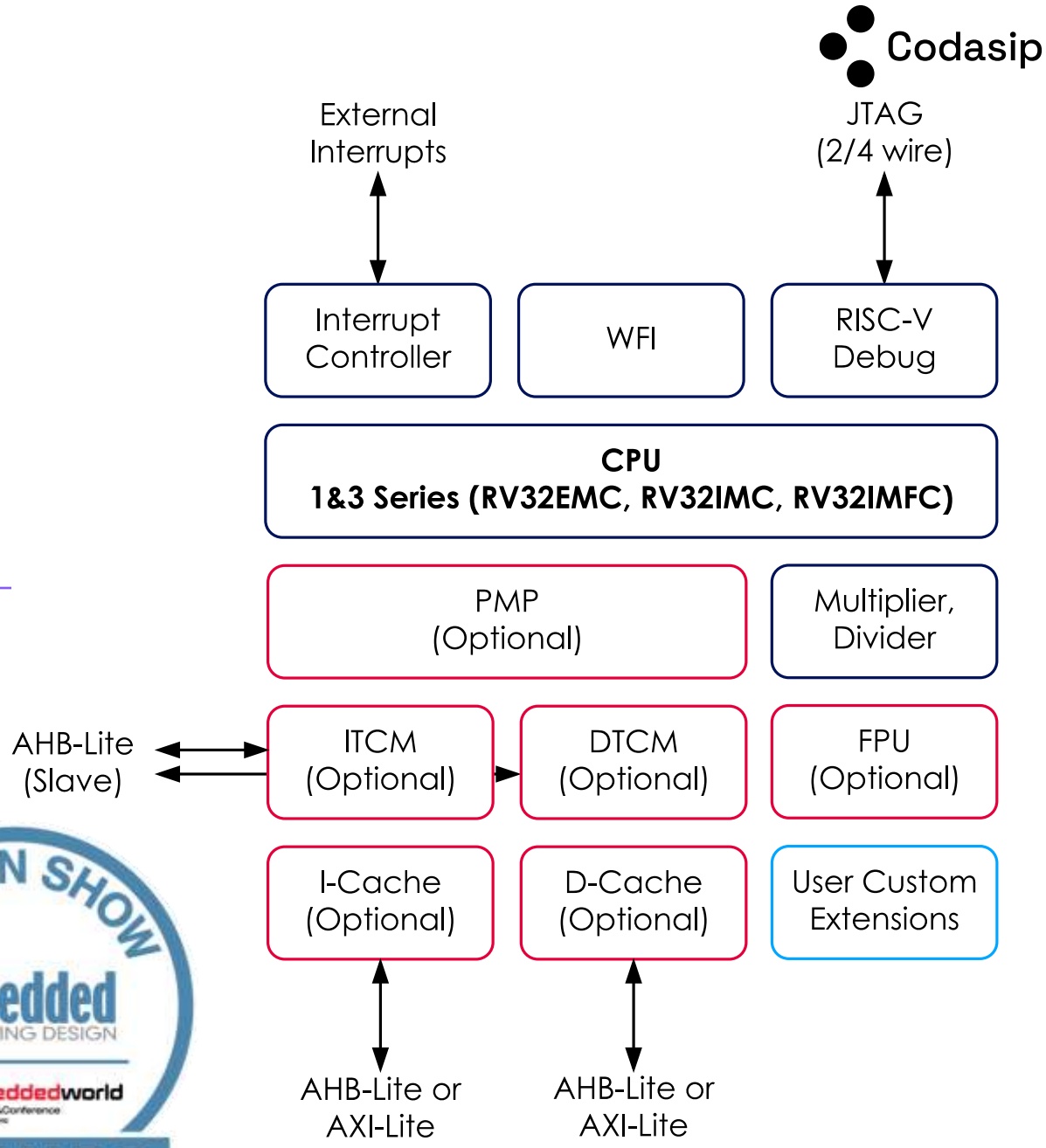
3 Series

電力効率と面積効率の良い**3段パイプライン**

圧縮命令で最小のコードサイズを実現

標準的32レジスタ内蔵命令セット (RV32I)

高性能な**並列乗算器**

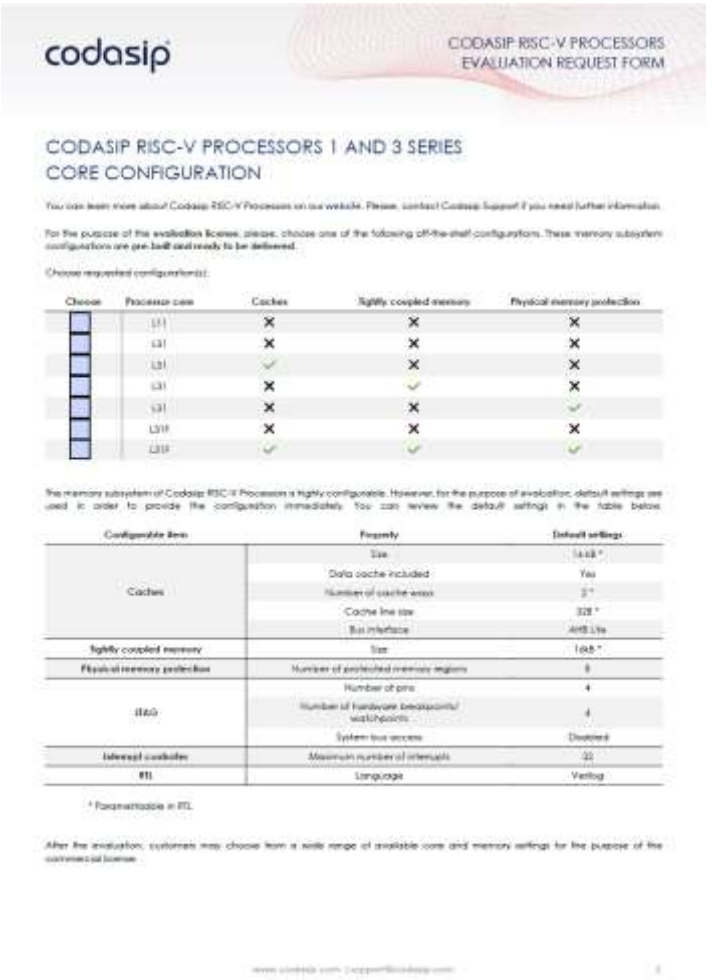


© 2023 Codasip. All rights reserved.

→ 幅広いコンフィギュレーション

- 3シリーズには全てのメモリオプション有

- コンフィギュラブル L1 キャッシュ
  - サイズ, キャッシュウェイ数,
    キャッシュラインのサイズ
- 密結合メモリ
  - 最大2MBまでカスタマイズ可能
- RISC-V PMP (Physical Memory Protection)
  - 最大16個


codasip

CODASIP RISC-V PROCESSORS
EVALUATION REQUEST FORM

CODASIP RISC-V PROCESSORS 1 AND 3 SERIES
CORE CONFIGURATION

You can learn more about Codasip RISC-V Processors on our website. Please, contact Codasip Support if you need further information.

For the purpose of the evaluation license, please, choose one of the following off-the-shelf configurations. These memory subsystem configurations are pre-built and ready to be delivered.

Choose requested configuration(s):

Choose	Processor core	Caches	Tightly coupled memory	Physical memory protection
☐	L11	✕	✕	✕
☐	L31	✕	✕	✕
☐	L31	✓	✕	✕
☐	L31	✕	✓	✕
☐	L31	✕	✕	✓
☐	L31F	✕	✕	✕
☐	L31F	✓	✓	✓

The memory subsystem of Codasip RISC-V Processors is highly configurable. However, for the purpose of evaluation, default settings are used in order to provide the configuration immediately. You can review the default settings in the table below.

Configurable item	Property	Default setting
Caches	Size	16 kB *
	Data cache included	Yes
	Number of cache ways	2 *
	Cache line size	32B *
	Bus interface	AHB Lite
Tightly coupled memory	Size	16kB *
Physical memory protection	Number of protected memory regions	8
JTAG	Number of pins	4
	Number of hardware breakpoints/ watchpoints	4
	System bus access	Disabled
Interrupt controller	Maximum number of interrupts	32
RTL	Language	Verilog

* Parametrizable in RTL

After the evaluation, customers may choose from a wide range of available core and memory settings for the purpose of the commercial license.

Example config, some items subject to change

© 2023 Codasip. All rights reserved.

→ 7 Series

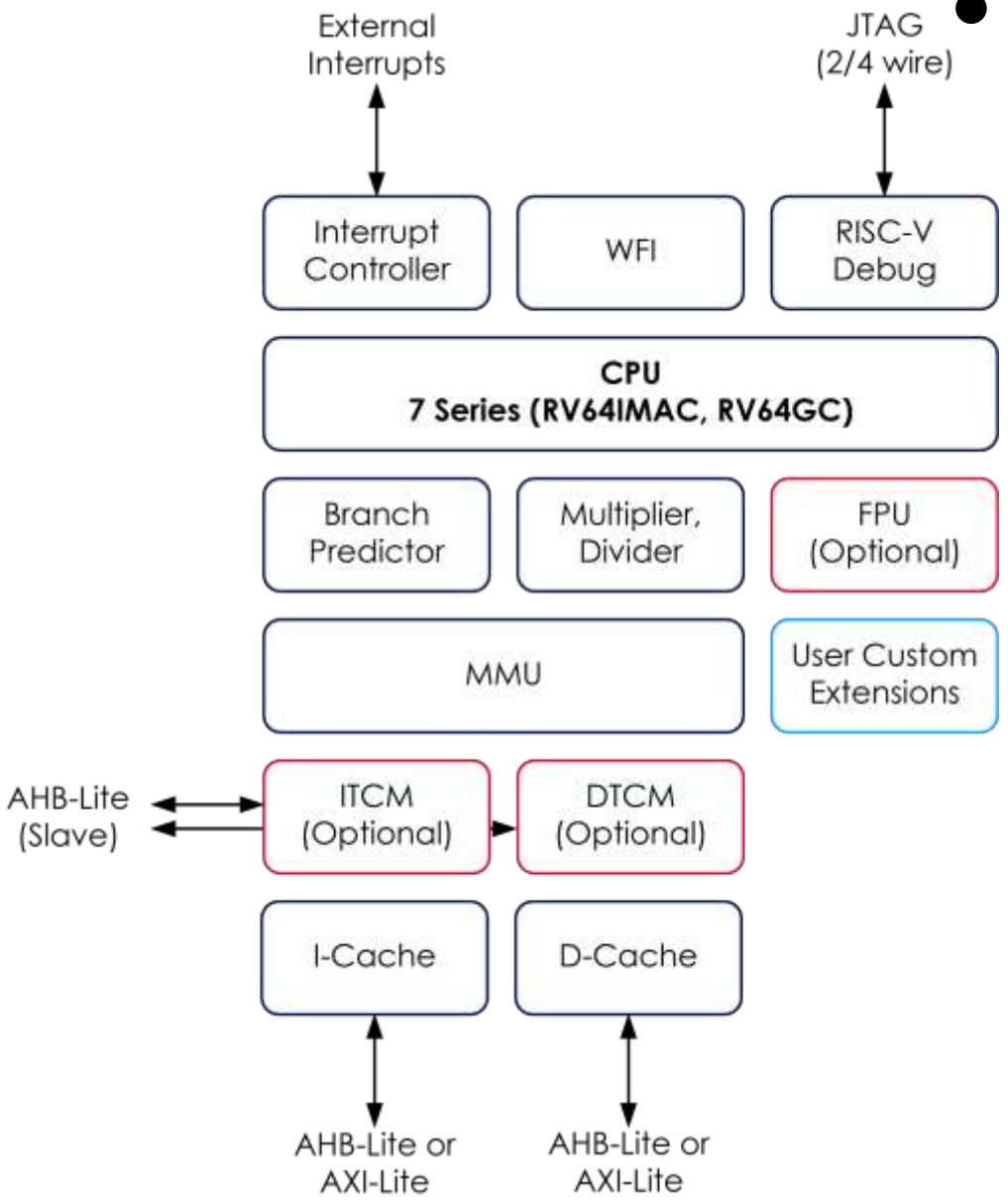
Codasip

複雑な7**段パイプライン**

圧縮命令で最小のコードサイズを実現

動的な**分岐予測**

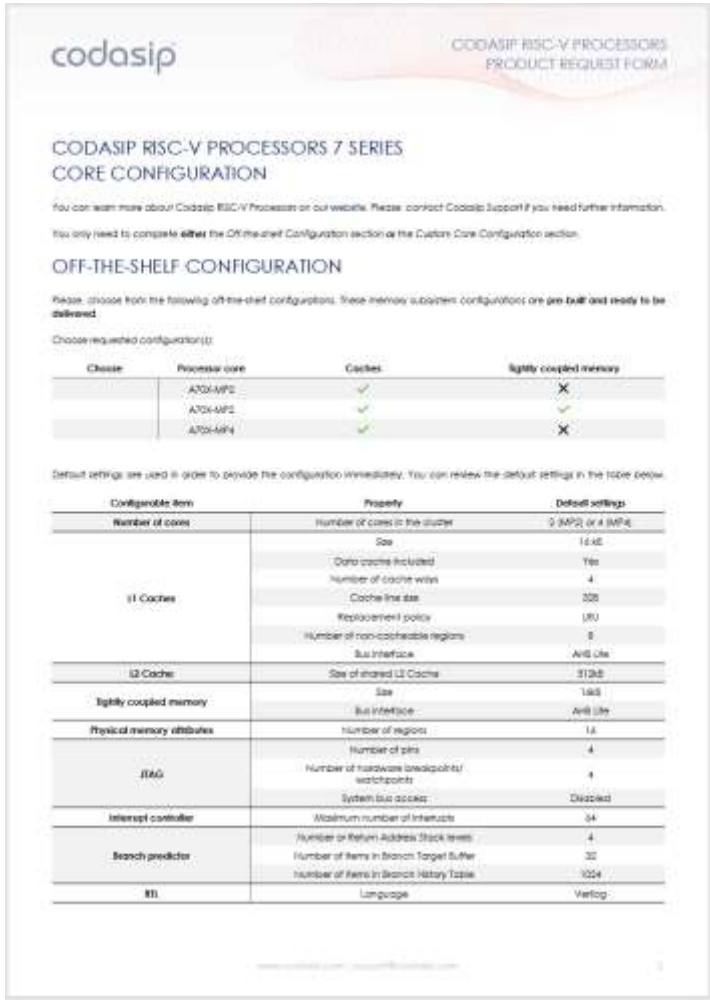
2/4コア有



© 2023 Codasip. All rights reserved.

→ 幅広いコンフィギュレーション

- A70コア・コンフィギュレーション
  - Local memories (L1 cache)
  - Branch predictor and more
- コア数
  - Up to 4 identical cores
- コンフィギュレーション可能なL2キャッシュ
  - Up to 8MB
  - 64-bit AXI interface
- コンフィギュラブルな割り込み制御
  - Up to 1000 interrupts



codasip

CODASIP RISC-V PROCESSORS
PRODUCT REQUEST FORM

CODASIP RISC-V PROCESSORS 7 SERIES
CORE CONFIGURATION

You can learn more about Codasip RISC-V Processors on our website. Please contact Codasip Support if you need further information.

You only need to complete **either** the *Off-the-shelf Configuration* section **or** the *Custom Core Configuration* section.

OFF-THE-SHELF CONFIGURATION

Please, choose from the following off-the-shelf configurations. These memory subsystem configurations are **pre-built and ready to be delivered**.

Choose requested configuration(s):

Choose	Processor core	Caches	Tightly coupled memory
	A70X-MP2	✓	✗
	A70X-MP2	✓	✓
	A70X-MP4	✓	✗

Default settings are used in order to provide the configuration immediately. You can review the default settings in the table below.

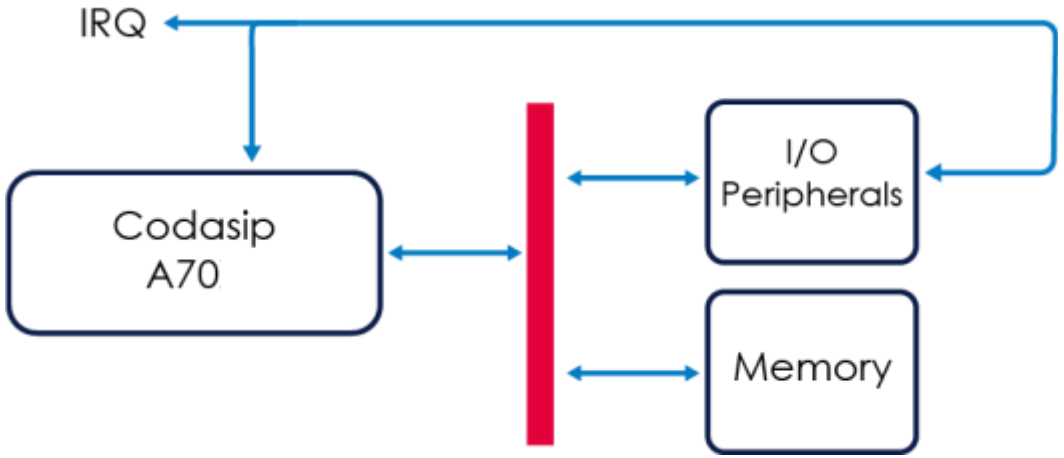
Configurable item	Property	Default settings
Number of cores	Number of cores in the cluster	2 (MP2) or 4 (MP4)
L1 Caches	Size	16 kB
	Data cache included	Yes
	Number of cache ways	4
	Cache line size	32B
	Replacement policy	LRU
	Number of non-cacheable regions	8
	Bus interface	AHB Lite
L2 Cache	Size of shared L2 Cache	512kB
Tightly coupled memory	Size	1MB
	Bus interface	AHB Lite
Physical memory attributes	Number of regions	16
JTAG	Number of pins	4
	Number of hardware breakpoints/ watchpoints	4
	System bus access	Disabled
Interrupt controller	Maximum number of interrupts	64
Branch predictor	Number of Return Address Stack levels	4
	Number of items in Branch Target Buffer	32
	Number of items in Branch History Table	1024
RTL	Language	Verilog

© 2023 Codasip. All rights reserved.

→ Codasip RISC-VコアFPGA評価プラットフォーム

Codasip

- RTOSまたはLinuxが動作するプロセッササブシステム
  - Codasip RISC-Vプロセッサを簡単に評価可能
  - ソフトウェアのRISC-Vへの移植に有用
- FPGAプラットフォームの推奨設定
  - Digilent Genesys2: MP4 or large MP2
  - Digilent Nexys-A7-100T
  - Digilent JTAG-HS2プログラミングケーブル
    - SeggerやIARのケーブルを使用可能
- 15分以内に起動可能
- 推奨FPGAボードのステップバイステップのクイック・スタート・ガイドを提供



© 2023 Codasip. All rights reserved.

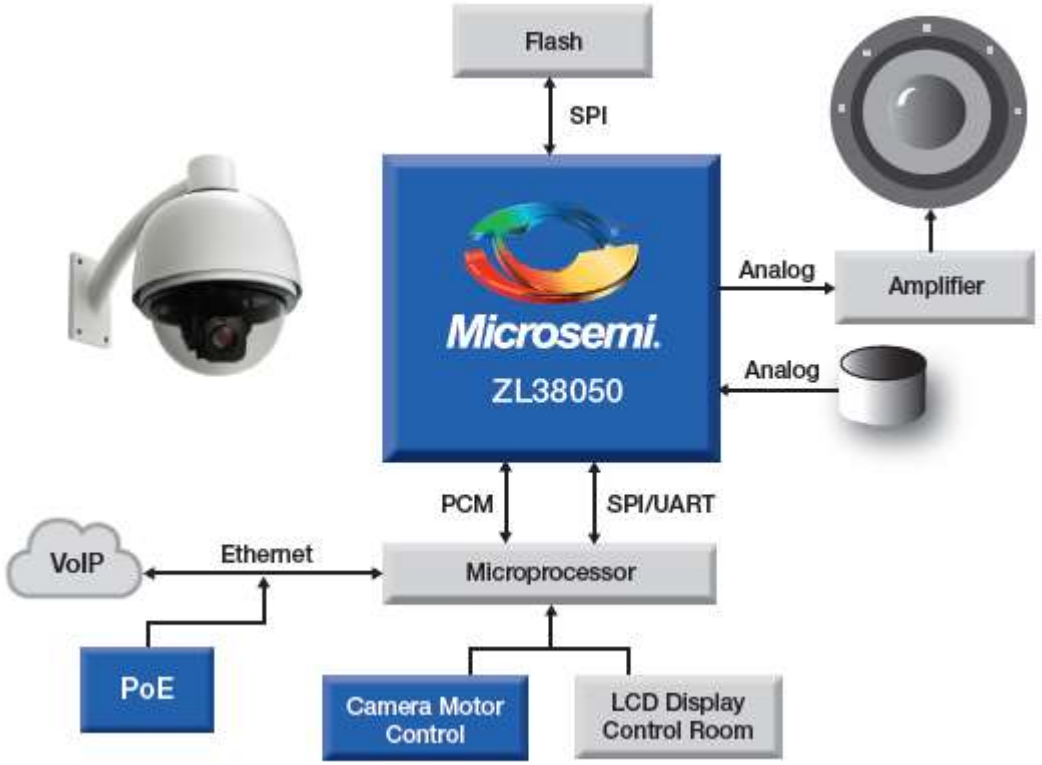
Codasip

↘
ケーススタディ

© 2023 Codasip. All rights reserved.

→ ケーススタディ: Microsemi社 (現Microchip社)

- オーディオ・イコライザー・アルゴリズム
- **要求事項**
  - 低価格化
    - 最適ROIの製造プロセスノード選択
    - IPライセンス料、ロイヤリティ等
  - 低消費電力（低動作周波数化等）
  - 多様な要件に対応する派生デザイン開発容易化
  - Time-to-market
    - デザインサイクル9-12ヶ月
  - Cortex-Mの置換え
- Codasip Studio**でのデザイン探求**
  - RV32Iから開発スタート
  - M拡張命令追加、 後にカスタムDSP命令追加

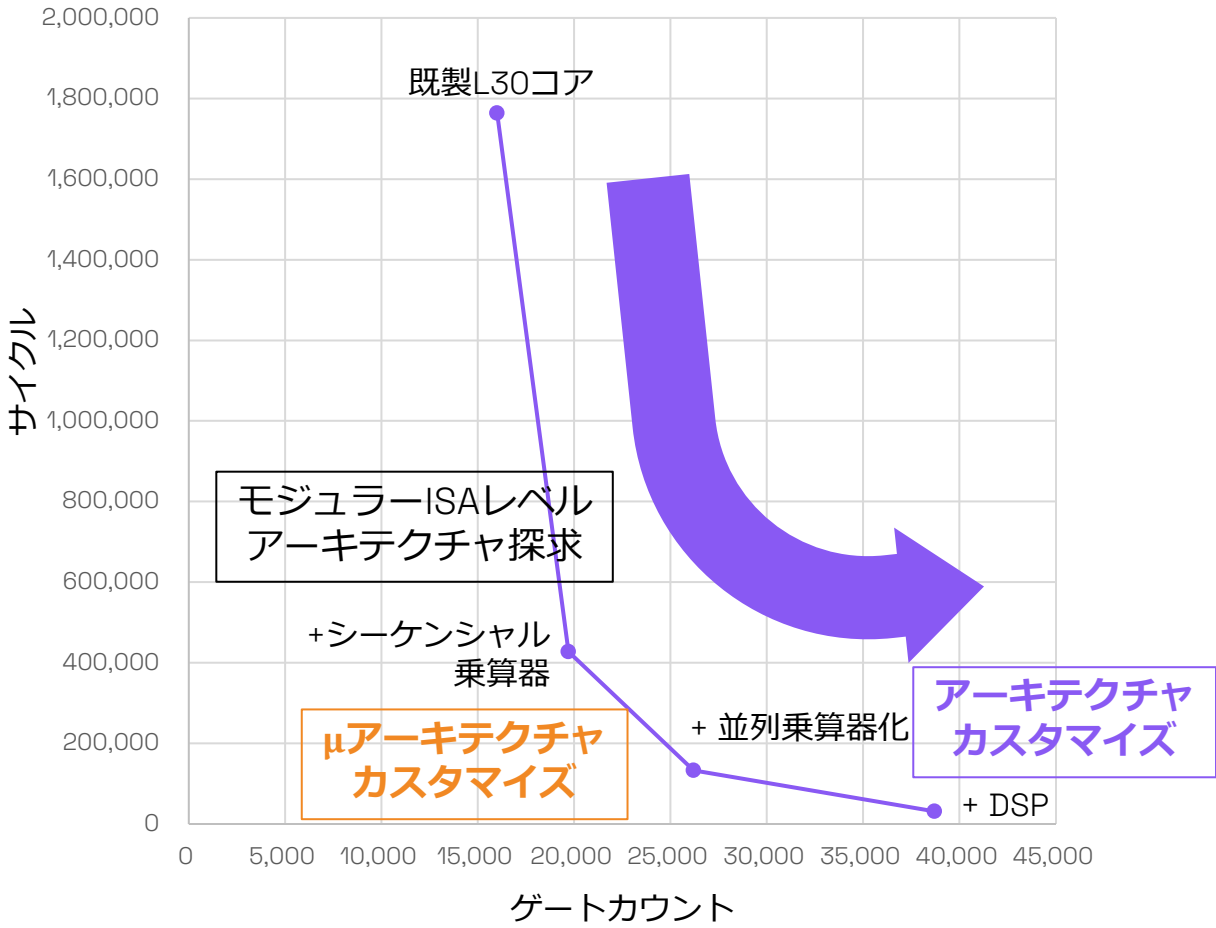


© 2023 Codasip. All rights reserved.

→ カスタム命令を追加するメリット

Codasip

- スループット 56.24倍 向上
- コードサイズ 3.62倍 圧縮
- ゲートカウント 2.43倍 増加
  - プロセッサコアではなく、命令メモリが領域で支配的であることに注意
  - いずれもベースのL30コアとの比較
- 古い製造プロセスノードでシリコン化することで大幅なコスト削減を実現



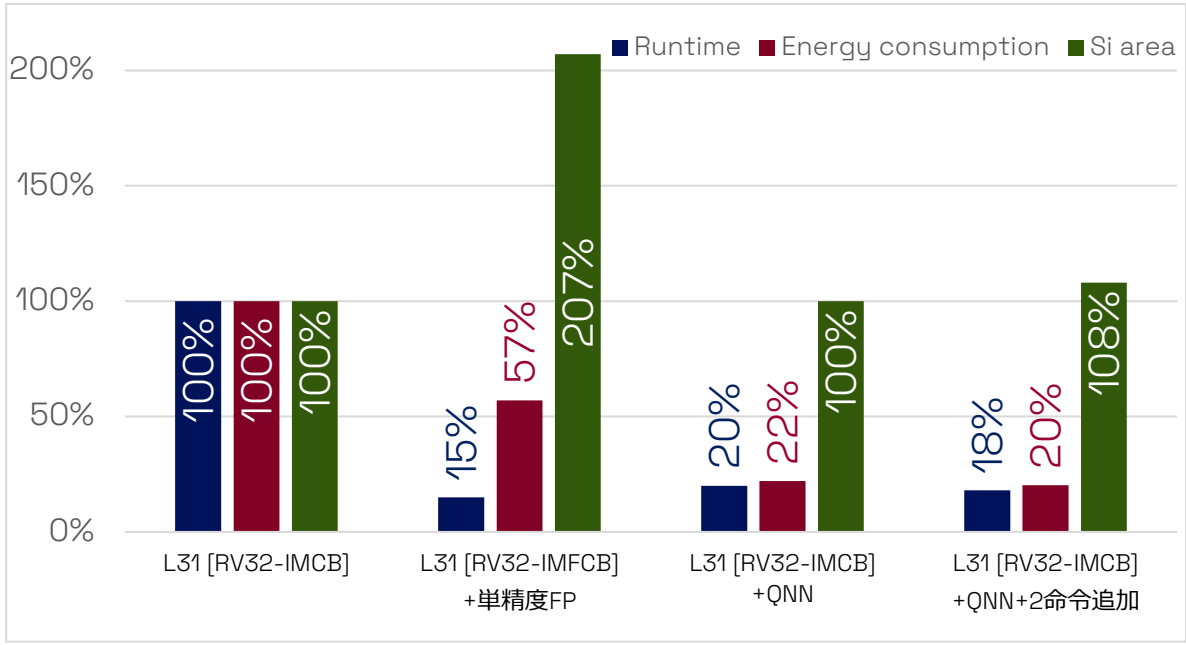
シンプルに、速く、正確に、デザインスペースの探求

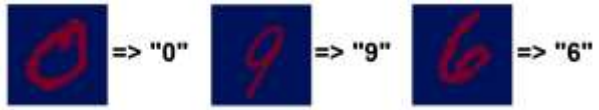
... 数日で完了

© 2023 Codasip. All rights reserved.

→ ケーススタディ: AI/ML

- TensorFlow Lite for MicrocontrollersをL31コアで実行
- IoT/エッジ・アプリケーションに最適化
  - 低消費電力化 / 限定的なメモリ量
- AIによる手書き文字認識 (MNIST)
  - ランタイムの削減 80%
  - 低消費電力化 78%
  - エリア増加**インパクト無**





MNIST “Handwritten Digits recognition” benchmark intelligent label assignment to grayscale 28x28 image.

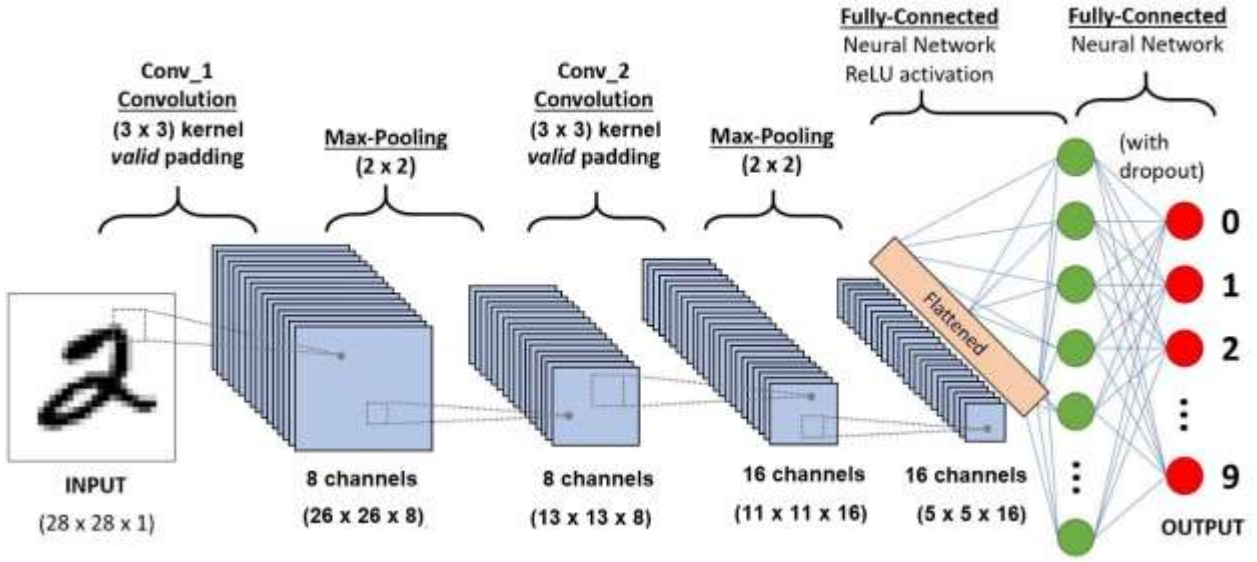
ホワイトペーパー（日本語有）
https://codasip.com/2022/02/24/embedded-ai-on-l-series-cores/

Symbol	Address	Instructions	Instructions Percent	Cycles	Cycles Percent
tflite::reference_integer_ops::ConvPerChannel	36fa6	6572379	86.3 %	9340321	83.9 %
tflite::reference_integer_ops::MaxPool	45e60	412255	5.4 %	710898	6.4 %
tflite::reference_integer_ops::FullyConnected	3e388	158370	2.1 %	236154	2.1 %

Codasip Studio**が生成するシミュレータを用いて、**
どのアルゴリズムが重要で、どこを最適化すべきかを簡単に確認できます

© 2 ved.

→ Profiling TFLite Image classification model



Source Code Coverage

Symbol	Address	Instructions	Instructions Percent	Cycles	Cycles Percent
tflite::reference_integer_ops::ConvPerChannel( )	2940c	8878981	89.9 %	15388368	90 %
tflite::reference_integer_ops::MaxPool( )	2b7e8	412616	4.2 %	710990	4.2 %
tflite::reference_integer_ops::FullyConnected( )	2a2de	157058	1.6 %	207101	1.2 %
...etc	26a60	95618	1 %	143346	0.8 %

```
   430   0.004%    for (int out_y = 0; out_y < output_height; ++out_y) {
                     const int in_y_origin = (out_y * stride_height) - pad_height;
  9092   0.092%     for (int out_x = 0; out_x < output_width; ++out_x) {
                       const int in_x_origin = (out_x * stride_width) - pad_width;
 46142   0.467%       for (int out_channel = 0; out_channel < output_depth; ++out_channel) {
 52460   0.531%         auto group = out_channel / filters_per_group;
 19562   0.198%         int32_t acc = 0;
596782   6.044%         for (int filter_y = 0; filter_y < filter_height; ++filter_y) {
                         const int in_y = in_y_origin + dilation_height_factor * filter_y;
390154   3.951%          for (int filter_x = 0; filter_x < filter_width; ++filter_x) {
132192   1.339%            const int in_x = in_x_origin + dilation_width_factor * filter_x;
                           // Zero padding by omitting the areas outside the image.
                           const bool is_point_inside_image =
                               (in_x >= 0) && (in_x < input_width) && (in_y >= 0) &&
                               (in_y < input_height);
435865   4.414%            if (!is_point_inside_image) {
                             continue;
                           }
491681   4.980%            for (int in_channel = 0; in_channel < filter_input_depth;
188064   1.905%                 ++in_channel) {
                             int32_t input_val =
                                 input_data[Offset(input_shape, batch, in_y, in_x,
                                                   in_channel + group * filter_input_depth)];
376128   3.809%              int32_t filter_val = filter_data[Offset(
                                 filter_shape, out_channel, filter_y, filter_x, in_channel)];
790788   8.009%              acc += filter_val * (input_val + input_offset);
                           }
                         }
                       }
```

- Image convolution (>89%) has a major impact on overall performance

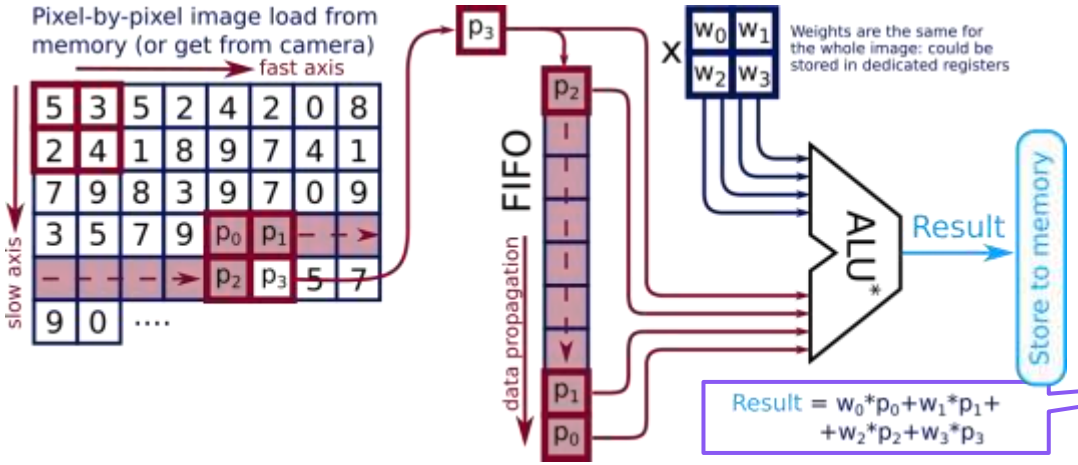
© 2023 Codasip. All rights reserved.

→ CONV accelerator in <200 lines of CodAL code

Codasip

element comprises convolution instruction assembly, binary and semantics in dedicated sections

push_conv instruction pushes image pixel from src to *fifo[]*, calculates the convolution result and stores it to the dst address



fifo[] is a hidden register file, load and store addresses are incremented automatically, thanks to fifo_counter and out_counter that contain the number of input image pixels loaded and output image pixels processed

Single instruction gets the convolution result (2x2 window)

```
element inst_push_conv
{
    use gpr_all as dst, src;                    // src - image pixel data, dst - address to store the result

    assembly { "push_conv" dst "," src };       // Assembly format
    binary { OPC_PUSH_CONV UNUSED:bit[5] dst src UNUSED:bit[9] };   // Instruction binary pattern
    semantics
    {
        uint32 p0,p1,p2,p3, result;

        p3 = rf_gpr_read(src);

                                                // Image edge condition handling
        if ((fifo_counter>=FIFO_DEPTH) && (fifo_counter%(FIFO_DEPTH - 1) != 0))
        {
            uint32 addr_dst;
            p0 = fifo[fifo_counter % FIFO_DEPTH];
            p1 = fifo[(fifo_counter - FIFO_DEPTH + 1)%FIFO_DEPTH];
            p2 = fifo[(fifo_counter - 1)%FIFO_DEPTH];

            result = p3 * weights[3] + p2 * weights[2]          // ALU*
                   + p1 * weights[1] + p0 * weights[0];

                                                // Storage and destination address increment
            addr_dst = rf_gpr_read(dst) + ARRAY_STEP * out_counter;
            store(OPC_ST, addr_dst, result);
            out_counter = (out_counter < CONV_RES_LEN - 1) ? out_counter + 1 : 0;
        }

                                                // Push current pixel data to FIFO, pointer increment
        fifo[w_index_0] = val;
        fifo_counter = (fifo_counter < IMAGE_LEN - 1) ? fifo_counter + 1 : 0;
    };
};
```

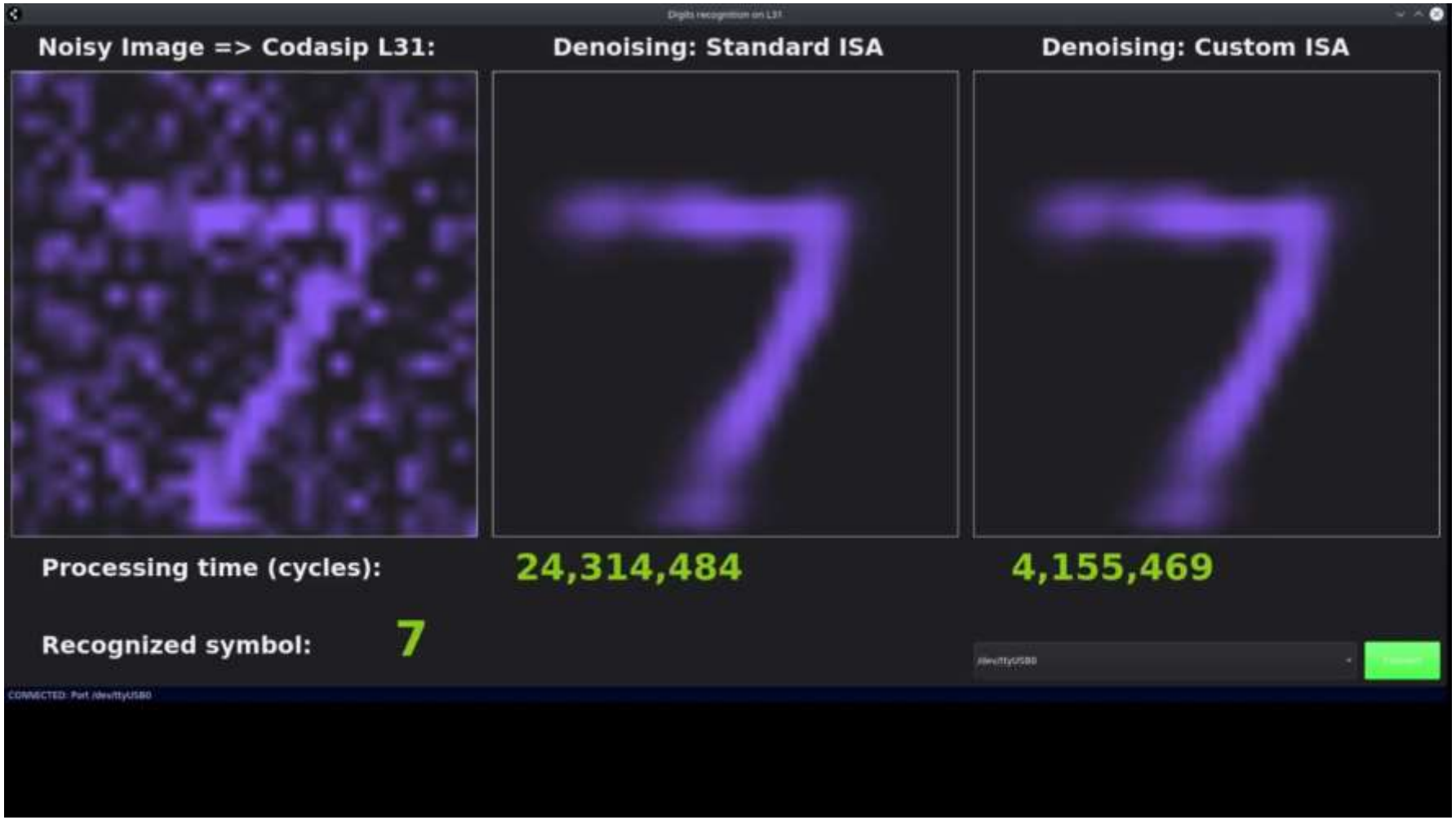
© 2023 Codasip. All rights reserved.

→ MNIST(手描き数字認識)速度比較 [標準RISC-V 対 カスタマイズRISC-V]

https://youtu.be/llKXAsFf2As

© 2023 Codasip. All rights reserved.

→ ノイズ除去 速度比較
[標準RISC-V 対 カスタマイズRISC-V]



© 2023 Codasip. All rights reserved.

→ ケーススタディ: SecureRF社 (現Veridify社)

	HWアクセラレータ無	HWアクセラレータ有	改善結果
検証時間	10.7 ms	3.8 ms	2.8x 高速化
署名の検証時間/秒（50MHz）	93	263	
コードサイズ	4,504 bytes	3,052 bytes	32% 小型化

命題は、Codasip RISC-Vプロセッサで実行されている
WalnutDSA（デジタル署名アルゴリズム）の高速化

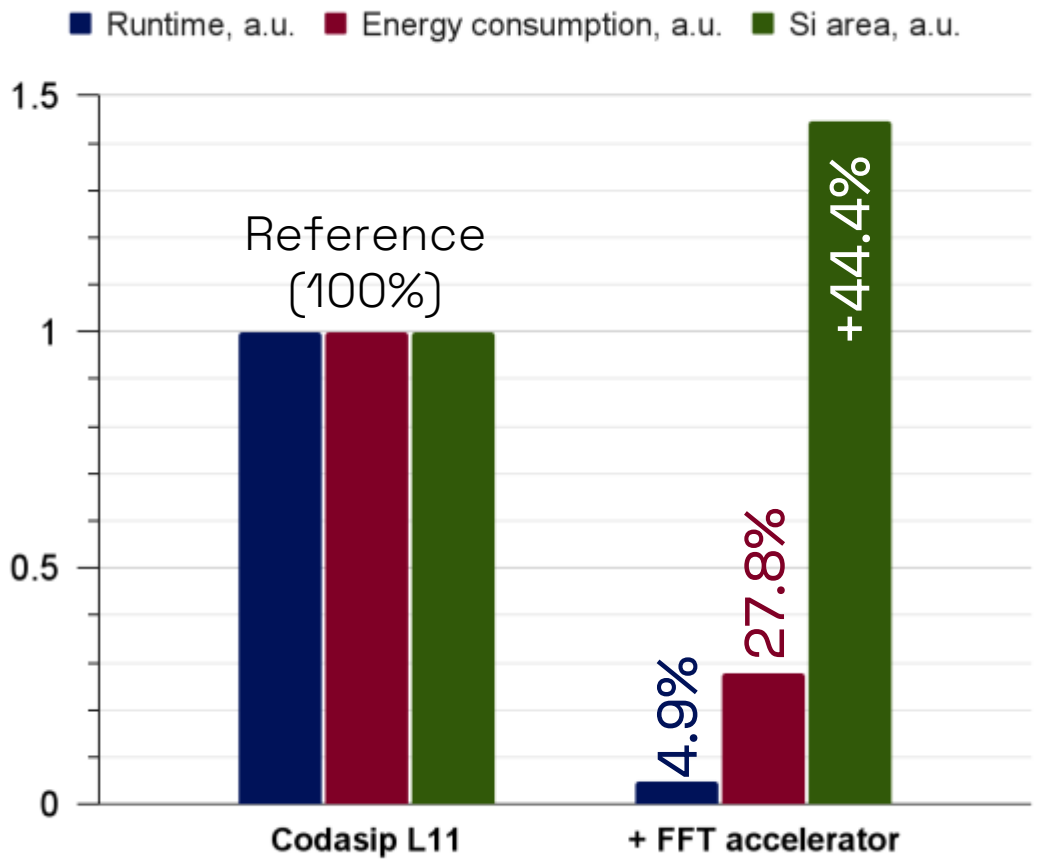
結果は、わずか2%の追加HWリソースで、実行時間は3倍改善

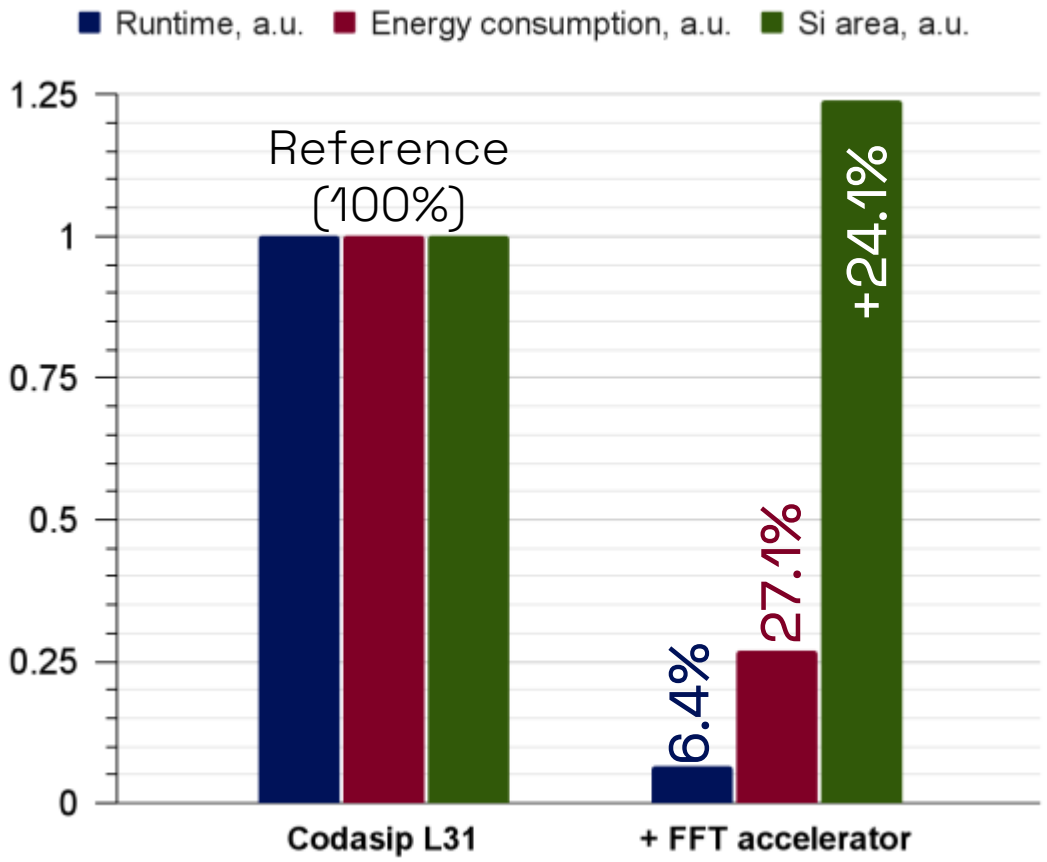
わずかな努力で、大きな改善が可能

© 2023 Codasip. All rights reserved.

→ ケーススタディ: FFTアクセラレータ

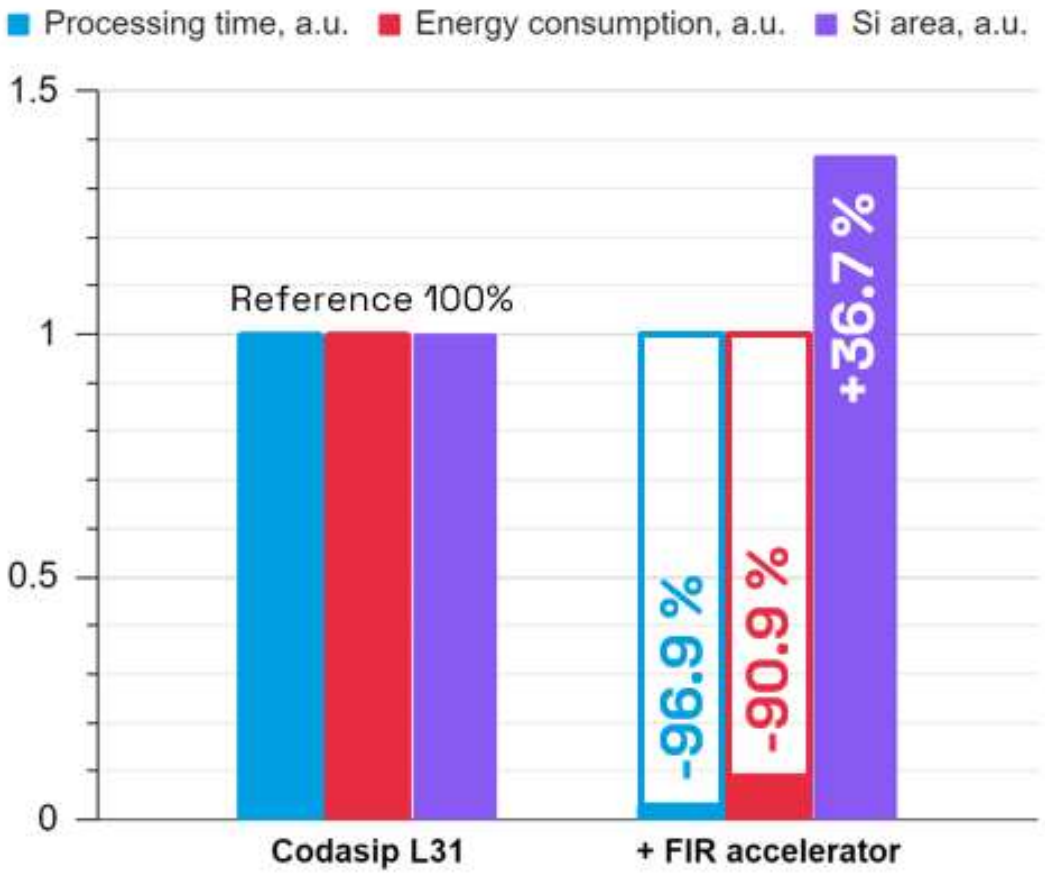
Codasip

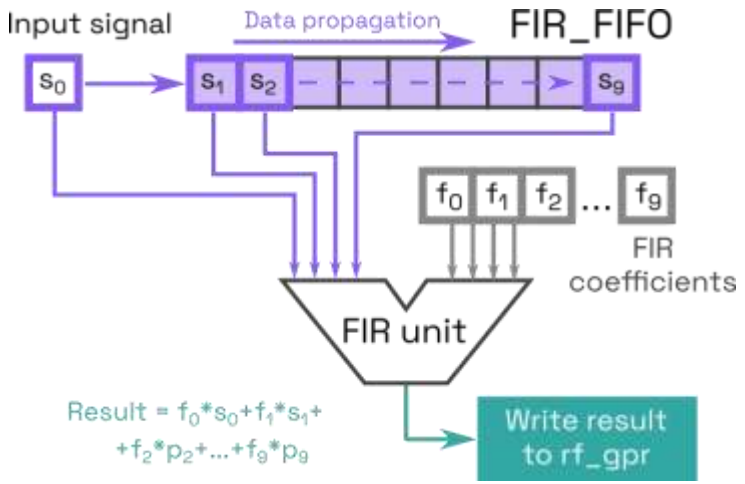




© 2023 Codasip. All rights reserved.

→ FIRフィルタ アクセラレータ



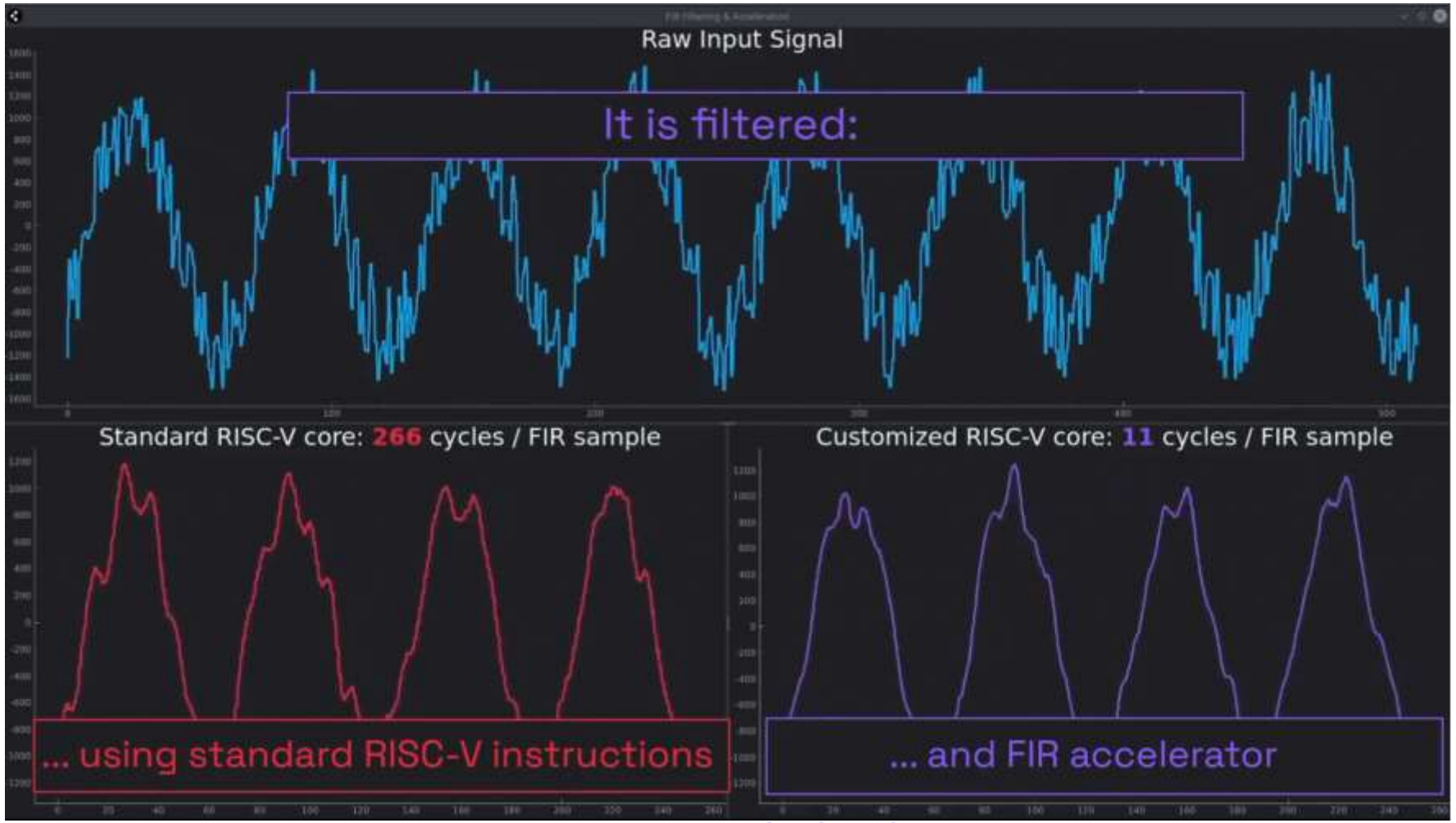


- 実行時間 96% 削減
- 消費電力 90% 削減
- 面積 +36% 増加
- 最高周波数に影響無

https://youtu.be/ODGcx9q_5sw

© 2023 Codasip. All rights reserved.

→ FIRフィルター アクセラレータ



https://youtu.be/ODGcx9q_5sw

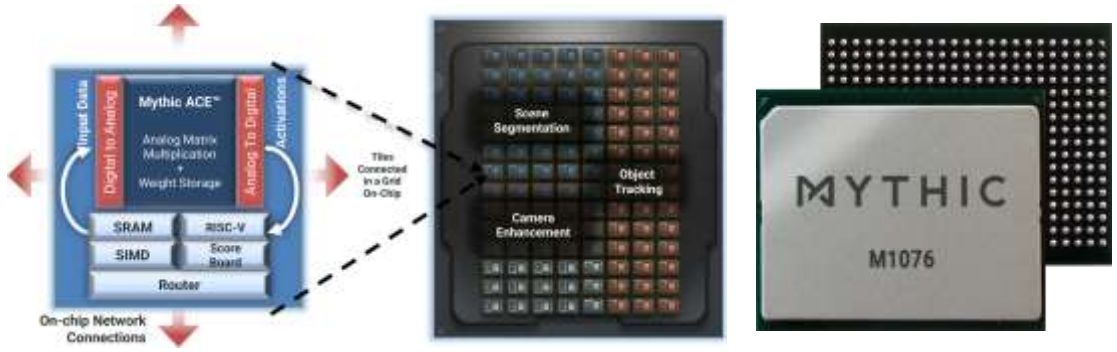
© 2023 Codasip. All rights reserved.

→ ケーススタディ: Mythic社

Codasip

MYTHIC

- Mythic社はAnalog Matrix Processor (AMP) M1076に独自のコアを必要とした
- いずれの既存コアも要求を満たせず: Mythic社はCodasip L30をベースにカスタマイズを実施
  - DSP、ビット操作、ゼロオーバーヘッドループ、
    コプロセッサ制御命令、カスタムコプロセッサインタフェース
- 従来設計より大幅な小型化・高性能化を実現
- 従来比、半分以下の時間で設計



"コダシップのソリューションは、我々のニーズに特化した、真にユニークなRISC-Vプロセッサを
柔軟に開発することを可能としました。これにより、自社でプロセッサを一から作る手間が省け、
製品開発における他の重要な分野に集中することができました。"

Ty Garibay, VP of Hardware Engineering at Mythic

→ 25TOPS / 3W

Mythicの「M1076」 出所：Mythic

Mythicは以前、「M1108」と「M1076」という2つの製品を発表した。M1076は25TOPS（INT8）のエッジチップで、3Wという電力エンベロープを備えており、動画分析に向けて開発された。YOLOv5でのレイテンシは33ミリ秒だった。製品発表時に同社が米国EE Timesに語ったところによると、M1076はビデオ監視、産業用マシンビジョン、ドローン、AR／VR（拡張現実／仮想現実）アプリケーションといった領域で弾みがついているとのことだった。

Source: https://eetimes.itmedia.co.jp/ee/articles/2303/17/news129.html

© 2023 Codasip. All rights reserved.

→ 韓SiliconArts社レイトレーシング GPU CodasipのCustom RISC-Vを採用

Codasip

Press Release

Codasip delivers custom RISC-V processing to SiliconArts ray-tracing GPUs

↓

2 November, 2022

With Codasip 7-series core IP and Codasip Studio tools

Codasip, the leader in customizable RISC-V processor IP, today announced that SiliconArts has adopted application-specific Codasip 7-series RISC-V processors with Codasip Studio customization tools. SiliconArts is a leader in innovations for high-end graphics enabling immersive experiences with its photo-realistic ray tracing graphics rendering. The combination of Codasip RISC-V processor IP and SiliconArts ray tracing GPUs will empower the next generation of the most demanding augmented reality applications.

SiliconArts ray tracing solution will incorporate Codasip's RISC-V processor core IP. Codasip Studio tools will enable customers to achieve high levels of optimization for their graphics applications. Full flexibility for the SiliconArts partnership is enabled by full access to Codasip's architecture licenses.

Source: https://codasip.com/press-release/2022/11/02/codasip-delivers-custom-risc-v-processing-to-siliconarts-ray-tracing-gpus/

© 2023 Codasip. All rights reserved.

→ グローバルでの浸透

Codasip

- 半導体大手はCodasipプロセッサを大量出荷済み
  - 世界で最初の商用RISC-Vプロセッサ
  - 20億個以上のコアが出荷済
- RISC-Vエコシステムとカスタム能力を活用した差別化設計
- ワイヤレス, イメージング, オーディオ、AI...

Tier 1 for Comms

2 of Top 3 Smartphone OEM

alhua TECHNOLOGY

vidtoo 微迪兔

LUMISSIL Microsystems

VITEC

Rambus

MYTHIC

Tier 1 for Imaging

SIGMA DESIGNS

analogix Bring Vision to Life

mobileye An Intel Company

Microsemi

UBILITE Hello IoT World

TRINAMIC Now part of Maxim Integrated

LATTICE SEMICONDUCTOR

Kameleon

2014 2015 2016 2017 2018 2019 2020 2021 2022

© 2023 Codasip. All rights reserved.

Codasip

↘ セキュリティ

© 2023 Codasip. All rights reserved.

→ コダシップ、Cerberus買収でRISC-Vのセキュリティを強化

2022年11月10日、ドイツ、ミュンヘン – カスタマイズ可能なRISC-VプロセッサIPのリーダーであるコダシップは、Cerberus Security Labs社を買収したことを発表しました。同社のIoTセキュリティIPと経験豊富なチームにより、コダシップのお客様はRISC-Vプロセッサの設計にセキュアなソリューションを迅速に組み込むことができるようになります。

<途中省略>

CerberusはIoT Security Foundationの創設メンバーとして、IoTにおけるセキュリティの必要性を積極的にアピールしてきました。これまでチームは、コンシューマ、自動車、半導体市場の主要企業での勤務経験、組み込みシステムのサイバーセキュリティに関する先進的な学術研究への参加および業界団体との関わりを経て、IoTアプリケーション向けの堅牢なセキュリティ製品の創出へつながったのです。コダシップは、Cerberusの実績を基にRISC-V が今日必要としている製品を独自に提供できるようになります。つまりカスタマイズされたドメイン特化型や低電力組み込みアプリケーション用のコアにおいても、安全性とセキュリティを提供できることを保証するものです。

コダシップのプロセッサIPは、Codasip StudioとCodAL言語に最適化されており、コダシップの顧客であるRISC-V開発者が容易にセキュア機能を統合できるよう、Cerberus ラボの組み込みセキュリティIPを迅速に統合する予定です。

© 2023 Codasip. All rights reserved.

→ コダシップ、 SecuRISC5 イニシアチブを開始

Codasip

2022年12月12日 - RISC-V Summit, San Jose - プロセッサ設計自動化とRISC-VプロセッサIPのリーダーであるコダシップは、コダシップIPとサードパーティ技術を組み合わせた検証済みの参照デザインを使って、顧客に安全でセキュアなカスタムコンピュートを提供するコダシップのイニシアチブであるSecuRISC5を発表しました。コダシップが最近立ち上げたCodasip Labは、SecuRISC5が注目すべき機会を特定する上で中心的な役割を果たすとともに、全産業コラボレーションを調整するハブとしての役割も担います。

最近のCerberus Security Labs社の買収に伴い、コーダシップはRISC-VプロセッサIPのセキュリティ開発を加速しており、2023年にセキュリティリファレンスデザインを提供する予定です。真にセキュアなシステムは単独では開発できないため、コーダシップはパートナーと協力し、完全でセキュアなRISC-Vエコシステムを提供します。

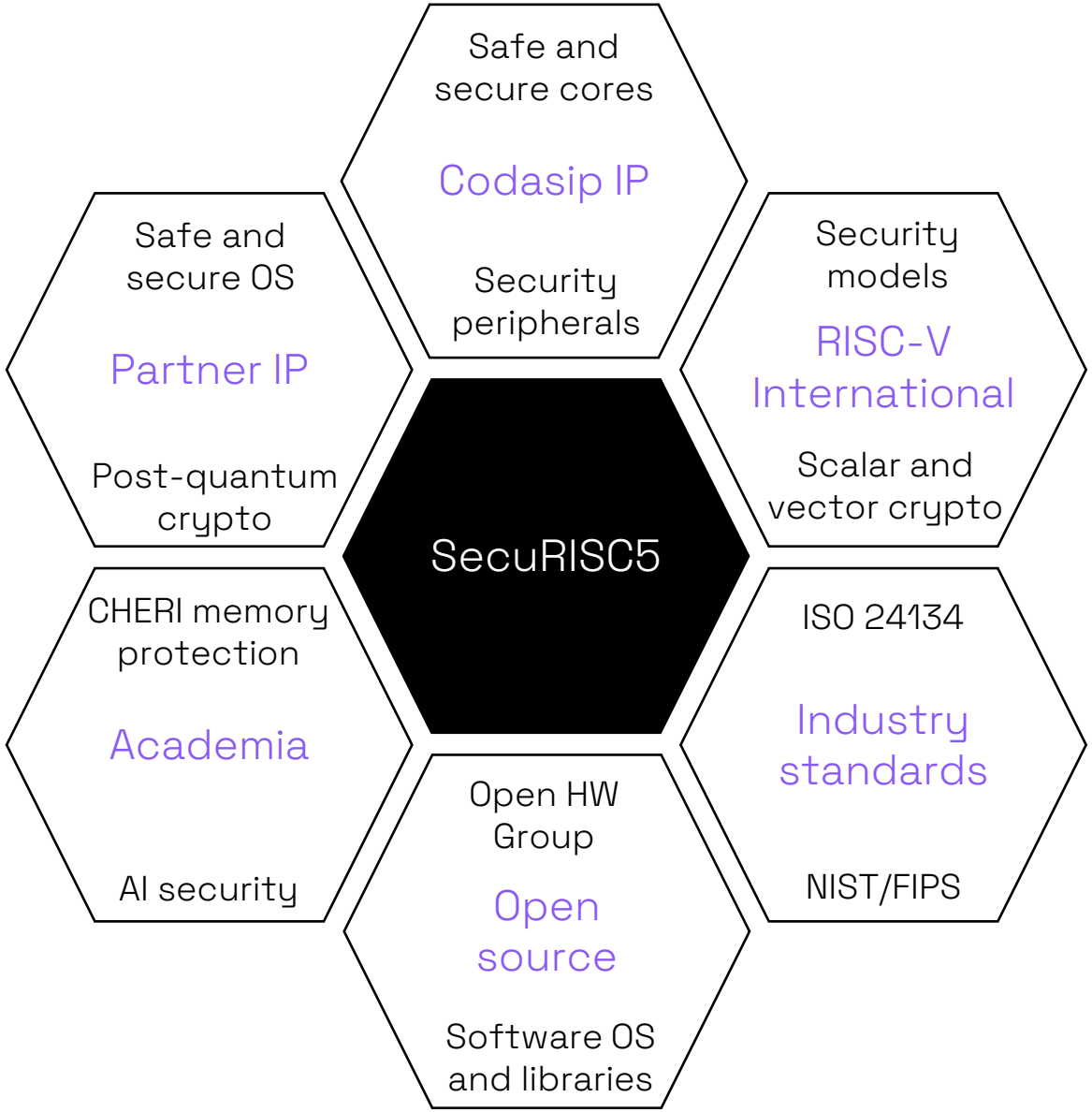
<途中省略>

RISC-V は、あらゆるタイプの安全なアプリケーションと機能、特に独自の ISA からのカスタム ソリューションや必要なサポートを取得するのに苦労しているドメイン固有の設計に対応する幅広いシステムを開発するための理想的なプラットフォームを提供します。RISC-V はセキュリティ システム1にますます実装されており、Codasip は RISC-V 標準プロセスで積極的に活動しています。SecuRISC5 イニシアチブは、RISC-V 国際ワーキング グループの作業に基づいて構築され、承認された新しい標準を実装します。

© 2023 Codasip. All rights reserved.

→ Security ecosystem

Codasip



© 2023 Codasip. All rights reserved.

→ これらキーワードが気になる方は、後ほど

Codasip

Fault protection

Multi-stage boot

Power trace prediction

RoT (Root of Trust)

ECC

Secure ledger

PUF

Memory Tagging

Side-channel protection

Fault emulation

Secure access

SHA

Random Number Generator

Random Number Generator

CHERI (Capability Hardware Enhanced RISC Instructions)

Device attestation

TEE (Trusted Execution Environment)

Secure Boot

EMP protection

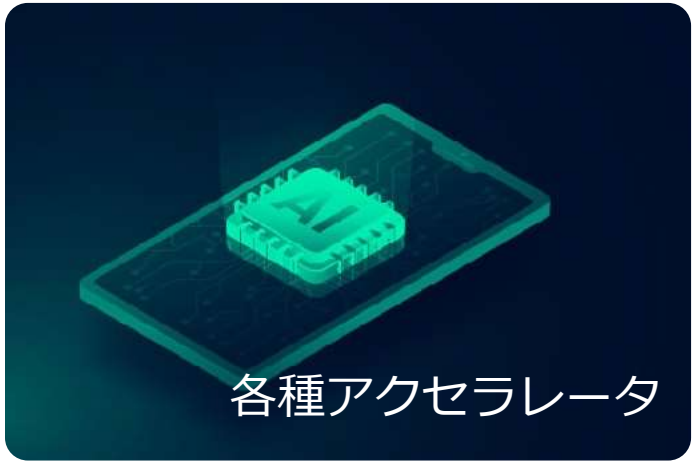
0x5 HEX-Five Security

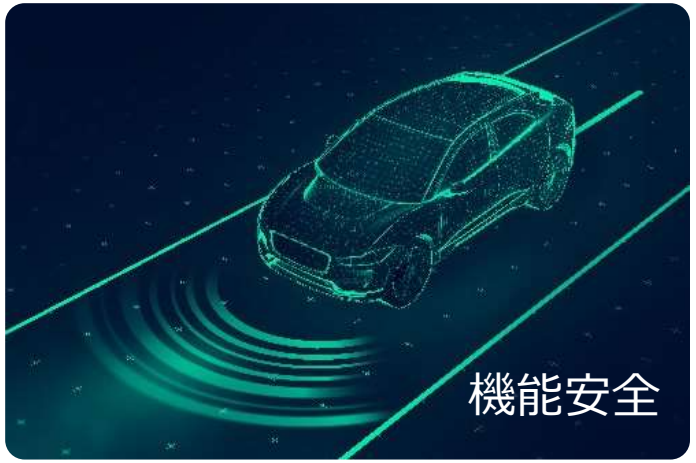
Keystone

opentitan

© 2023 Codasip. All rights reserved.

→ Codasip Lab - 注力イノベーション分野





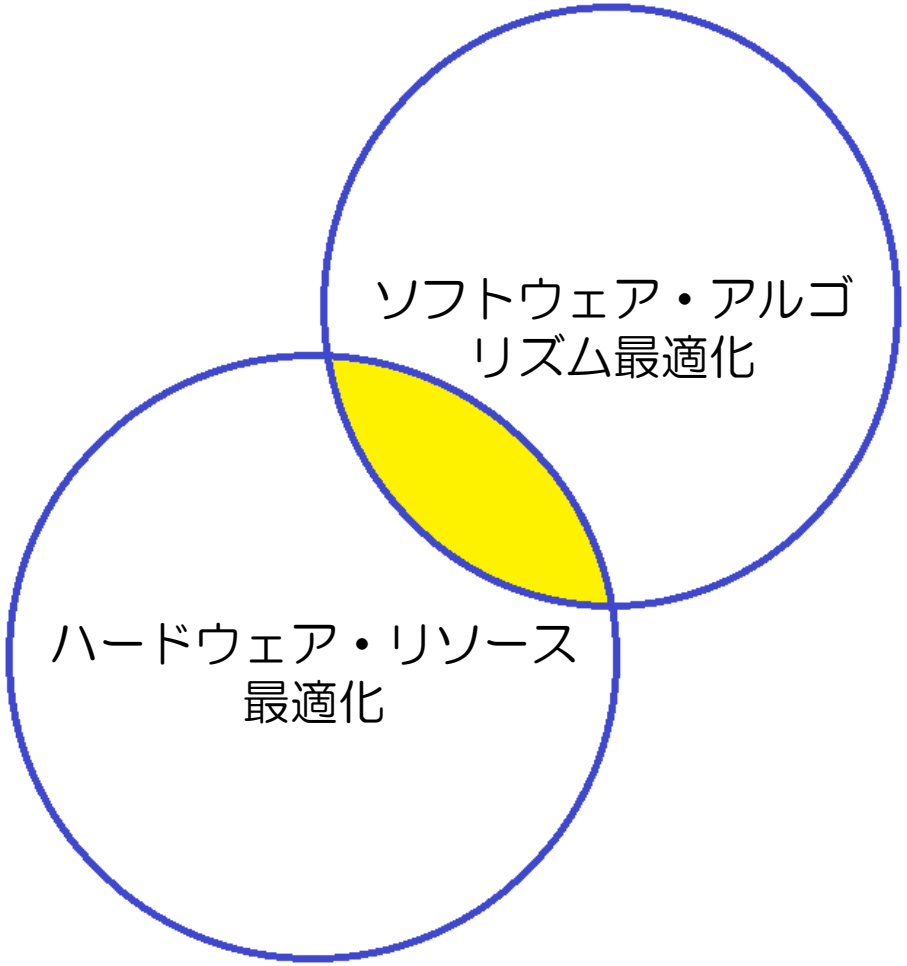
© 2023 Codasip. All rights reserved.

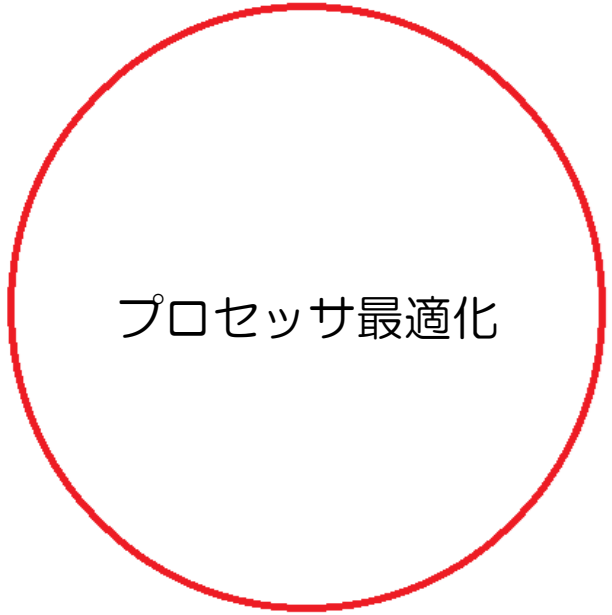
→ 従来の研究における壁

Codasip

- 従来の研究では、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





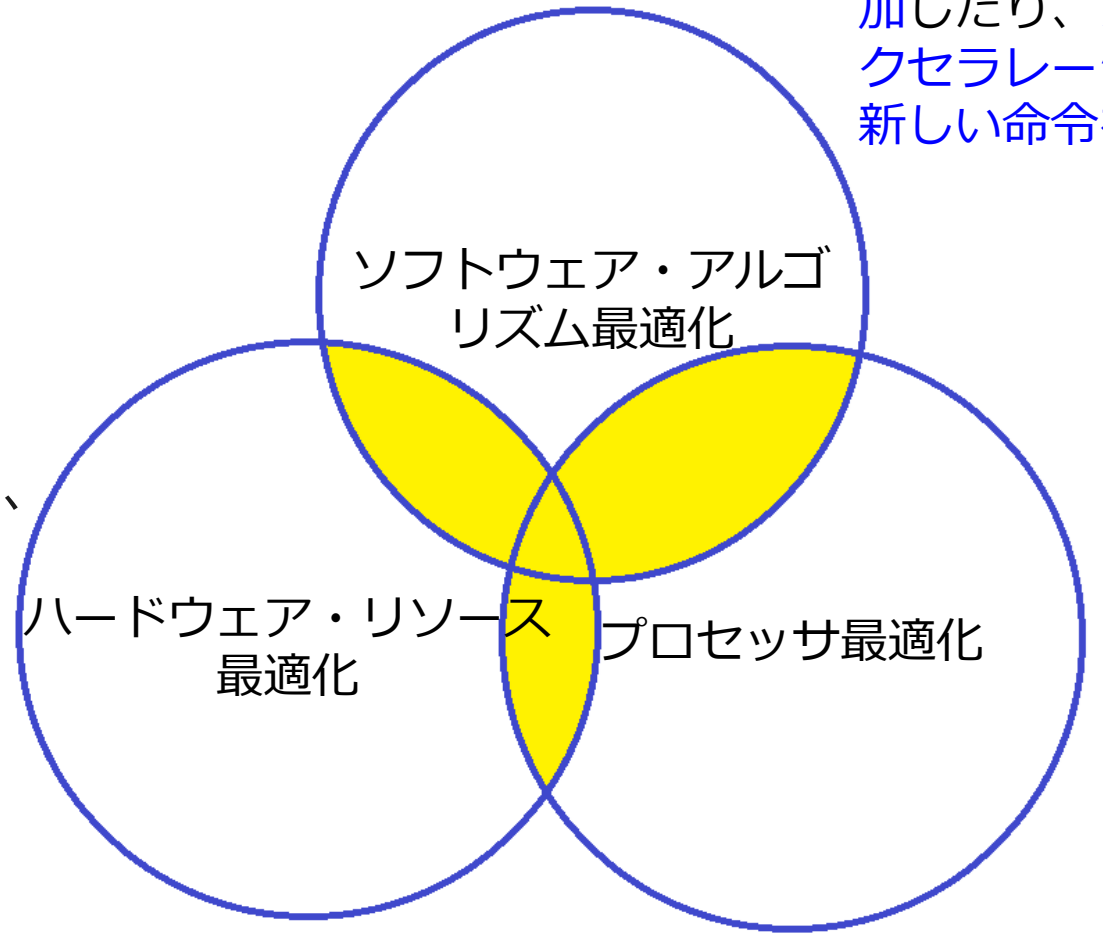
© 2023 Codasip. All rights reserved.

→ 従来の研究における壁を突破する

例: CPUレジスタファイルへのアクセス全てに、2ビットエラー検出と1ビットエラー訂正のためのハミングエンコーディングを追加したり、アプリケーションアクセラレーションのための新しい命令を追加したりする

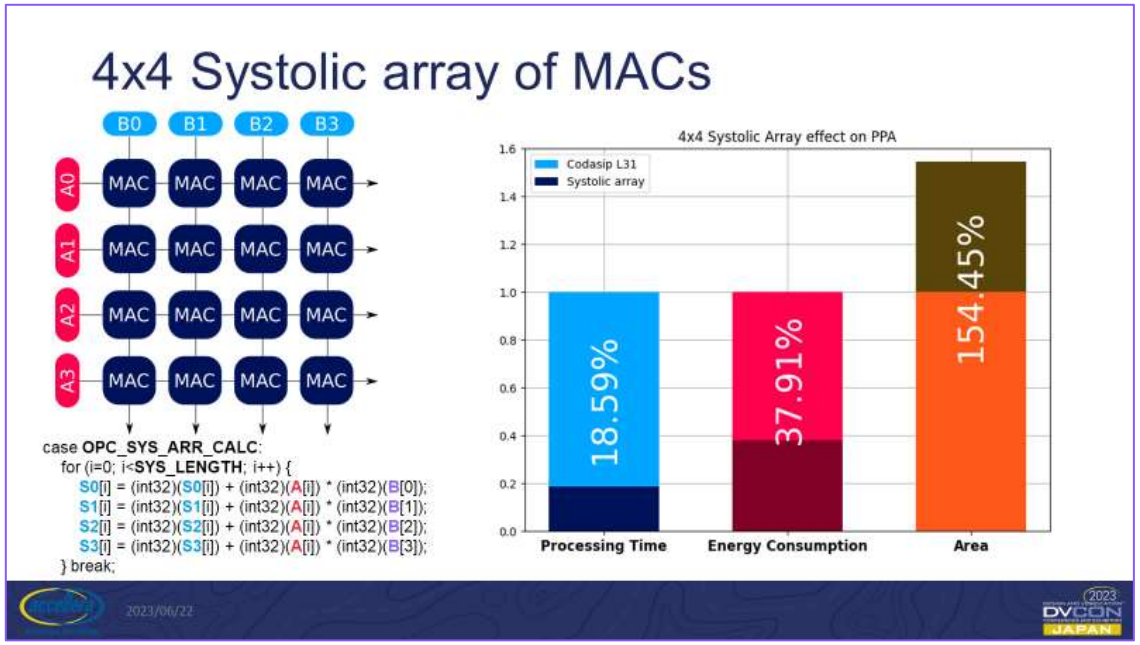
- RISC-V IPとCodasip Studioの組み合わせで、研究の壁を突破する

- 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です



© 2023 Codasip. All rights reserved.

→ その他の事例紹介



- DV Con 2023 Japan にて
  - 6月22日(木) 12:30 – 13:00
  - 川崎市産業振興会館 Room1

- 当社 伊藤が
  テクニカルセッション
  「RISC-V Custom Compute」セッションにて「4x4 Systolic array」の事例をご紹介

© 2023 Codasip. All rights reserved.

Codasip

→

Thank you!

本日 ホワイエ A
にて出展中

是非、
お立ち寄りください

contact_japan@codasip.com

© 2023 Codasip. All rights reserved.