コダシップのカスタムRISC-Vソリューション

RISC-V Days Tokyo 2022 Spring #day1_10_codasip

2022/5/31 | Tokyo, Japan

コダシップ , ジャパン カントリーマネージャー
明石 貴昭

コダシップ グループ

プロセッサ開発を
よりシンプルに
より速く
より安く

コダシップとは？

- RISC-VプロセッサIPとプロセッサ設計自動化ツールのリーディング プロバイダー
- 2014年にチェコ共和国（EU）ブルノで設立
  - 10年にわたるブルノ大学でのプロセッサ設計自動化の研究成果をもとに起業
  - RISC-V International創設メンバー（以前の RISC-V Foundation), www.riscv.org
  - 2015年11月に世界初のライセンサブルRISC-VプロセッサIPを発表
  - 20億個以上のCodasip RISC-V プロセッサが出荷済み
- 本社、R&Dセンターは、 EU圏
  - 約120名の社員
  - 世界に営業拠点有

マネージメントチーム

- 約120名の社員 （昨年度比+33%）
  - R&D 100名
  - 商品化・カスタマーサポート、財務、人事等 20名
  - 2022年度末で>250名の計画
- 社内の専門知識を生かした強力なポートフォリオ:
  - マイクロアーキテクチャ、RTL・論理設計、高位合成、コンパイラ、ソフトウェア開発ツール、厳密な検証
- ヨーロッパ企業:
  - ドイツ本社
  - チェコ共和国ブルノに技術本部と研究開発センター
  - ミュンヘン(独)、ソフィア・アンティポリス(仏)、ブリストル(英)、ケンブリッジ(英)にデザインセンター
- RISC-V Internationalの創設メンバー
  技術運営委員に選出

Karel Masařík
President & Founder

Over 20 years in processor development and design tools. Founded Codasip based on his PhD thesis and currently leading Codasip's innovation initiatives.

Zdeněk Přikryl
Chief Technology Officer

Over 15 years of experience in processors design from small MCUs to complex DSPs/VLIWs, embedded systems design, HLS, simulation, driving R&D.

Ron Black
Chief Executive Officer

Over 25 years running software and semiconductor businesses in Europe and the US, including microprocessor and IP licensing businesses. He specializes in corporate transformations and has a history of significant exits.

Simon Bewick
Design Centre Director
United Kingdom

Over 30 years experience designing and delivering SoC devices from architecture to silicon volume production, managing development teams across multiple geographies.

Mélaine Facon
Design Centre Director
France

Over 20 years of experience in the semiconductor IP industry, from HDL design and engineering to program and project management.

Vladimír Koutný
Chief Financial Officer

Over 15 years of experience in high-growth technology companies. All-round financial and operational expertise.

Rupert Baines
Chief Marketing Officer

Over 30 years of global experience with customers and markets. Lived in USA, Germany, and Spain, and worked for a number of deep-tech start-ups as a CEO.

Kateřina Smrčková
Chief People Officer

Over 10 years of HR experience in international companies, both big and small. All-round HR expertise. Personal motto: *Making Codasip a happy place to work and grow.*

Brett Cline
Chief Revenue Officer

Over 25 years of leadership experience in sales, marketing, and engineering. Proven methodology for growing sales. Involved with seven company mergers/acquisitions.

コダシップのソリューション

codasip
RISC-V PROCESSORS

選択:

アプリケーションに適した
RISC-Vプロセッサ コアを

- 低消費電力、高性能な組み込み型
- アプリケーション・プロセッサ
- RISC-V 準拠
- コンフィギュラブル

カスタム:

Codasip RISC-Vプロセッサを
ベースとしてCodasip Studioで

- カスタム命令を追加して
  アルゴリズムを高速化
- 既存のRISC-Vプロセッサを
  カスタマイズのベースとして使用

codasip
STUDIO

作成:

高レベル記述を用い
プロセッサをゼロから

- 性能、面積、消費電力を考慮した
  設計のファインチューニング
- ソフトウェアとハードウェアの開発
  キットを自動生成

コダシップを選んだお客様

codasip
RISC-V PROCESSORS

TRINAMIC
Now part of Maxim Integrated

LATTICE SEMICONDUCTOR

vidtoo 微迪兔

VITEC

Kameleon

MYTHIC

DONGWOON
(주)동운아나텍

analogix
Bring Vision to Life

Microsemi

UBILITE

NeuLinker

alhua
TECHNOLOGY

codasip
STUDIO

MOBILEYE

SIGMA DESIGNS

Rambus

LUMISSIL Microsystems
A Division of ISSI

コダシップを適用した分野

イメージ/ビデオDSP

MOBILEYE
VITEC
analogix
Bring Vision to Life
DONGWOON
(주)동운아나텍

オーディオDSP

Microsemi

AI/ML拡張

MYTHIC
vidtoo 微迪兔

セキュリティ拡張

Rambus

IoT

SIGMA DESIGNS
UBILITE

MCUs

TRINAMIC
Now part of Maxim Integrated
alhua TECHNOLOGY

codasip

RISC-V PROCESSORS

拡張可能なコア・ファミリー

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)

低消費電力 エンベデッド向け

- 最小で省電力
  - レガシーな独自コアに代わる魅力的な選択肢
- 幅広い32ビットプロセッサ群
  - 浮動小数点演算ユニット使用可
  - マシンおよびユーザーの特権モード
  - 最大128の外部割込み
- 1シーリーズから5シリーズがベース
  - 最小の16レジスタ組み込みコントローラから
  - 動的分岐予測搭載の
    バランス型5段プロセッサまで

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**

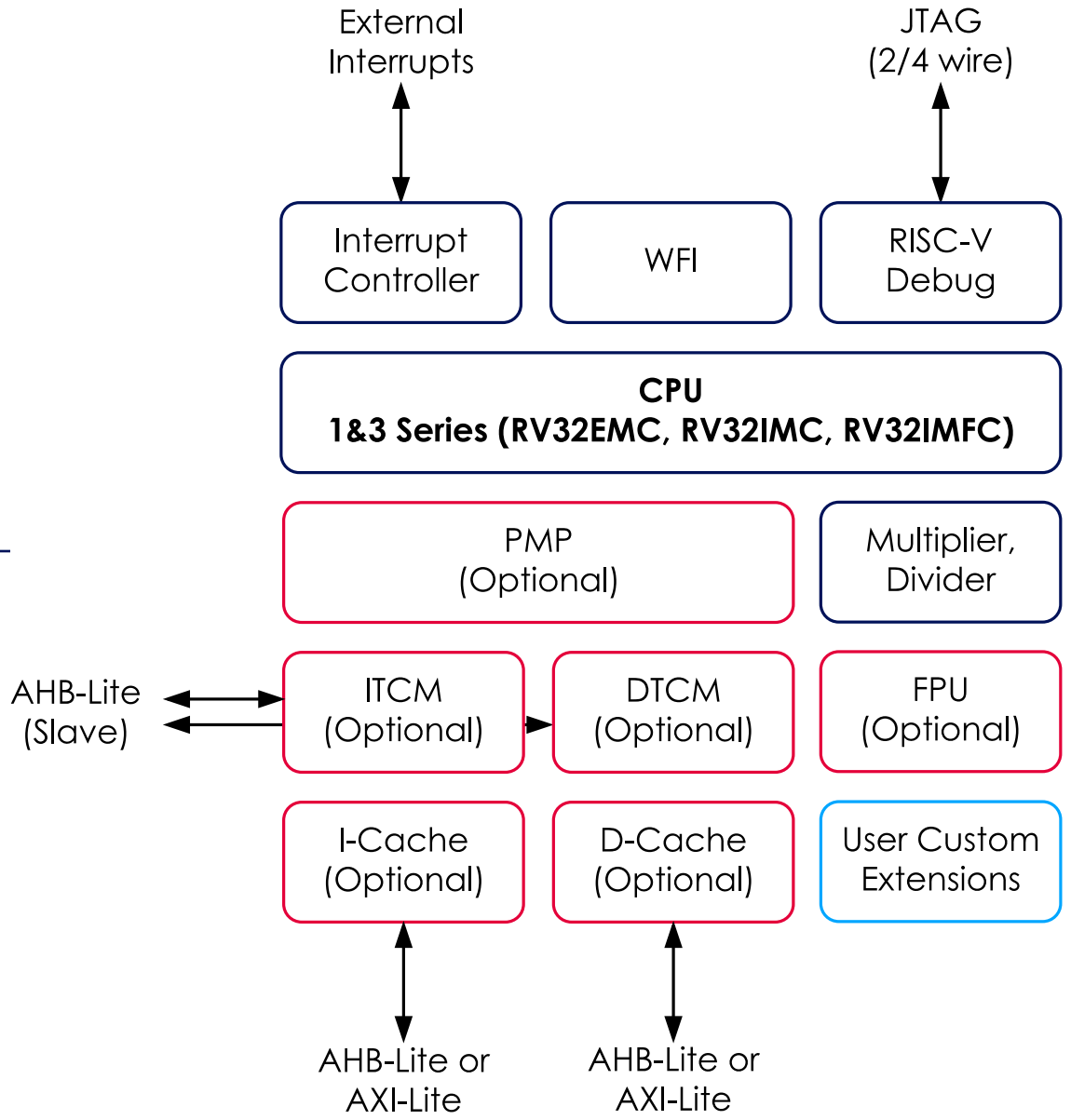
3 SERIES

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**



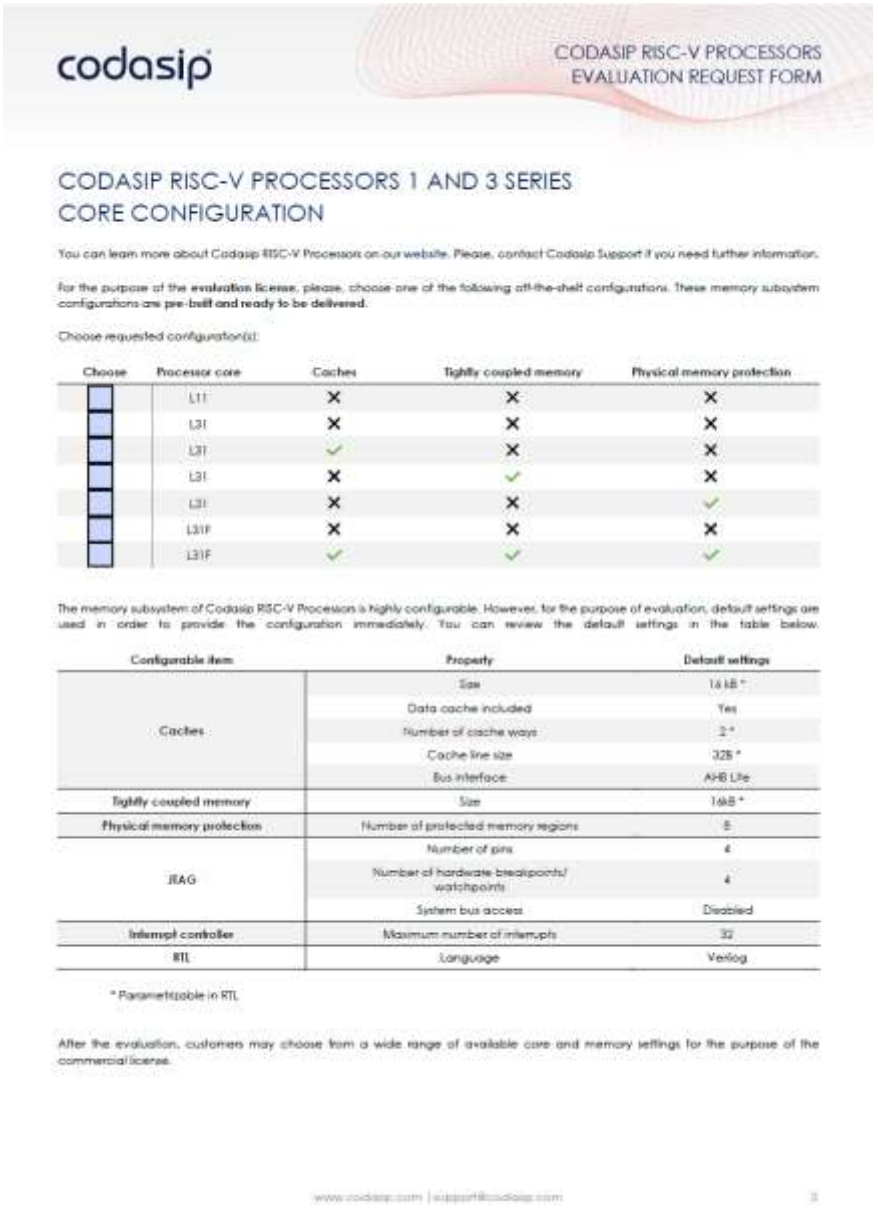
L11/L31/L31Fの主な特徴

インストラクションセット	RV32IMCB (L11はRV32EMC)
特権モード	Machine User (L31/L31Fのみ)
ローカル割込コントロール	RISC-V CLINT （最大128割込）
キャッシュ	L31/L31Fでカスタマイズ可能
密接合メモリ	L31/L31Fでカスタマイズ可能
JTAG & RISC-V デバッグ	カスタマイズ可能

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 settings
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

高性能エンベデッド向け

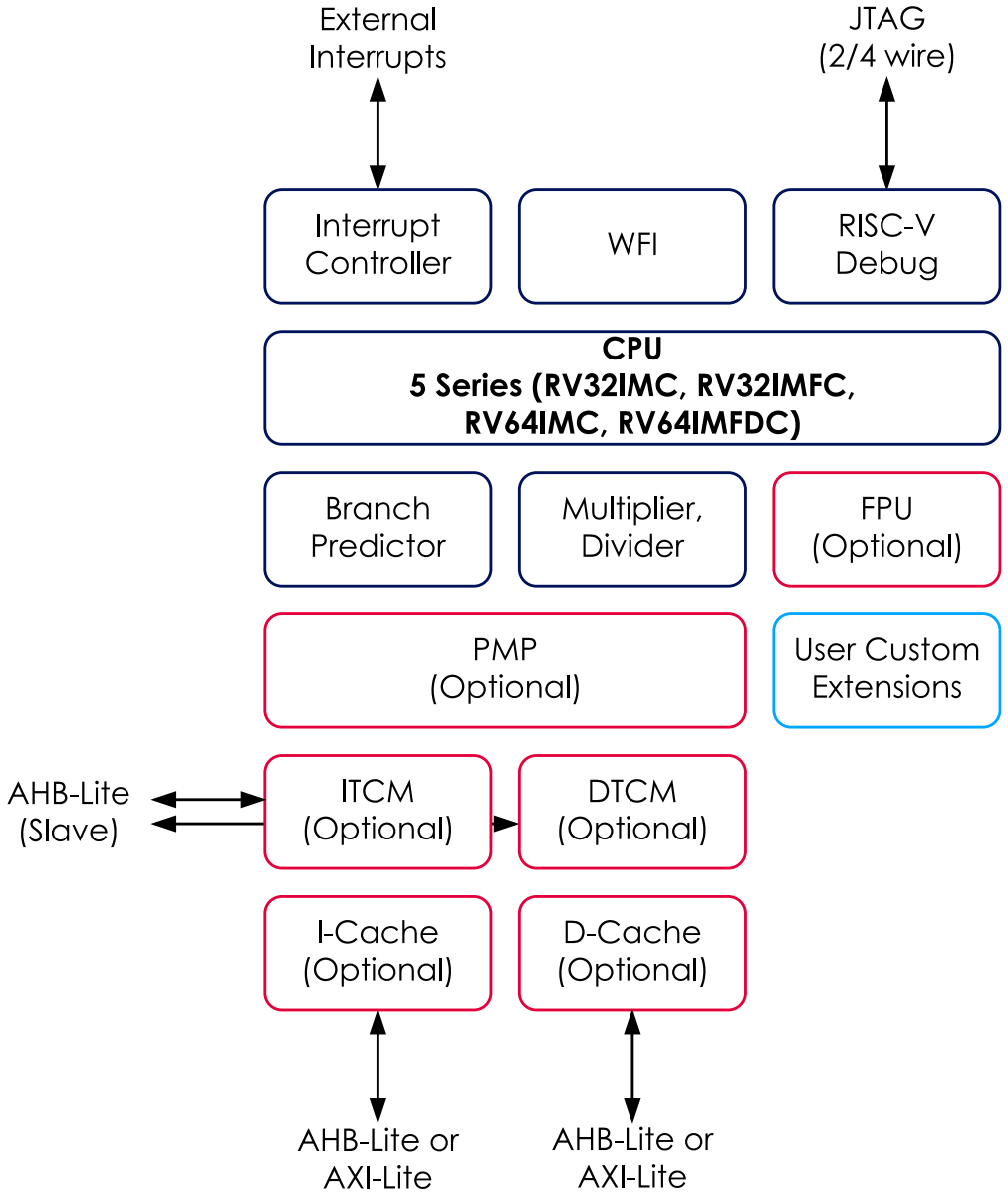
- 高周波数・高スループット設計
  - ストレージやネットワークなど、最新のデータインテンシブ アプリケーションに最適
- 64ビット プロセッサを追加し、組込み向けプロセッサ ラインナップを拡充
  - 浮動小数点演算ユニット使用可能
  - マシン権限モードとユーザー権限モード
  - 最大256の外部割込み
- 5シリーズがベース

5 SERIES

バランスの良い**5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**



アプリケーション向け

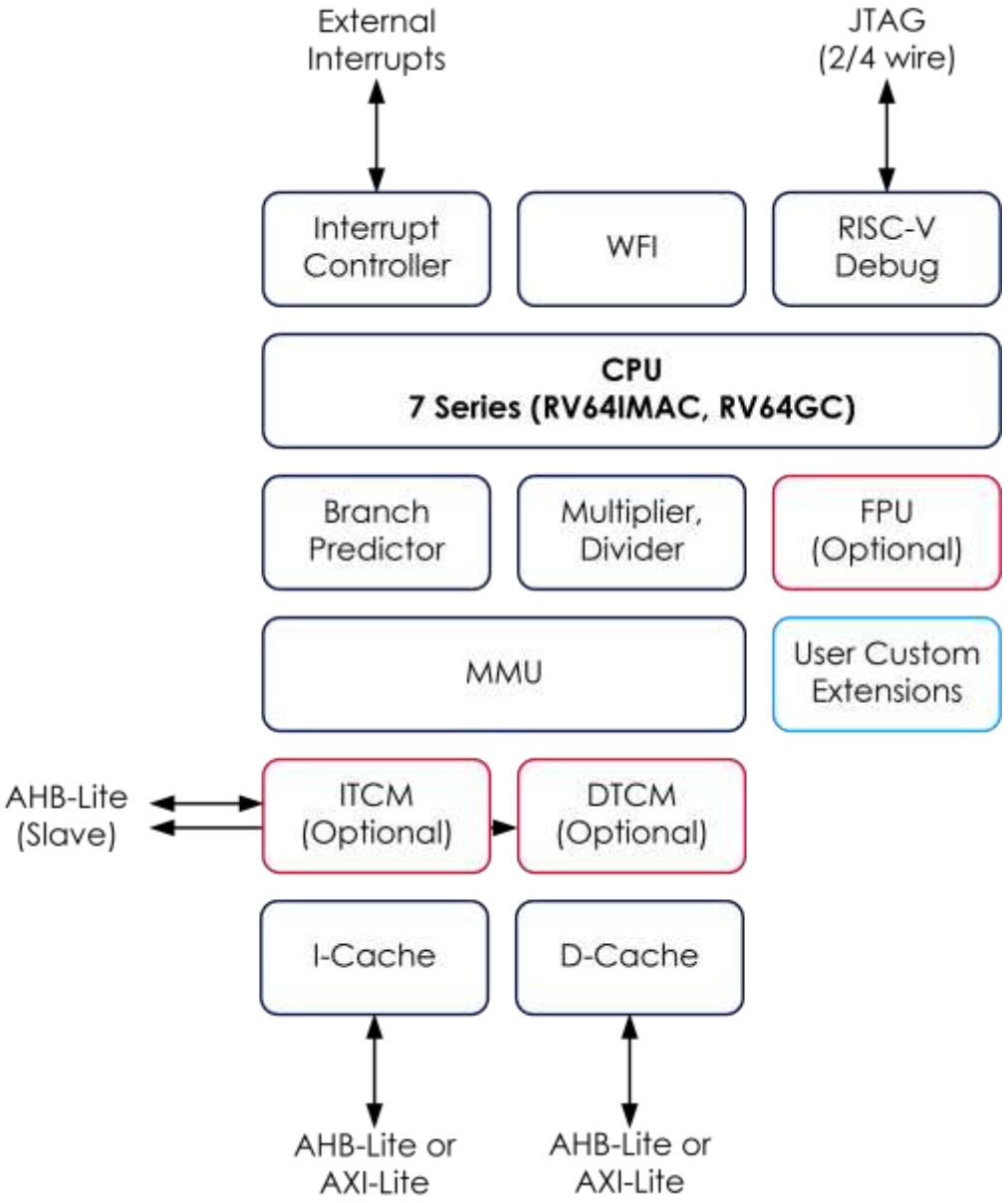
- Linuxが動作するように設計
  - 高度なユーザーインターフェース、高度なI/O、ネットワークが必要なアプリケーションに最適
- リッチなオペレーティングシステムに対応すべく、必要な機能を備えた64ビットプロセッサ
  - メモリ管理ユニット
  - スーパーバイザー権限モード
  - アトミック命令セット
- 7シリーズがベース

7 SERIES

複雑な**7段パイプライン**

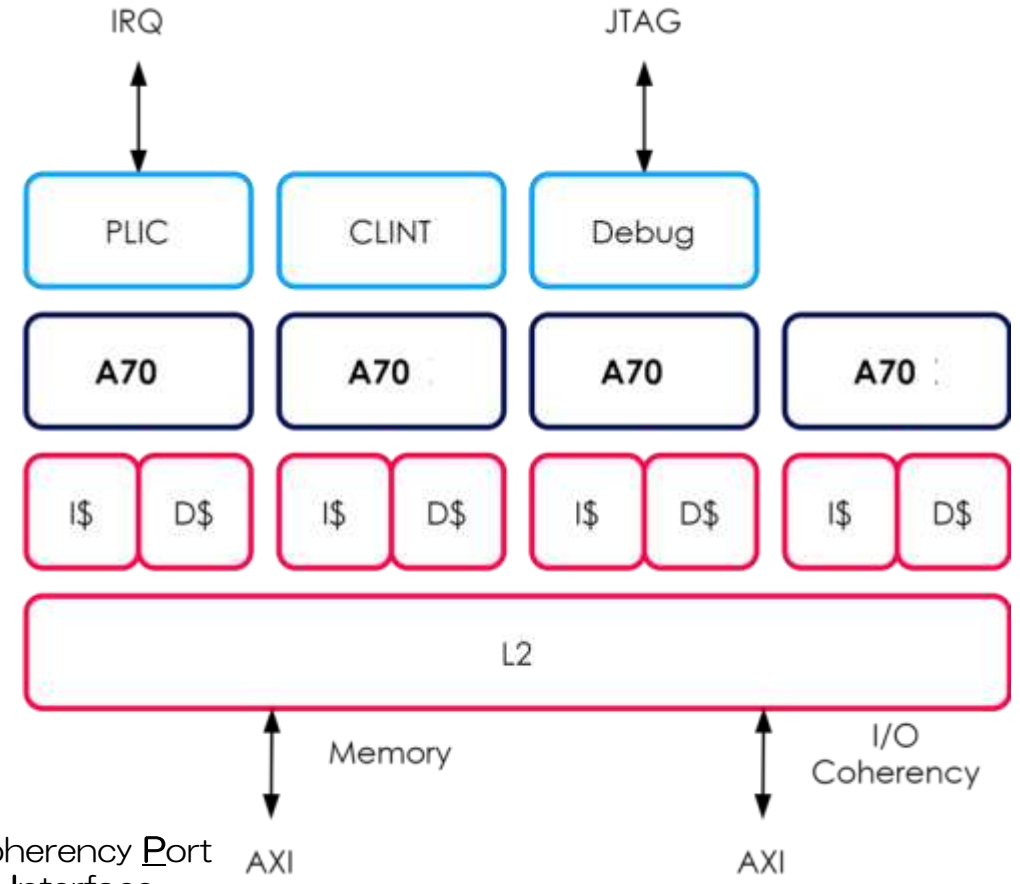
圧縮命令で最小のコードサイズを実現

動的な**分岐予測**



マルチコアによる圧倒的なパフォーマンス

- 最大4つのA70またはA70Pプロセッサをクラスタに搭載
- コヒーレントL1/L2キャッシュを搭載したスケーラブルなマイクロアーキテクチャ
  - キャッシュはコンフィギュラブル
  - ACPポート付きAXIインターフェース（I/Oコヒーレンシ）の提供を開始
  - CHIプロトコル（フルシステムコヒーレンシー）は近日中に利用可能
- RISC-V標準の割り込みコントローラ
  - CLINT (コア レベル)
  - PLIC (プラットフォーム レベル)
- A70-MP、A70P-MPがマルチコア対応



* ACP: Accelerator Coherency Port
* CHI: Coherence Hub Interface
* CLINT: Coreplex Local INTerrupts
* PLIC: Platform-Level Interrupt Controller

アプリケーションに最適なコアを選択

ALL CORES Standard RISC-V debug JTAG (4pin/2pin) Compressed instructions AMBA buses	LOW POWER EMBEDDED 32-bit Up to 128 interrupts	HIGH PERFORMANCE EMBEDDED 64-bit Up to 256 interrupts	APPLICATION 64-bit FPU Linux support
7 SERIES 7-stage pipeline IMC instruction set 32 registers Branch predictor Parallel multiplier			Codasip A70 Codasip A70-MP Codasip A70P Codasip A70P-MP
5 SERIES 5-stage pipeline IMC instruction set 32 registers Branch predictor Parallel multiplier	Codasip L50 Codasip L50F	Codasip H50 Codasip H50F	
3 SERIES 3-stage pipeline IMC instruction set 32 registers Parallel multiplier	Codasip L31 Codasip L31F Codasip L30* Codasip L30F*		
1 SERIES 3-stage pipeline EMC instruction set 16 registers Sequential multiplier	Codasip L11 Codasip L10*		

* Not recommended for new designs
F = Floating Point Unit, P = RISC-V P Packed SIMD Extension, MP = Multiprocessing

+

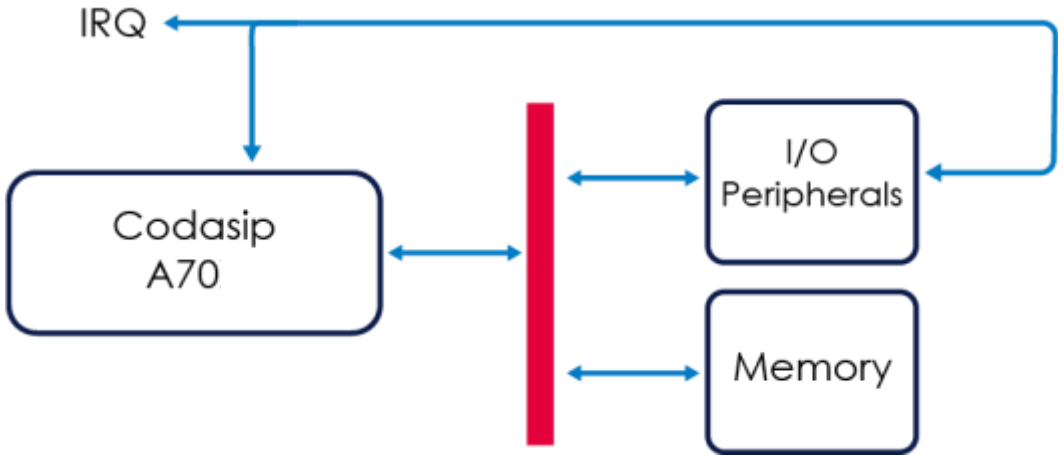
キャッシュ
命令とデータ
サイズのカスタマイズ
キャッシュウェイ数
キャッシュラインのサイズ

密結合メモリ
TCM: Tightly Coupled Memories

PMP: Physical Memory Protection

Codasip FPGA評価プラットフォーム

- 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することが可能
  - 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 Nexys-A7-100T FPGAボード
  - Digilent JTAG-HS2プログラミングケーブル
    - SeggerやIARのケーブルを使用することも可能
- 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なクイック・スタート・ガイドを提供



codasip

RISC-V PROCESSORS CUSTOMIZATION

競合に差をつけたくありませんか？

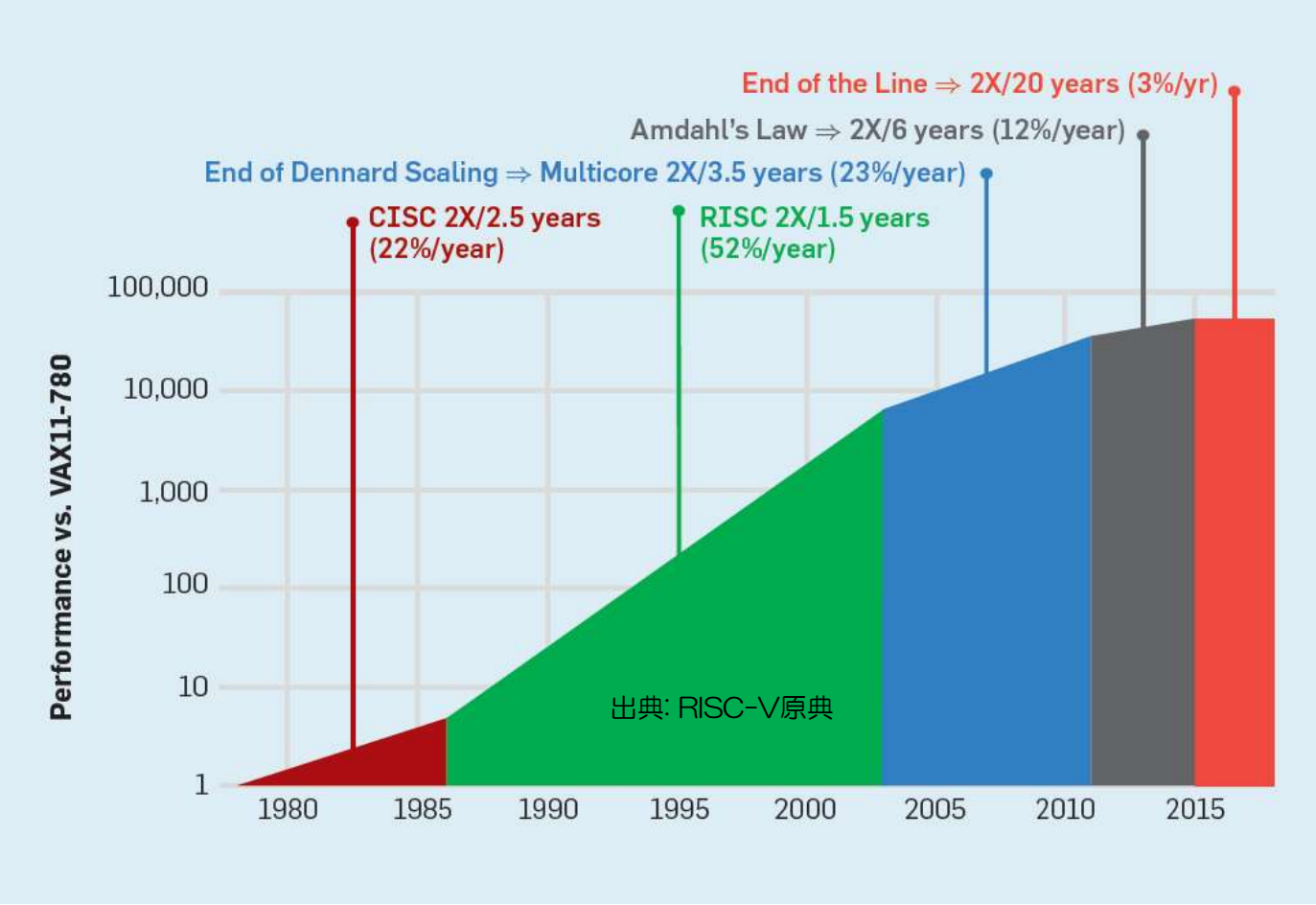
#DesignForDifferentiation

お探しのプロセッサが見つかりませんか？

- RISC-Vのオープン命令セットアーキテクチャは、カスタマイズを前提に設計されています
  - RISC-Vエコシステムの利点を維持したまま、製品を差別化する
  - ISAを最適化し、より良いPPAを実現する
    - 1タスクあたりの命令数の削減
      → 性能アップ
    - 総サイクル数の削減
      → 低消費電力化（低周波数化）
    - コードおよびデータサイズの削減
      → 小面積化（ロジックおよびメモリ）

- 既存のCodasip RISC-Vプロセッサをカスタマイズのベースラインとして使用しましょう
- RISC-Vプロセッサを自分でカスタマイズできる!
- プロセッサのカスタマイズを自動化する方法については、Codasip Studioの概要をご確認ください
- ご希望のカスタマイズがコダシップで実現可能かどうか、是非ご相談ください!

スケーリングの崩壊と次の選択肢



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

- 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
- RV64MACX[ext]
  – MAC + 非標準ユーザー拡張

準備されている「非標準ユーザー拡張」

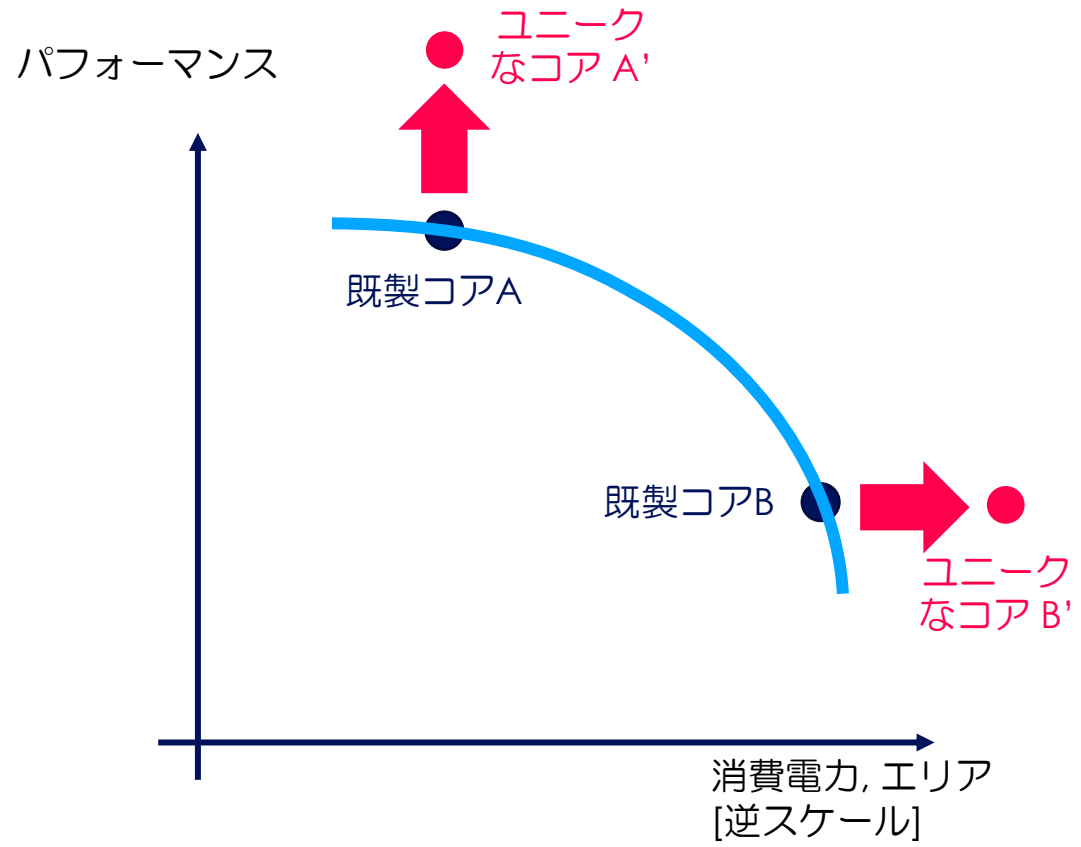
31 26	25 15	14 12	11 7	6 0	
funct6	*custom*	funct3	*custom*	opcode	Recommended Purpose
6	11	3	5	7	
100011	*custom*	0	*custom*	SYSTEM	Unprivileged or User-Level
110011	*custom*	0	*custom*	SYSTEM	Unprivileged or User-Level
100111	*custom*	0	*custom*	SYSTEM	Supervisor-Level
110111	*custom*	0	*custom*	SYSTEM	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 v. 20211203
https://github.com/riscv/riscv-isa-manual/releases/download/Priv-v1.12/riscv-privileged-20211203.pdf

既製コアは全ての要望にジャストフィットできない

- **既製コアのPPAは1つ**
  - 典型的ボリューム要件に寄せた設計
  - 一方お客様のアプリケーションは千差万別
  - 既製コアを選択するということは、妥協することを意味します
- **あなたの要件に適したPPAを得る**
  - Codasip Studioによるカスタマイズで、お客様のアプリケーションに最適なPPAを実現します



何をカスタマイズできるか？

- 命令セットアークテクチャレベルでは:
  - スカラ命令
  - SIMD/ベクター命令
  - 単精度/倍精度 浮動小数点演算 命令
  - 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
    - ロード/ストア x プリ/ポストインクリメント
  - 複数レジスタファイル
- マイクロアーキテクチャレベルでは:
  - パイプライン段数の定義
    - ステージ難読化
  - メモリサブシステムへの接続定義
    - プロトコル
    - Bit幅
  - データ/制御/構造 ハザードのハンドリング
  - リソースシェアリング
  - MMU
  - 物理メモリプロテクション

その他いろいろ...

なぜ自動化なのでしょう？

カスタマイズされたプロセッサには、カスタマイズされたSDKが必要...

標準的なカスタマイズ（カスタムISA拡張を手動で追加）：

1. 新しい命令をモデル化し、シミュレーション
2. コンパイラの修正
3. アセンブラを修正
4. デバッガでの対応追加
5. 検証, 検証, 検証…

>>挑戦的、時間がかかる、費用がかかる…

カスタマイズツール自動生成の**利点**:

- ツールの改造に要する時間の短縮
- カスタムプロセッサの開発コスト削減
- カスタマイズ プロセッサは、標準的C/C++を使用して普通にソフト開発可能
- 実績あるオープンソースのテクノロジとフレームワークにより、RISC-Vエコシステムに容易に統合可能

コダシップは、カスタマイズを自動化するための独自のツールセットを提供: Codasip Studio.

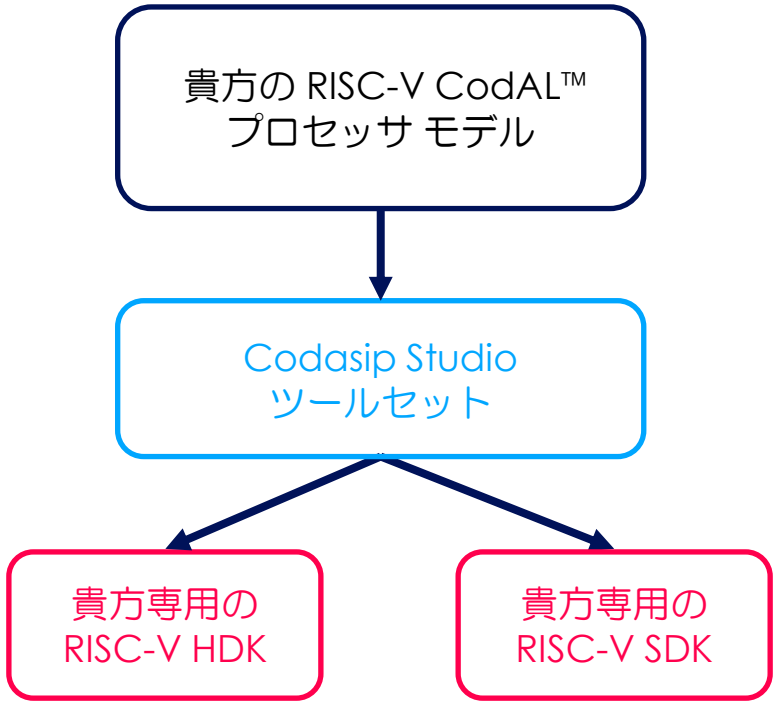
codasip STUDIO

私たちの技術基盤

Codasip Studioとは？

RISC-Vプロセッサを高速かつ容易に改造するためのユニークなツール群
オールインワン、高度な自動化。2014年発売、大手ベンダーでシリコン実証済

- プロセッサを高位アーキテクチャ記述言語で記述
- 下記のカスタマイズを可能に:
  - 命令セット アーキテクチャ
    (ISA: Instruction set architecture)
  - マイクロ アーキテクチャ
- どちらでも選べます:
  - 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**



命令セット精度モデル

Codasip Studioからの自動生成
- コンパイラ
- シミュレータ
- プロファイラ
- デバッガ

ISA探求に必要なもの全て

サイクル精度モデル

Codasip Studioからの
追加自動生成:
- RTL
- 検証環境

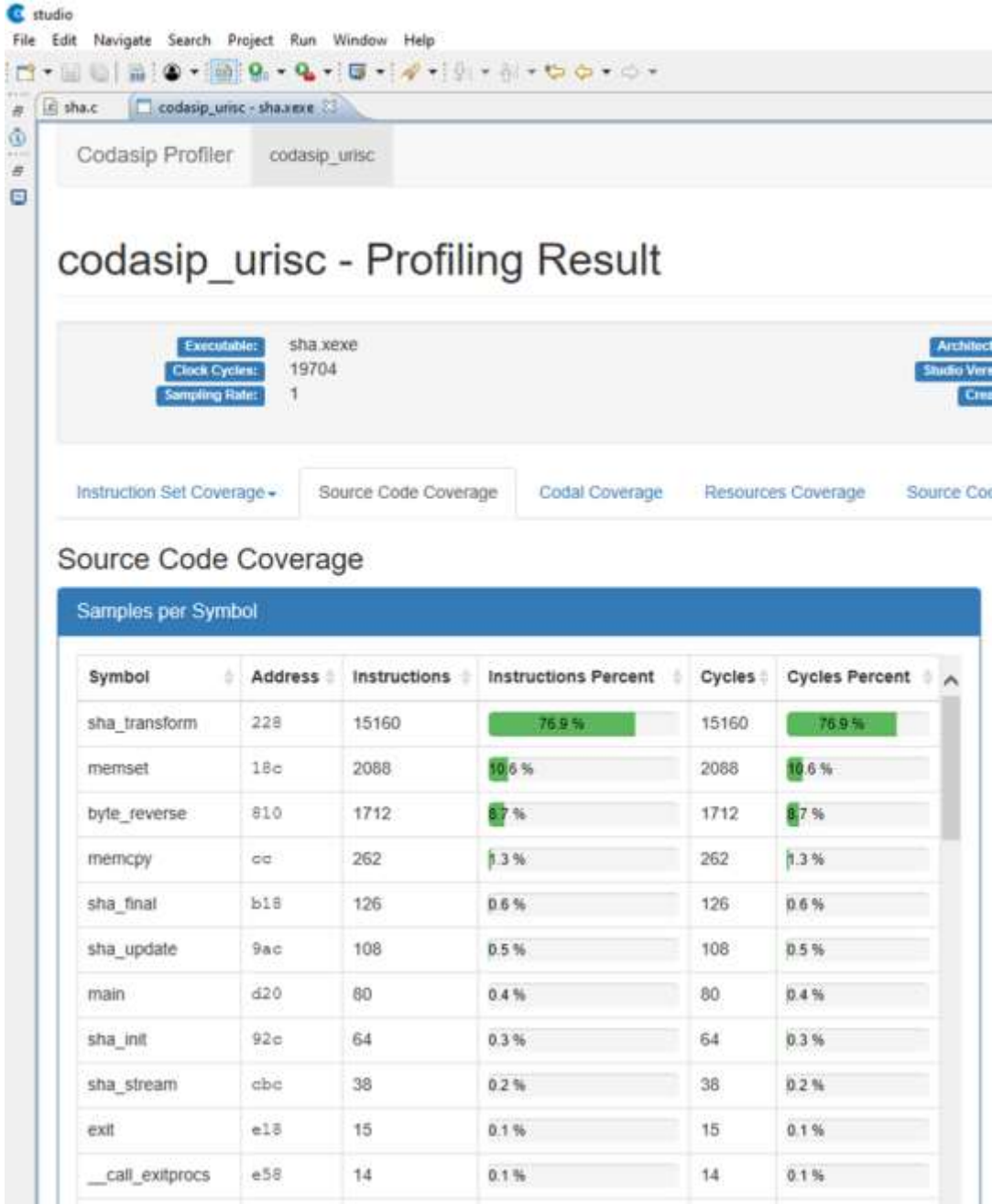
ISA探求完了後に生成

インストラクション精度モデルは、
そのまま使用

Codasip Studioは、**高速プロトタイピング開発**に最適!

デザインスペースの探索

- 自動生成されるコンパイラツールチェーンとプロファイラにより、独自のアプローチで高速なデザインスペース探索が可能
  - 組込みソフトウェアの**プロファイリング**により、プロセッサ最適化の可能性を探索
  - ハイレベルなコーディング手法により、ISA拡張を**迅速に実装**可能
  - Codasip Studioは、SDK全体を自動生成し、その影響を迅速に分析することが可能

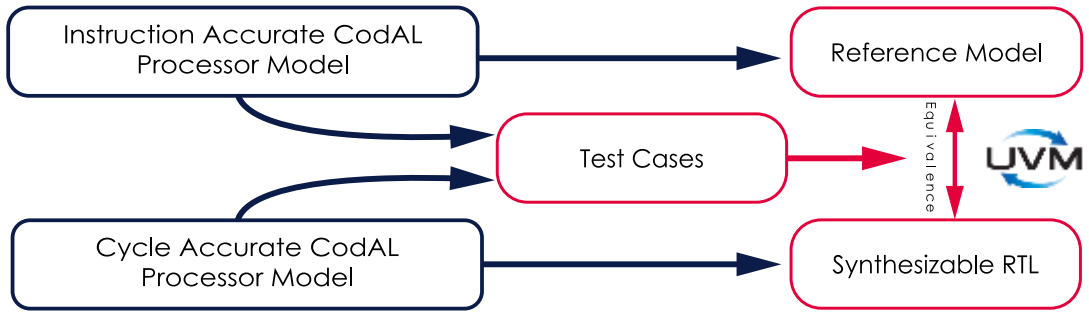


HDKとSDKの自動生成

- Hardware Development Kit (HDK)
  - RTL (Verilog/VHDL/SystemVerilog)
    - 高い可読性
    - CodALソースへのリンク
  - 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
- 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
  - サイクル精度(CA)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


プロセッサIPの検証

- 標準化されたアプローチ、シミュレーション、静的形式解析に基づく**強力な手法**
- 整合性チェック機能
- ランダム アセンブラ プログラム生成ツール
- **UVM検証環境**
  - Codasip StudioがSystemVerilogの環境を自動生成
  - RTLが仕様に合致しているかを確認



Codasip Architectural Language

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

- プロセッサ開発に特化した言語
- 1つのCodALモデルで
  複数のマイクロアーキテクチャの
  実装が可能

理解し易いC言語ライク言語、高抽象度でプロセッサのアーキテクチャ、構造を表現し、自動生成可能にする


```
/*    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];
            };
};
```

コンパクトで表現力豊かな言語

SystemVerilog → codal

```
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
```

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

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コードのリンク
  - 次のページに示します

```
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
```

例: B命令拡張のファンクション モデル

- CodALモデルを一人のエンジニアが10日間で実装
- Codasip Studioが下記等を含む Software development kit（SDK）を自動生成:
  - インストラクション精度シミュレータ (IA: Instruction accurate simulator)
  - 拡張機能の影響が確認できるプロファイラ
  - Cコンパイラ
    - 命令のサブセットを自動的に使用 （回転、コンパクト命令、シフトなど）

```
element i_gzip
{
    use opc_gzip as opc;
    use reg_any as dst, src1;
    use shift_imm as imm ;
    assembly { opc dst “,” src1 “,” imm};
    binary { opc[OPC_FRAG_SHIFT] imm src1 opc [OPC_FRAG1] dst opc [OPC_FRAG0]};
    semantics
    {
        rf_gpr_write (dst, gzip_uXlen(rf_gpr_read(src1), imm));
    };
};
set isa_b += i_gzip;
```

```
uXlen gzip_uXlen (const uXlen rsc1, const uXlen mode)
{
    uint32 zip_mode, x ;
    x = rsc1;
    zip_mode = mode & 31;
    if (zip_mode & 1)
    {
        if(zip_mode & 2)
            x = gzip_stage (x, MASK_ZIP2_L, MASK_ZIP2_R, 1);
        if(zip_mode & 4)
            x = gzip_stage (x, MASK_ZIP4_L, MASK_ZIP4_R, 2);
        if(zip_mode & 8)
            x = gzip_stage (x, MASK-ZIP8_L, MASK_ZIP8_R, 4);
        if(zip_mode & 16)
            x = gzip_stage (x, MASK_ZIP16_L, MASK_ZIP16_R, 8);
    }
```

例: B命令拡張のマクロアーキテクチャ実装モデルの追加

- CodALモデルを一人のエンジニアが3週間で実装
- Codasip Studioが下記等を含む Hardware development kit (HDK) を自動生成:
  - RTL
  - テストベンチ
  - UVMベースの検証環境

```
#ifdef OPTION_EXTENSION_B
    case SLO:
        ex_result = ones_shifter_32(SLO, ex_aluop1, ex_aluop2);
        break;
    case SRO:
        ex_result = ones_shifter_32(SRO, ex_aluop1, ex_aluop2);
        break;
    case ANDC:
        ex_result = (uXlen)ex_aluop1 & (~ ex_aluop2);
        break;
    case ROTR:
        ex_result = ex_aluop1 >>> ex_aluop2;
        break;
    case ROTL;
        ex_result = ex_aluop1 <<< ex_aluop2;
        break;
    case CTZ;
        ex_result = codasip_ctlz_unint32(ex_aluop1);
        break;
    case CLZ;
        ex_result = codasip_cttz_unint32(ex_aluop1);
        break;
#endif
```

codasip
CODESPACE

ソフトウェア開発用IDE

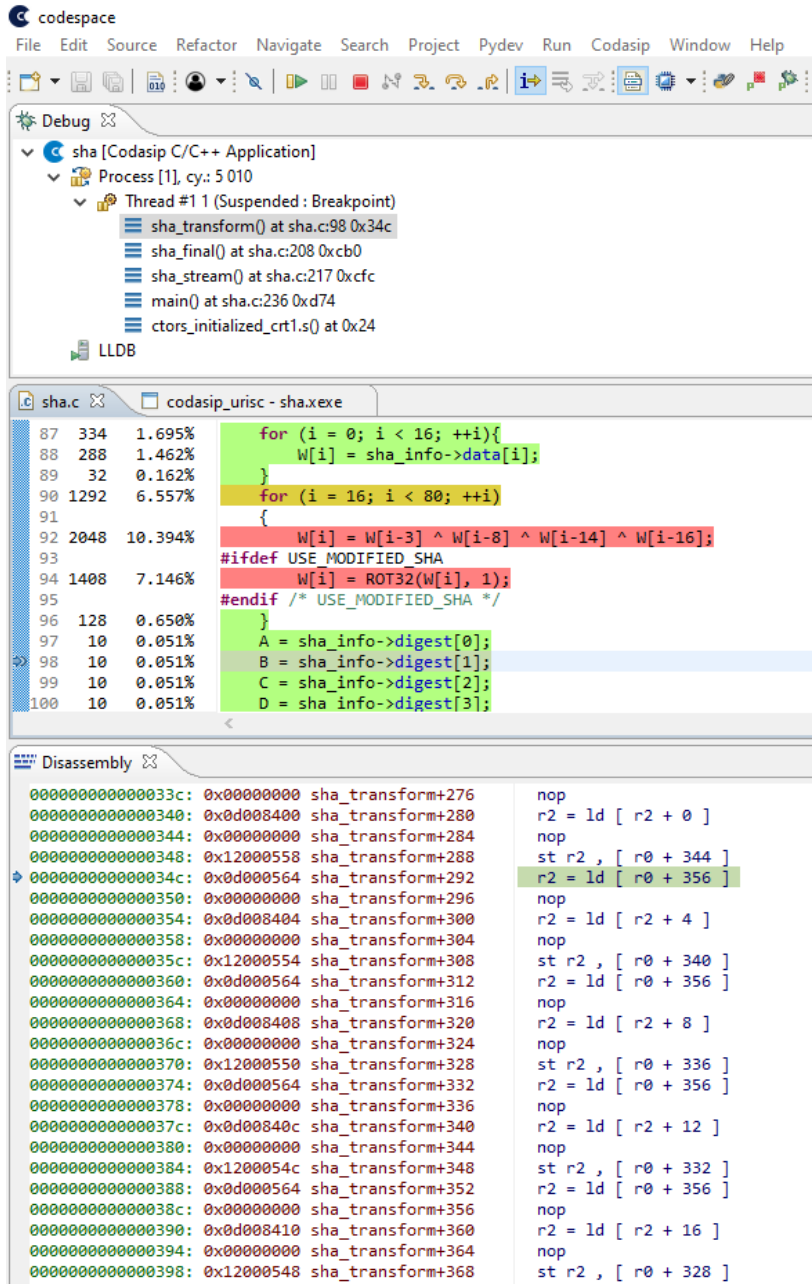
Codasip CodeSpaceとは？

- RISC-Vプロセッサ用のソフトウェアを開発するために必要なすべての機能を提供する**統合開発環境**
  （IDE: integrated development environment）

- 複数のSDKをIDEで管理し、ソフトウェア・プロジェクトを割り当てることが可能

- **プロファイラ**をエディタに直接統合

- **デバッグ作業の効率化**
  - ポート、信号、パイプラインを表示可能
  - 同じ環境下で、同じソフトウェアプロジェクトをシミュレータから**オンチップ デバッグ**へ移行可能



オンチップ・デバッグ

- 全Codasip RISC-V プロセッサは RISC-V Debug をサポート(0.13.2)
  - OpenOCD + LLDBベース
- デバッガ・オプション
  - Codasip CodeSpace debugger
    - Codasip RISC-Vプロセッサのカスタマイズ対応
- Segger, IAR, Lauterbach
- JTAGサポート
  - 4ピン または 2ピン
  - OpenOCD対応プローブ
  - Segger J-Link, IAR I-Jet, Lauterbach TRACE32

まとめ

codasip
RISC-V PROCESSORS

商用品質のRISC-V IP
を提供するリーディングカンパニー

- 包括的なオフザシェルフ（既製品）ポートフォリオ
- 完全な事前検証済みIP
- 専門性の高い専任のカスタマーサポートスタッフが対応

RISC-Vのカスタマイズ
を簡単に自動化

- カスタマイズで、圧倒的な真の差別化を実現
- コダシップは、カスタマイズのための完全なツールやリソースのセットを提供

codasip
STUDIO

カスタムアーキテクチャ
を設計するためのユニークな手法

- Codasip Studioを用いた高度な自動化プロセス
- お客様や社内のプロセッサ開発で幾度も使用されている高水準の設計フロー

ケーススタディ

サクセスストーリー

Thank you.

Now, it’s your turn!

www.codasip.com

お問い合わせは、takaaki.akashi@codasip.com まで