RISC-V (リスク・ファイブ)
について語り合おう

講師　竹内 陽児　(イーソル)
講師　塩谷 亮太　(東京大学)
講師　柴田 貴康　(プライムゲート)
コーディネータ　山崎 進　(北九州市立大学)

RISC-Vへの誘い

- RISC-Vを学ぶ利点

1. なぜこのように設計したのか，理由や意図が明示されている
2. ISAがシンプル
3. なんとフリー！ 太っ腹！
    1. free beer の意味の free 無償な
    2. free speech の意味の free 自由な
4. ツールが充実している
5. 多くのメーカーが参入を表明している

順番について

- 資料の順番

1. 柴田さん
2. 塩谷先生
3. 山崎
4. 竹内さん

- 実際の発表順番(議事録順)

1. 竹内さん
2. 塩谷先生
3. 山崎
4. 柴田さん

RISC-Vのゆる～い話

第2回マイコンボードもんもん会@京都
（ALGYAN関西支部）
2019年01月20日(日)

@shibatchii

発表を聞くにあたっての注意点

- あまり真剣に見ないでください。
  - ボロが丸わかりです。
- あまり深く考えないでください。
  - え～それって〇〇じゃん、こじつけだーってとこ多いです。
- 笑って、和やかな心で見てください。
  - その方が幸せになれます。
- メモ取って真剣に聞くような内容は出てきません。
  - 会社に帰って報告書書けなくっても知りません。
  - @shibatchii個人の趣味のお話です。(会社ほぼ関係ナシ)

自己紹介

- 名前： @shibatchii
- 仕事：ASIC、FPGAの設計検証
  某ゲーム機のDDR I/Fとか
  本拠地は山口県宇部市
- 趣味：オートバイでツーリング
  FPGA,マイコン いじり

- ブログ:http://shibatchii.cocolog-nifty.com/blog/
- ホームページ:http://shibatchii.cool.coocan.jp/
- SlideShare:https://www.slideshare.net/TakayasuShibata/edit_my_uploads
- Twitter:@shibatchii フォロー歓迎

さてさて、RISC-Vとは

突然ですが、アンケート

- RISC-Vって知ってる？

■ よく知っている
■ なんか聞いたことある。興味ある
■ よく知らない

RISC-Vとは

- もともとカリフォルニア大学バークレイで教育用として作られていた。今はRISC-V財団が管理
- CPUの命令セット・アーキテクチャ (ISA)
- グーグル、オラクル、ヒューレット・パッカード・エンタープライズ（HPE）などが開発に参加
- 完全にオープンで自由に使える命令セットアーキテクチャ
- アドレッシングは32/64/128bitサポート

- 上記、Wikipedia参照しました。https://ja.wikipedia.org/wiki/RISC-V
- 詳細は FPGAマガジンNo.18とかRISC-VのWebページを参考にしてください。

@shibatchii

特徴

- 自由度が高い
  - 必要なバス幅、命令セットを選んで構成できる
  - バスも自由：AMBA,Sonic,VME,オレオレバス,etc
  - メモリマップも自由：スタートベクタが途中にあってもOK
- 命令セット構成例
  - RV32I MAFDQC

この並べ方にも規則があるよ

■ RV32I:基本命令　■ M:乗徐算命令　■ A:アトミック命令　■ C:圧縮
■ F:単精度小数点　■ D:倍精度小数点　■ Q:4倍精度小数点　■他にも L,B,J,T,P,V,Nなど有

ここまで必要ないなら、例えばRV32IMACだけでもOK

@shibatchii

モジュール一覧

ベース	バージョン	凍結?	機能
RV32I	2.0	Y	32bit 基本整数 (レジスタ数 : 32)
RV32E	1.9	N	32bit 基本整数 (レジスタ数 : 16)
RV64I	2.0	Y	64bit 基本整数 (レジスタ数 : 32)
RV128I	1.7	N	128bit 基本整数 (レジスタ数 : 32)
拡張	**バージョン**	**凍結?**	**機能**
M	2.0	Y	整数乗算および除算標準拡張
A	2.0	Y	原子命令の標準拡張
F	2.0	Y	単精度浮動小数点標準拡張
D	2.0	Y	倍精度浮動小数点標準拡張
Q	2.0	Y	倍精度浮動小数点標準拡張
L	0.0	N	10進浮動小数点の標準拡張
C	2.0	Y	圧縮された命令の標準拡張
B	0.0	N	ビット操作の標準拡張
J	0.0	N	動的に翻訳された言語の標準拡張
T	0.0	N	トランザクションメモリの標準拡張
P	0.1	N	圧縮された(パックされた)SIMD 命令の標準拡張
V	0.2	N	ベクトル演算標準拡張
N	1.1	N	ユーザーレベル割り込みの標準的な拡張

凍結(Freeze)となっているところは今後仕様が変わらない。
そうでないところはまだ検討中だったり、影も形もない。

注：RV32G とか”G"の場合がある。GはGeneral purpose (汎用、一般的用途)
これは G=(IMAFD) と読み替える。つまりRV32G = RV32IMAFD

@shibatchii

I 基本命令一覧

命令	説明
SLL rd,rs1,rs2	論理左シフト
SLLI rd,rs1,shamt	論理側値左シフト
SRL rd,rs,rs2	論理右シフト
SRLI rd,rs1,shamt	論理即値右シフト
SRA rd,rs1,rs2	算術右シフト
SRAI rd,rs1,shamt	算術即値右シフト
ADD rd,rs1,rs2	加算
ADDI rd,rs1,imm	加算即値
SUB rd,rs1,rs2	減算
LUI rd,imm	ロード即値上位
AUIPC rd,imm	PCに上位即値加算
XOR rd,rs1,rs2	排他的論理和
XORI rd,rs1,imm	排他的論理和即値
OR rd,rs1,rs2	論理和
ORI rd,rs1,imm	論理和即値
AND rd,rs1,rs2	論理積
ANDI rd,rs1,imm	論理積即値
SLT rd,rs1,rs2	符号付き比較
SLTI rd,rs1,imm	符号付き即値比較
SLTU rd,rs1,rs2	符号なし比較
SLTIU rd,rs1,imm	符号なし即値比較
BEQ rs1,rs2,imm	分岐 ＝
BNE rs1,rs2,imm	分岐 ≠
BLT rs1,rs2,imm	分岐 ＜
BGE rs1,rs2,imm	分岐 ≧
BLTU rs1,rs2,imm	符号なし分岐 ＜
BGEU rs1,rs2,imm	符号なし分岐 ≧

命令	説明
JAL rd,imm	PC保存とジャンプ
JALR rd,rs1,imm	PC保存とレジスタ加算ジャンプ
LB rd,rs1,imm	メモリバイトロード
LH rd,rs1,imm	メモリハーフワードロード
LBU rd,rs1,imm	メモリバイト符号なしロード
LHU rd,rs1,imm	メモリハーフワード符号なしロード
LW rd,rs1,imm	メモリロードワード
SB rs1,rs2,imm	メモリバイトストア
SH rs1,rs2,imm	メモリハーフワードストア
SW rs1,rs2,imm	メモリワードストア
CSRRW rd,csr,rs1	CSRリードライト
CSRRS rd,csr,rs1	CSRリードセットビット
CSRRC rd,csr,rs1	CSRリードクリアビット
CSRRWI rd,csr,imm	CSR即値リードライト
CSRRSI rd,csr,imm	CSR即値リードセットビット
CSRRCI rd,csr,imm	CSR即値リードクリアビット
ECALL	実行呼び出し例外
EBREAK	デバッガ呼び出し例外
FENCE	メモリとIOの同期
FENCE.I	メモリとIOの同期即値指定

全47命令、ただし簡易な実装では黄、桃、赤色の~~泥臭い命令~~所はSYSTEM命令1個、NOPに置き換えられるので38命令でOK
CSRはコントロール・ステータス・レジスタ

@shibatchii

疑似命令

- みんな大好きNOP命令が無い？ nop
  - 大丈夫 addi x0,x0,0 で置き換え
- 単純JUMPしたいんだけど？ jmp offset
  - jal x0,offset でいける
- 戻ってくる命令ないじゃん？ ret
  - jalr x0,x1,0 を使うのさ
- ２の補数を作りたいんだけど？ neg rd,rs
  - sub rd,x0,rs でやってちょ

これらはアセンブラが置き換えてくれる。他にも疑似命令いろいろ有。

★ x0レジスタは0固定になっているレジスタ

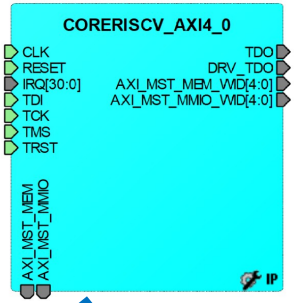
レジスター覧

レジスタ	ABI 名	説明	セーバー
x0	zero	ハードワイヤードゼロ	ー
x1	ra	リターン アドレス	呼び出し元
x2	sp	スタック ポインタ	呼び出し先
x3	gp	広域 ポインタ	ー
x4	tp	スレッドポインタ	ー
x5	t0	一時的/代替リンク・レジスタ	呼び出し元
x6-7	t1-2	一時変数	呼び出し元
x8	s0/fp	保存レジスタ/フレームポインタ	呼び出し先
x9	s1	保存レジスタ	呼び出し先
x10-11	a0-1	関数の引数/戻り値	呼び出し元
x12-17	a2-7	関数の引数	呼び出し元
x18-27	a2-11	保存レジスタ	呼び出し先
x28-31	t3-6	一時変数	呼び出し元

@shibatchii

Microsemi FPGA 実装例

Catalogから
SmartDesignに持っ
てくるとこんな感じ

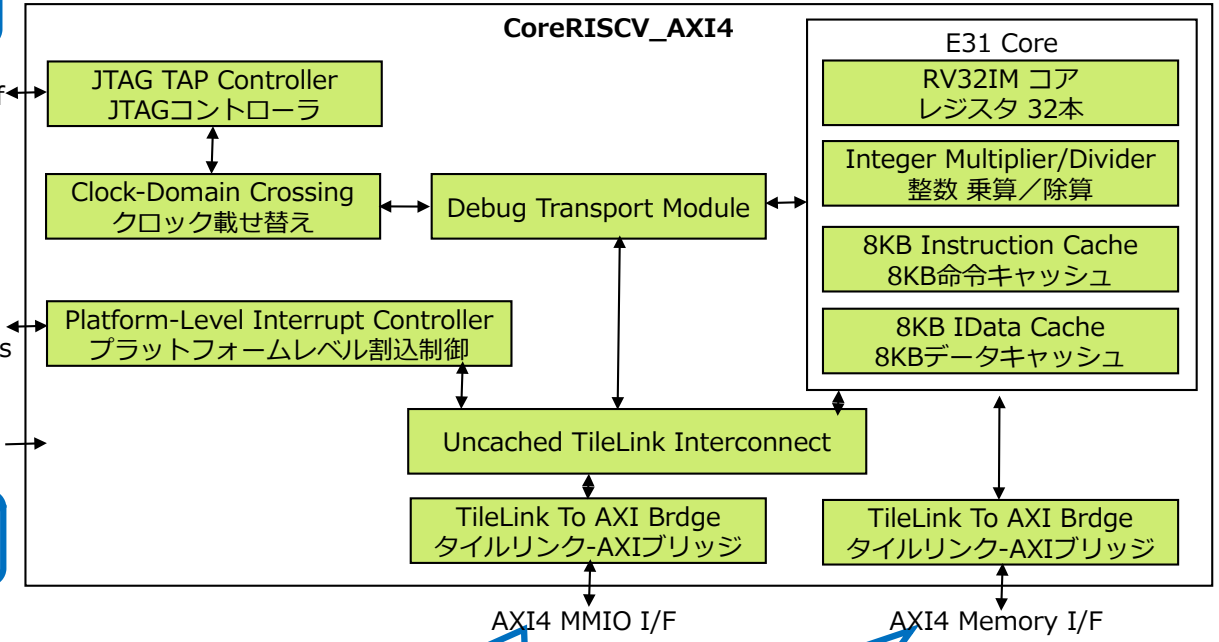


中身はSiFive's E31
Coreplex

JTAG

割り込
み入力

CLK
RESET



周辺デバイス用AXI4
キャッシュ無し

メモリ用AXI4
キャッシュ有り

@shibatchii

細かいことはいいんだよ
ともかく、使ってみたい！

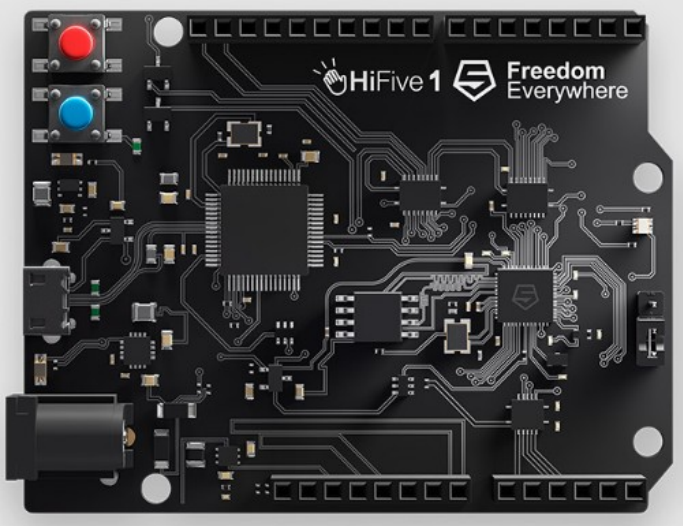
@shibatchii

シミュレータで動かす

- Cコンパイラ、ライブラリなど、ダウンロードしてがんばって環境作る
  - https://riscv.org/software-status/#c-compilers-and-libraries
- シミュレータをダウンロードして、がんばって 略)
  - https://riscv.org/software-status/#simulators

@shibatchii

ボードで動かす

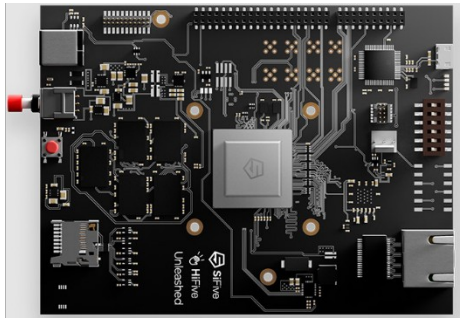


- SiFive の Freedomボード
  - マルツエレクトロニクスで 9980円で購入可
  - 開発環境は Freedom E SDK とか Arduino IDE

FPGAで動かす

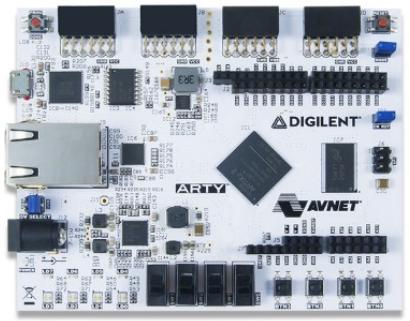
- Future Electronics の Creativeボード
  - Future Electronicsから12000円位で購入可
  - 開発環境は Microsemi LiberoSoc,CoreConsole

その他いろいろあるよ

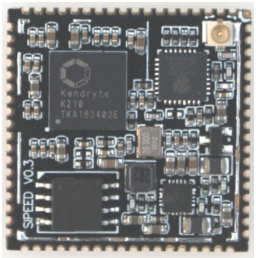


- SiFive HiFive Unleashed
  140000円位

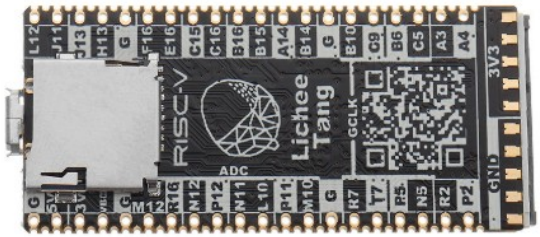
- LoFive RISC-V
  3000円位



- Digilent Arty A7:
  Artix-7 FPGA XilinX
  15000円位



- Sipeed MAIX: Fisrt
  RV64 AI board for edge
  600円位



- Lichee Tang RISC-V開発
  ボードMini PC
  3000円位

- FPGA Development
  Board RISC-V
  Development Board
  2500円位

@shibatchii

オススメの書籍、Web

- FPGAマガジン No.18 Googleも推すオープンソースCPU RISC-Vづくり
  CQ出版社
  - http://cc.cqpub.co.jp/lib/system/doclib_item/1149/
  - https://shop.cqpub.co.jp/hanbai/books/46/46281.html

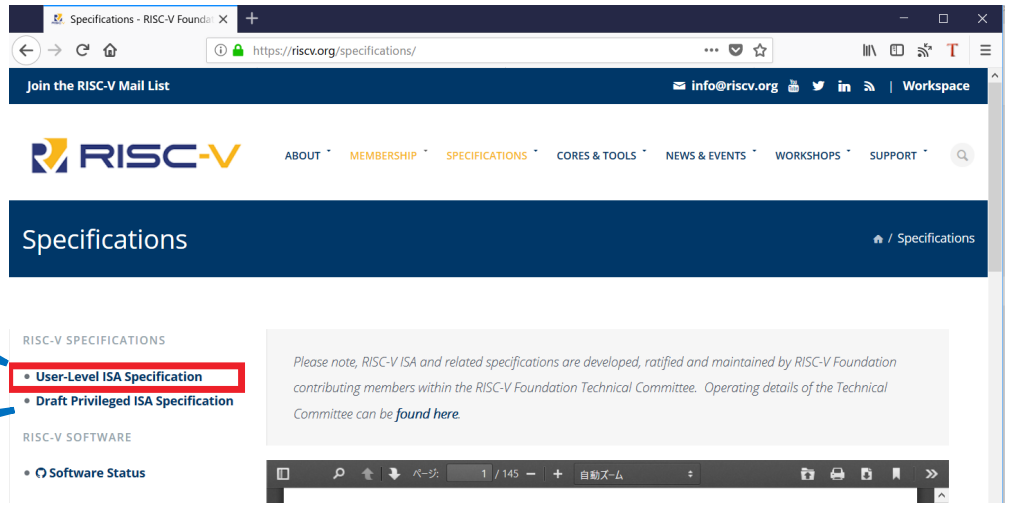
- RISC-V原典 オープンアーキテクチャのススメ　日経BP社
  - https://www.nikkeibp.co.jp/atclpubmkt/book/18/269170/

- プログラマのためのFPGAによるRISC-Vマイコンの作り方 Kindle
  堀江 徹也さん　@tetsuya_horie
  - https://www.amazon.co.jp/gp/product/B07G2CHSK3/ref=oh_aui_d_detailpage_o00_?ie=UTF8&psc=1

- FPGA開発日記 msyksphinzさん　@dev_msyksphinz
  - http://msyksphinz.hatenablog.com/

RISC-V仕様書

RISC-Vの仕様書はここ。英語だよ。https://riscv.org/specifications/



@shibatchii

日本語訳の場所 @shibatchii翻訳

- GitHub
  - https://github.com/shibatchii/RISC-V
    - RISC-V_spec_manual_v2.2_jp.pdf
    - riscv-privileged-v1.10_jp.pdf
    - MSoffice Word、 LibreOffice Writer 形式も有

ANDI、ORI、XORI は、レジスタ rs1 と符号拡張 12 ビットの即値をビット単位で AND、OR、XOR し、その結果を rd に格納する論理演算です。
注：XORI rd、rs1、-1 は、レジスタ rs1 のビット単位の論理反転を実行します（アセンブラ疑似命令 NOT rd、rs）。

31 25	24 20	19 15	14 12	11 7	6 0
imm[11:5]	imm[4:0]	rs1	funct3	rd	opcode
7	5	5	3	5	7
0000000	shamt[4:0]	src	SLLI	dest	OP-IMM
0000000	shamt[4:0]	src	SRLI	dest	OP-IMM
0100000	shamt[4:0]	src	SRAI	dest	OP-IMM

定数によるシフトは、I 型形式の特殊化としてエンコードされます。
シフトされるオペランドは rs1 であり、シフト量は I-即値フィールドの下位 5 ビットにエンコードされます。
右シフト型は、I 即値の上位ビットで符号化されます。
SLLI は論理左シフトです（ゼロは下位ビットにシフトされます）。 SRLI は論理右シフトです（0 は上位ビットにシフトされます）。
SRAI は算術右シフトです（元の符号ビットは空いている上位ビットにコピーされます）。

@shibatchii

質問タイム

以上ここまで。

なにか質問あればどうそ。
なんでも良いですよ。

後で聞こうとか思わずここで聞こう。
みんなで情報共有できるよ。(^^)/

@shibatchii

ありがとうございました。

@shibatchii

SWEST21 RISC-V について語り合おう

東京大学大学院 情報理工学系研究科 創造情報学専攻
塩谷 亮太
shioya@ci.i.u-tokyo.ac.jp

塩谷 亮太

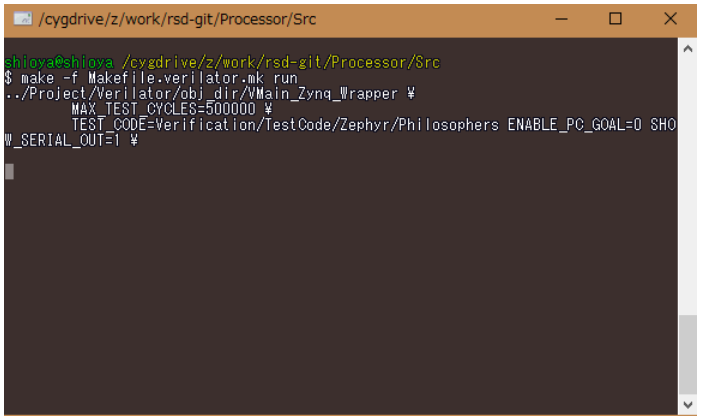
- 所属：
  - ◇ 東大 情報理工学系研究科 創造情報学専攻 准教授
- 専門：コンピュータ・アーキテクチャ
  - ◇ 主に汎用 CPU のマイクロアーキテクチャを研究しています
  - ◇ 回路やセキュリティ，言語処理系とかも
  - ◇ 自動運転用 SoC の研究・開発（株 ティアフォー）
- 去年着任のため，まだ学生さんがおりません
  - ◇ 修士・博士（社会人含む）の学生さん募集中！

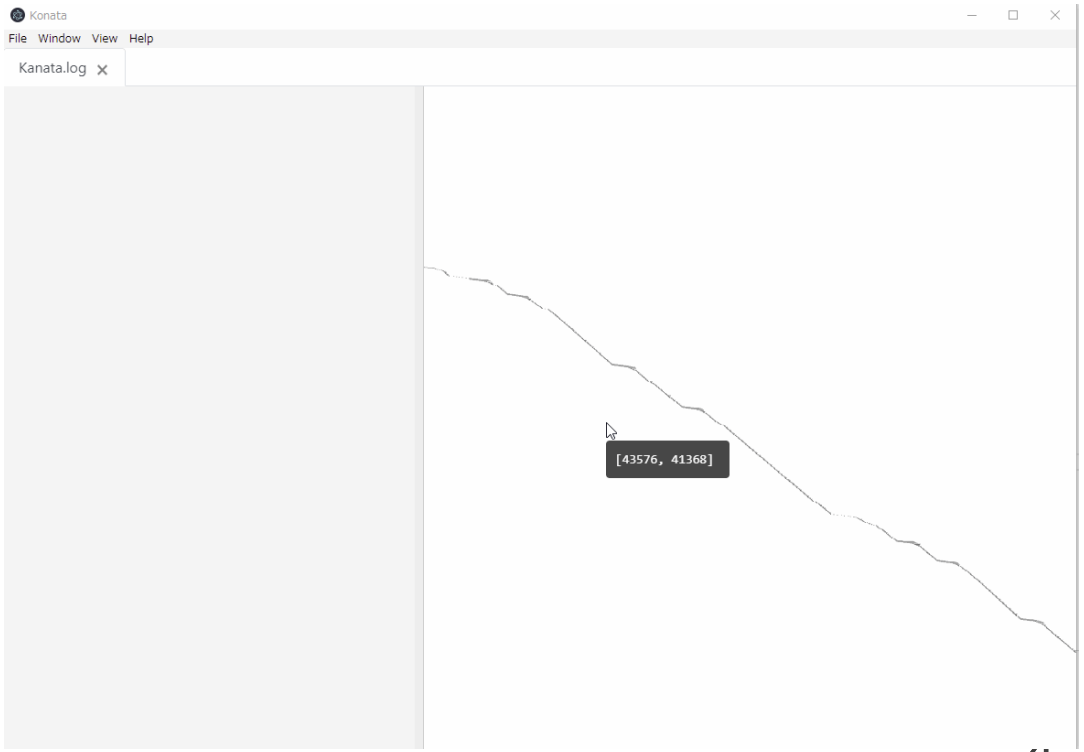
今日のお題：RISC-V

- 最近，世界的に盛り上がっている
  - ◇ 上記は RISC-V Foundation のプラチナスポンサーのみなさん
  - ◇ 企業でも使用に向けて動いている
  - ◇ SiFive のものや中国の謎ボード等，動くものも発売されている

塩谷の RISC-V に対する取り組み

- CPU シミュレータ「鬼斬」
  - C++ で実装されたサイクル・アキュレートなシミュレータ
  - 32bit/64bit 実装で，SPECCPU 2006/2017 の全ベンチマークが動作
- プロセッサ「RSD」
  - SystemVerilog で実装された OoO スーパスカラ・プロセッサ
  - 32bit IM+特権命令の実装で，Coremark や Zephyr OS のアプリが動作

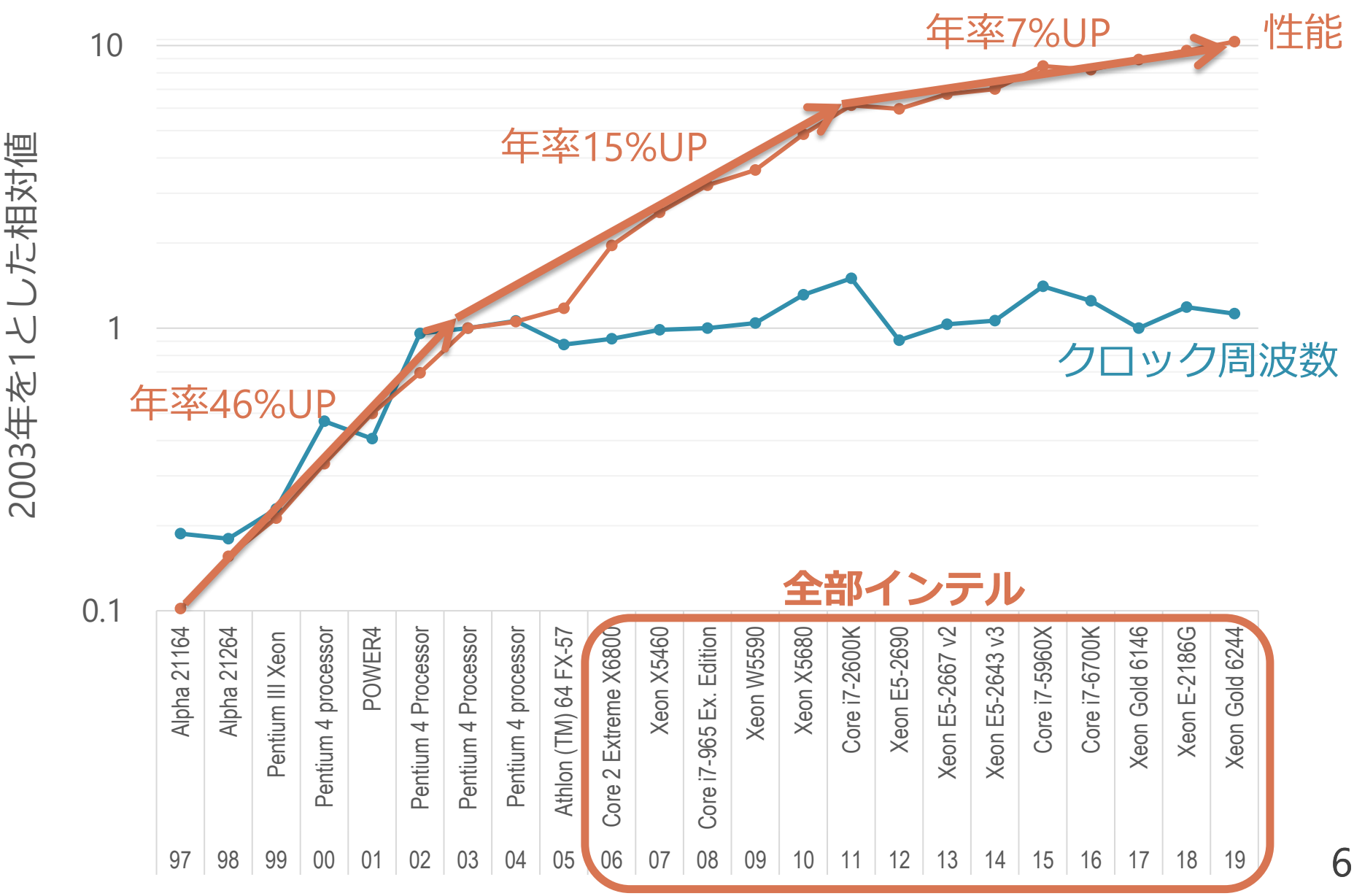




今日のお題：RISC-V

- RISC-V
  - ◇ 比較的最近登場した，CPU の命令セットのオープンな規格
- 研究者としての塩谷の立場：
  - ◇ 汎用 CPU を速く&省電力にしたい
    - □ 速く → 普通のプログラムのシングルスレッド性能のこと
  - ◇ RISC-V は非常に良い機会をあたえてくれている

CPU のシングルスレッド性能の変化
各年ごとの SPECCPU int の最高スコア [飯沼宥光 2019]



汎用 CPU の最近の状況

- 長い間，サーバー・デスクトップ向けはインテルがほぼ独占
  - ◇ ほかがみんな自爆したので，競争がない
  - ◇ 敵がいないので性能を伸ばそうと言う動機がない
- 今年に入り，AMD ZEN2 が登場
  - ◇ 性能を大きく伸ばした
  - ◇ （さっきのグラフにはまだ反映されていないです
- なぜ性能が伸びたのか？
  - ◇ **さまざまな研究成果の技術をアグレッシブに取り入れた**
  - ◇ 研究者レベルでは，１０年以上前からわかっていたものが多い

研究と製品化

- 塩谷を含め，研究者はずっと色々な技術を提案してきた
  - ◇ 性能を上げるもの
  - ◇ 性能を保ちながらコンパクトに作る方法
- Oot-of-order スーパスカラの範疇を超えるものも多い
  - ◇ In-order との密結合ハイブリッド
  - ◇ オペランドがレジスタ番号じゃなくて命令間距離なもの

研究と製品化

- 企業は研究成果を入れてくれない
  - 競争がないので入れる動機がない
    - 「大きく変えると検証が大変なので～」
    - 基本構造は98年の Alpha 21264 から変わっていない
- だからといって，勝手に互換 CPU を作ると怒られる

RISC-V

- 命令セットのラインセンスがフリー
  - 実装したハードの設計を自由に公開できる
  - オープンな実装が公開されており，利用できる
- めいめいが勝手に CPU を作れる
  - まず，単純にプレイヤーが増える
    - いままでは超巨大企業に限られていた
  - 攻めた設計のものが出てくる余地が生まれる
  - 塩谷も自作 CPU を RISC-V に移行

RISC-V の利点

- 研究の上
  - ◇ 自由に実装ができるので，プレイヤーが増える
  - ◇ 新しいものを積極的に試せる・出てくる
- 製品としては？

RISC-V を利用する際のメリット

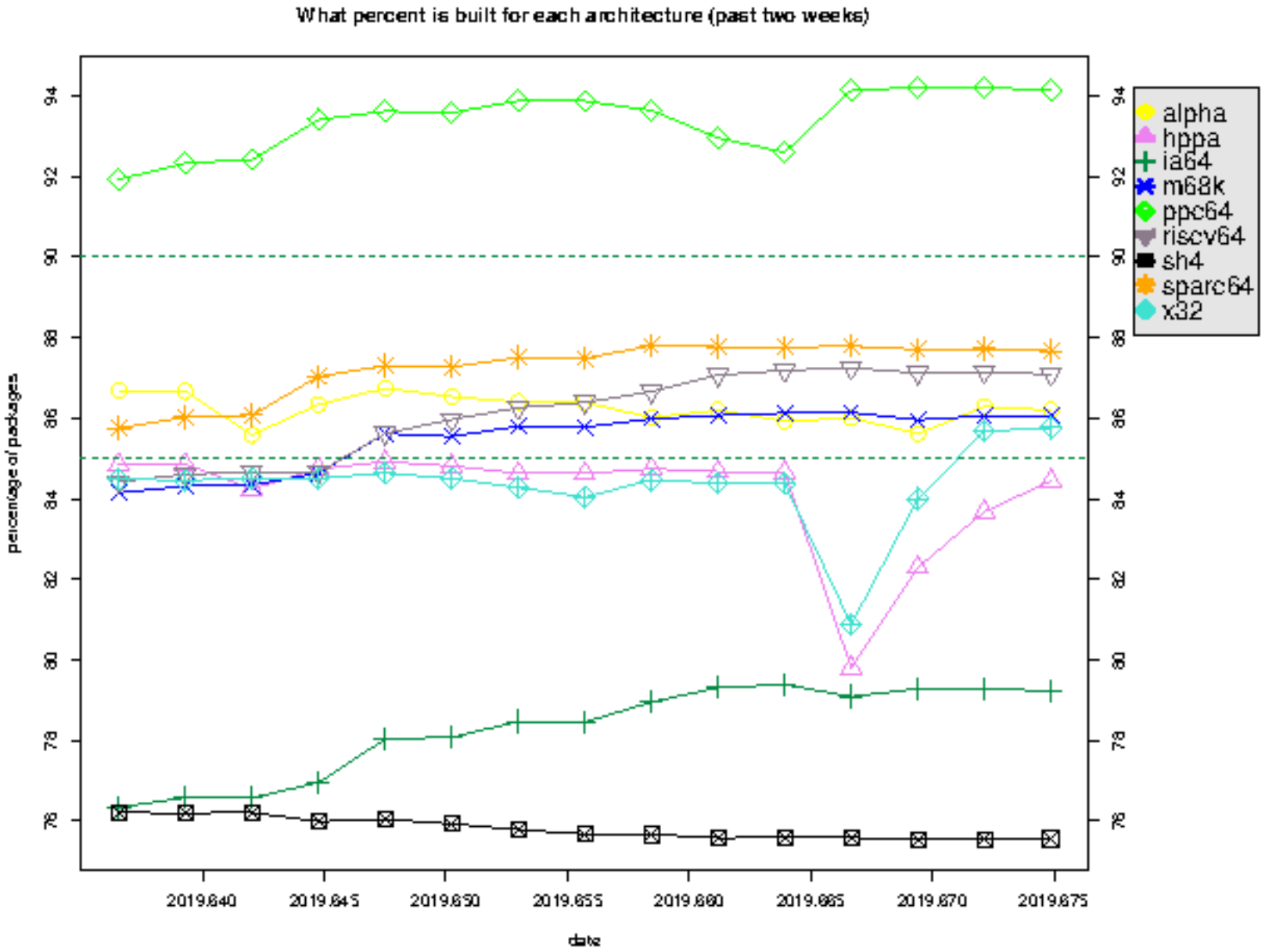
1. 命令セットのラインセンスがフリー
   - ◇ オープンな実装が公開されており，利用できる
   - ◇ 特定のクローズドな IP に縛られない
   - ◇ 使用可能な RISC-V 搭載ボード等も販売されはじめている
2. ツール・チェインがすぐに使える状態で一通りある
   - ◇ 新しいバージョンが使えて，メンテナンスも期待できる
   - ◇ 1. も含めて，メンテナンスのコストがかからない

ツール・チェイン
これらは，すべて本家のソース・ツリーで正式に対応されています

- gcc (g++/gfortran)
  - ◇ 新しいものが使用可能（Ver. 8.1 まで確認）
  - ◇ まれにバグを踏むことがある
- clang/LLVM
  - ◇ 32bit は対応が進んでいるが，64bit はまだあやしい
- QEMU
  - ◇ ほぼ完全に対応が済んでいるようにみえる
  - ◇ バグを踏んだことはない
- Linux
  - ◇ いくつかのディストリビューションで対応
  - ◇ Fedora と Debian が準備を進めている・・・らしい

Debian のビルド済みパッケージ数

https://wiki.debian.org/RISC-V より



RISC-V を利用する際のメリット

1. 命令セットのラインセンスがフリー
2. ツール・チェインがすぐに使える状態で一通りある
3. 実装の簡単さや，独自拡張への考慮
   - ◇ 「最低限の制御のための整数命令」+「アクセラレータ独自拡張」みたいなものも作りやすい
   - ◇ ツールチェインの恩恵を受けられる

まとめ

- RISC-V
  - 比較的最近登場した，CPU の命令セットのオープンな規格
  - 自由に実装ができるので，プレイヤーが増える
    - 新しいものが積極的に試せる・出てくる
  - ツール・チェインがすぐに使える状態で一通りあり，
    実際に試せるボードも出てきている
- 盛り上がっていってほしい

RISC-V ポジショントーク 自己紹介

- 北九州市立大学 国際環境工学部 准教授
- **New!** ナッジ社会実装研究センター
  センター長
- **New!** 学生・就職支援担当
- Personal Vision Co-Creator
- KK-SHiFT
- SWEST実行委員
- Elixir 推し / fukuoka.ex / Hastega 改め
  Pelemay / ZEAM
- 技術相談，共同研究依頼，進路相談，
  適職診断など，随時受付ます

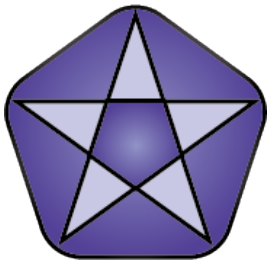
プログラミング言語処理系から見たときの RISC-V

・ RISC-Vを学ぶ利点(下記)は，そのままプログラミング言語処理系にとっても利点です

1. なぜこのように設計したのか，理由や意図が明示されている
2. ISAがシンプル
3. なんとフリー！ 太っ腹！
    1. free beer の意味の free 無償な
    2. free speech の意味の free 自由な
4. ツールが充実している
5. 多くのメーカーが参入を表明している

Elixirからの視点 (1/3)

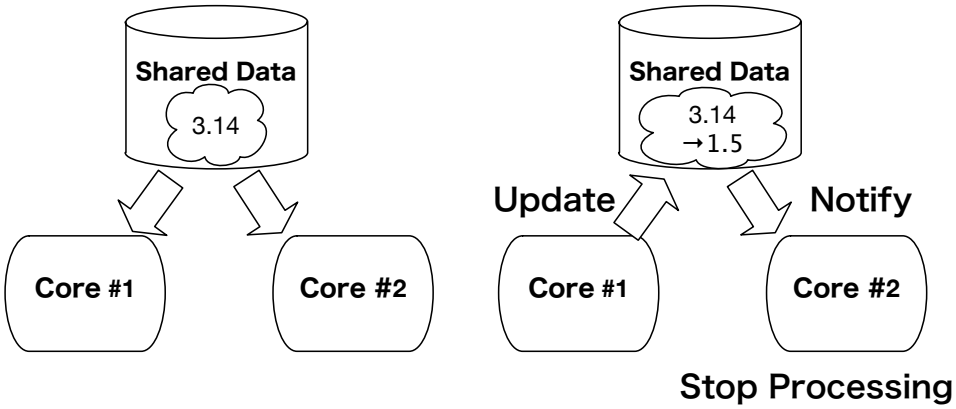
- Hastega 改め Pelemay ではSIMD命令を用いた超並列化が可能
- RISC-V には，パック形式SIMD命令標準拡張機能 RVP を制定しようとしている
- また，肝いりのベクタ方式の超並列機構，ベクトル演算標準拡張機能 RVV も制定されつつある
- 超並列マニアとしては，これらは制覇したい

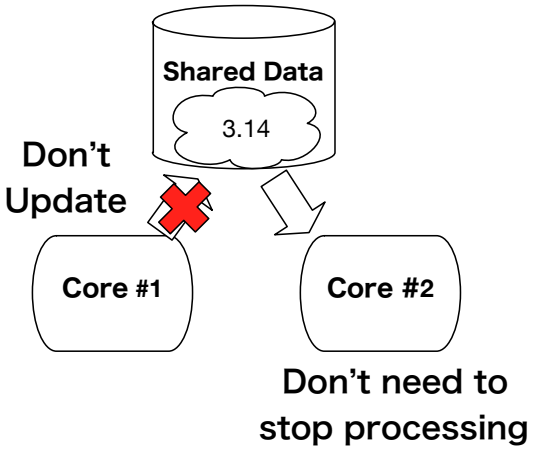


Elixirからの視点 (2/3)

- Elixir の得意は並列プログラミング
- 得意な理由は破壊的更新がないので
  余計な同期・排他制御を排除できるから
- つまりコア数が多ければ多いほど,
  Elixir は有利になる
- RISC-Vはシンプルなので，他の
  アーキテクチャに比べてコアサイズが小
  さい

➡1,000コア以上のマルチコアCPUを
RISC-Vで実現できるのでは？
Elixir の潜在能力を思いっきり発揮できるのでは？





Kilocore

- MIMD方式1000コア (cores)
- 12メモリモジュール (memory modules)
- 32nm プロセスルール (process rule)
- 最大 1.78GHz (1.1V) (maximum)
- 0.9V 時 1.24GHz 17mW
  5.3 pJ / 命令 (low energy)
- Intel i7 や NVIDIA GPU と比べ,
  1.1V時スループット4.3倍以上,
  エネルギー効率9.4倍以上


IEEE.org | IEEE Xplore Digital Library | IEEE-SA | IEEE Spectrum | More Sites

IEEE Xplore Digital Library

Institutional Sign In

Browse | My Settings | Get Help | Subscribe

All | Enter keywords or phrases (Note: Searches metadata only by default. A search for 'smart grid' = 'smart AND grid')

Journals & Magazines > IEEE Journal of Solid-State C... > Volume: 52 Issue: 4

KiloCore: A 32-nm 1000-Processor Computational Array

Publisher: IEEE

8 Author(s) Brent Bohnenstiehl ; Aaron Stillmaker ; Jon J. Pimentel ; Timothy Andreas ; Bin Liu ; Anh T. Tran ; Emmanuel... View All Authors

12 Paper Citations | 1120 Full Text Views

Abstract

Document Sections

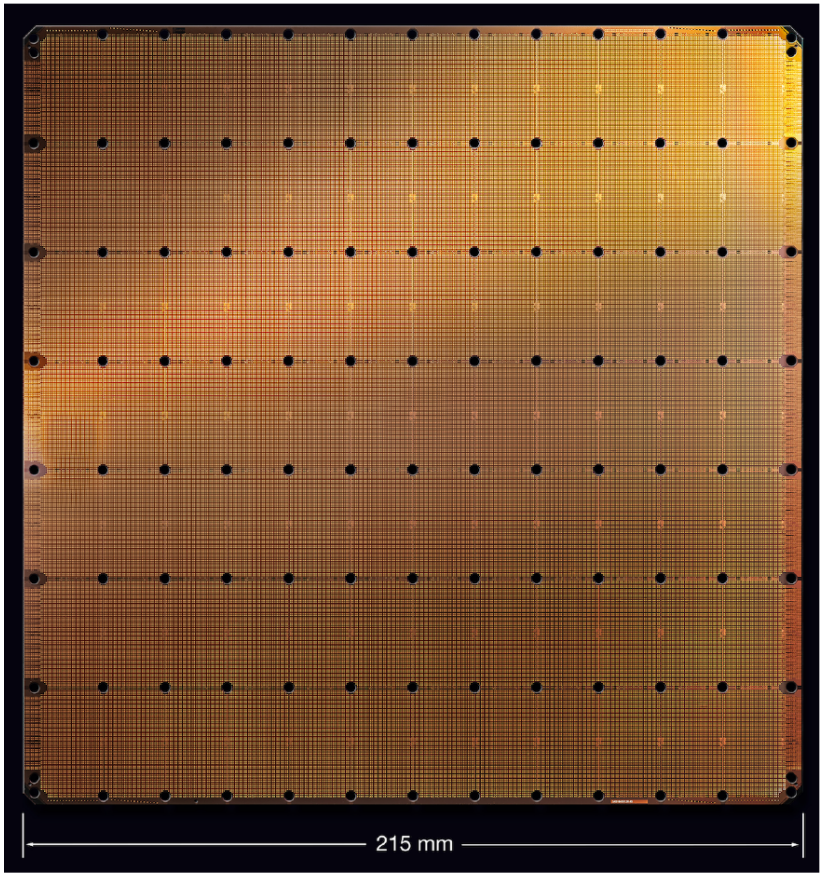
I. Introduction
II. High-Level Architecture
III. Fine-Grain Clocking
IV. Design and Implementation
V. Measured Results

Abstract:
A processor array containing 1000 independent processors and 12 memory modules was fabricated in 32-nm partially depleted silicon on insulator CMOS. The programmable processors occupy 0.055 mm ^{2} each, contain no algorithmspecific hardware, and operate up to an average maximum clock frequency of 1.78 GHz at 1.1 V. At 0.9 V, processors operating at an average of 1.24 GHz dissipate 17 mW while issuing one instruction per cycle. At 0.56 V, processors operating at an average of 115 MHz dissipate 0.61 mW while issuing one instruction per cycle, resulting in an energy consumption of 5.3 pJ/instruction. On-die communication is performed by complementary circuit and packet-based networks that yield a total array bisection bandwidth of 4.2 Tb/s. Independent memory modules handle data and instructions and operate up to an average maximum clock frequency of 1.77 GHz at 1.1 V. All processors, their packet routers, and the memory modules contain unconstrained clock oscillators within independent clock domains that adapt to large supply voltage noise. Compared with a variety of Intel i7s and Nvidia GPUs, the KiloCore at 1.1 V has geometric mean improvements of 4.3× higher throughput per area and 9.4× higher energy efficiency for AES encryption, 4095-b low-density parity-check decoding, 4096-point complex fast Fourier transform, and 100-B

B. Bohnenstiehl *et al.*, "KiloCore: A 32-nm 1000-Processor Computational Array," in *IEEE Journal of Solid-State Circuits*, vol. 52, no. 4, pp. 891-902, April 2017.

The Cerebras Wafer-Scale Engine (WSE)

- ウェハーサイズのチップ
- 1.2兆個のトランジスタ数
- 1辺が 215 mm
- 18GBのオンチップメモリ
  (on-tip memories)
- 40万コア (cores)
- 消費電力 15kW



© 2019 Cerebras systems.

Elixirからの視点 (3/3)

- Elixir のプログラムはデータフローをとても解析しやすい

- 理由はイミュータブルなので，変数の値が変化しない
  (ただし変数を再束縛することはある)

➡依存関係解析をとてもやりやすい

- コアサイズをより小さくしようと思うと，
  スーパースケイラー方式(アウトオブオーダー実行)をやめるのも一手

- 代わりにレジスタ割当や命令スケジューリングを頑張り，メモリアクセスやパイプラインがストールしないようにする

- あるいは合成命令を発行したり， VLIW (Very Long Instruction Word) のような静的に命令並列性を記述するプロセッサ方式を採用したりするのも一手

- RISC-V をベースに，独自プロセッサを開発しやすいのが強み

RISC-V と Elixir で拓ける
新たなシステム実装の研究への道

RISC が登場したときのように…

RISC-Vについて語り合おう(自由について)

https://swest.riscv.jp/

自己紹介：竹内陽児(@ytakeuch)

- 複写機の会社に入社する
- パタヘネ本でRISC Architectureを学ぶ
- MIPSに繋げるASICを幾つか設計する
- MIPSのブートストラップコードを書く
- GNU Hurdの活動でGRUBをいじる
- 後に、モバイルゲームの会社に転職する

黒歴史

- i960に繋がる画像コプロセッサ
- Am29200/Am29240のプリンタコントローラ開発
- 組み込み機器向けCeleron
- MXP5800/5400(別部署)
- DAPDNA-IMX(別部署)

モバイルゲームの会社では

オープンソースのプロダクト

- Arctic.js
- ExGame
- JSX
- H2O
- A2O(元社員)

オープンソースのメリット

- みんなで自由に使ったり、貢献出来る
- 退職後もメンテナンスされる(場合もある)

自由なISAがあったら?

黒歴史は回避できた、、か?

RISC-Vの自由とは?

- BSDライセンス(確認中)
- 自由に実装できる(無償/有償)
- 命令セットを自由に拡張できる
- 研究に自由に使える
- 教育に自由に使える

RISC-Vの自由がもたらす多様性

- 複数の実装から選択できる自由
- Linuxのディストリビューションような多様な進化?

話が飛躍します

RISC-Vは成功するか?

- 自由だから?
- 無償だから?
- 優れたアーキテクチャだから?
- エコシステムが充実しているから?

語り合いたいこと

- 自由であることの利点
- RISC-V成功の鍵は?