

マイクロセミ社FPGAとそのRISC-V 対応状況紹介と、RISC-Vマニュアル 日本語訳のポイント

Design Solution Forum 2018
2018年09月12日(水)

@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/>
- Twitter:@shibatchii フォロー歓迎

@shibatchii

Microsemi FPGAの紹介

@shibatchii

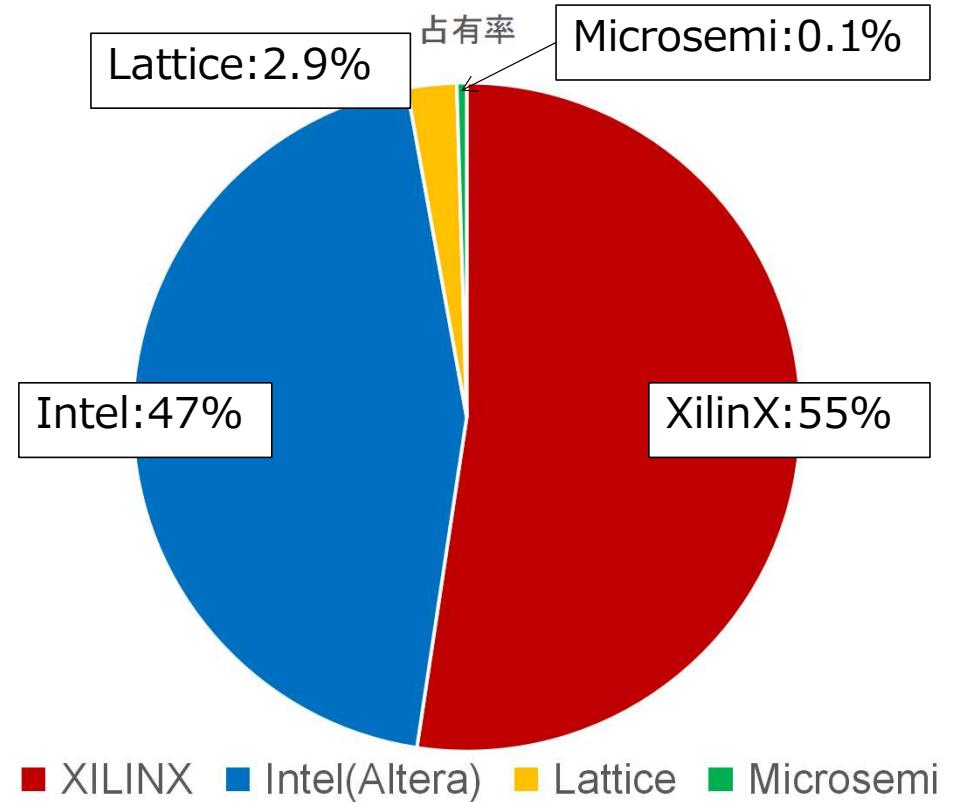
突然ですが、アンケート

■ 使ったことのあるFPGAに手を上げてね。

- Xilinx
- Intel(Altera)
- Lattice(AT&T)
- Microsemi(Actel)

FPGA占有率

- 個人や仕事で使っている人のFPGA占有率
(@shibatchii超主観)



Microsemiってどんな会社？



- 米国 カリフォルニア州 在社
日本オフィスは港区高輪にあるらしい
- 売上高 15億ドル(1800億円)
- 従業員 4800人
- 航空宇宙・防衛、通信、データセンター、産業市場向けの包括的な半導体/システムソリューションを提供

取扱品

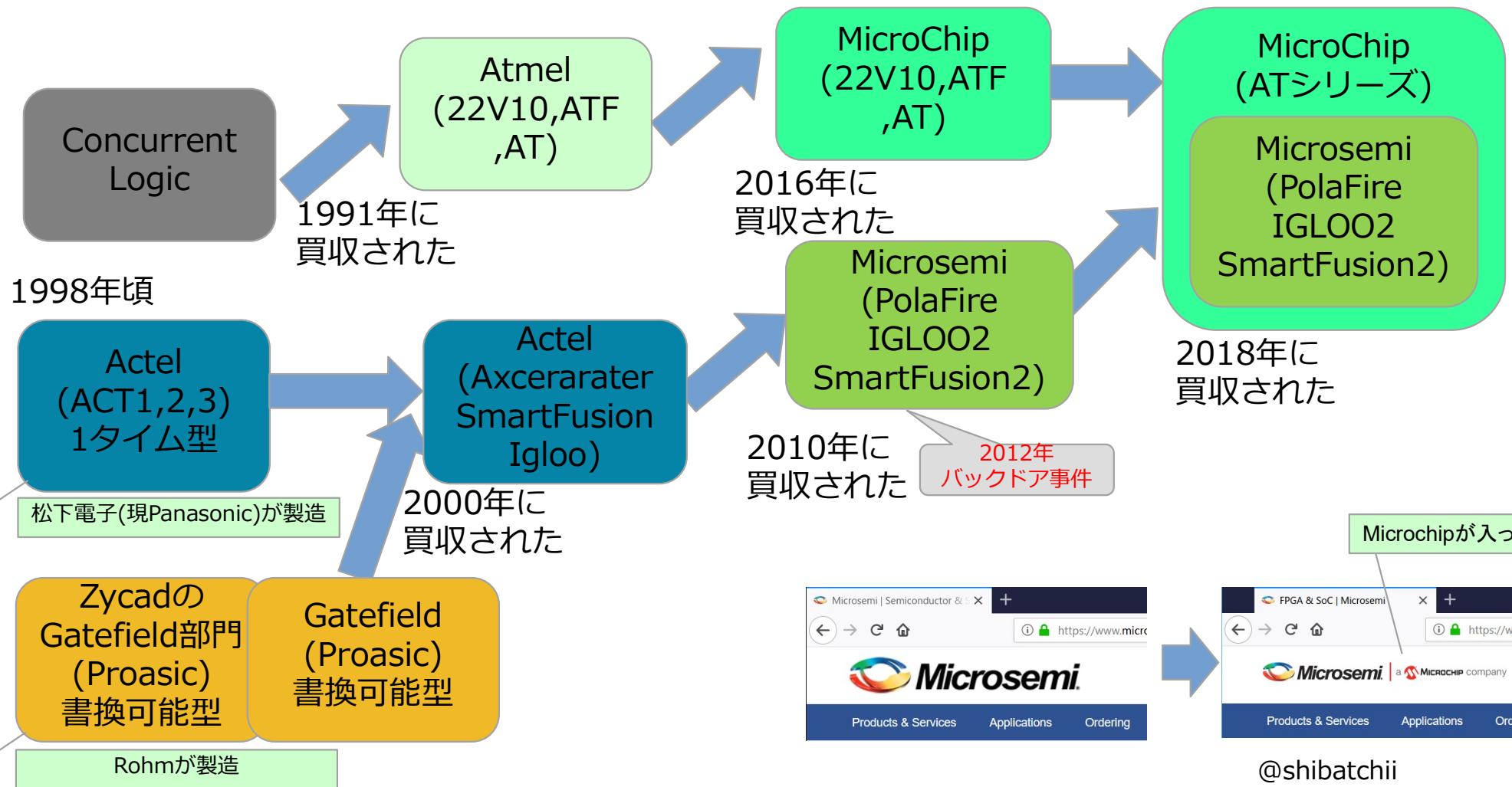
Products		Services
Audio & Voice	Partners	FPGA & Soc Services
Circuit Protection	Power Discretes & Modules	Mixed-Signal ASIC Design Services
Drivers, Interfaces, and PCIe Switches	Power Management	Module & Hybrid Design Services
Ethernet and PoE Solutions	RF Microware & Millimeter Wave	Package Miniaturization Services
FPGA & SoC	Storage Solutions	Rugged Power Supply Design Services
High-Reliability	Support	Synchronization Services
Optical Networking	Timing & Synchronization	

- 部品全般を扱っている。もちろんFPGAもある。
- Ethernetとかストレージの機器やボードなどもあり
日本でいうと、ローム+ムラタみたいな会社

※ 以下資料画像はMicrosemi <https://www.microsemi.com/> より引用

@shibatchii

FPGA遍歴



FPGAラインナップ その1

1タイム型

Antifuse FPGA	
Axcelerator FPGA	eX FPGA
・不揮発性、高速アンチヒューズFPGA	・低電力、低密度 アンチヒューズ FPGA
・125K～2Mのシステムゲート	・3K～12Kシステムゲート
・350MHzのシステム性能	・350MHzのシステム性能
・0.15um,CMOSアンチヒューズプロセス	・0.22umCMOSアンチヒューズプロセス
SX-A FPGA	MX FPGA
・Sea-of-modules アンチヒューズモジュールFPGA	・混合電圧と5Vのみの動作
・12K～108Kシステムゲート	・3K～54Kシステムゲート
・250MHzのシステム性能	・250MHzシステム性能
・66MHzのPCI準拠	・デュアルポートSRAMモジュールおよびプレクスI/O

- AntiFuseともいう。1回書き込んだら書換できない。一発勝負。
- その代わり信頼性は高い。

FPGAラインナップ[®] その2

書換可能型

Flash Base FPGA	
PolaFire FPGA	IGLOO2 FPGA
・コスト最適化、低消費電力、ミッドレンジFPGA	・最も豊富な低密度デバイス
・250 Mbps～12.7 Gbpsトランシーバ	・5K～150Kロジックエレメント
・3100K～500Kロジックエレメント、最大33MビットのRAM	・10K LEデバイス以上でのPCIe Gen2のサポート
・クラス最高のセキュリティと優れた信頼性	・高性能メモリサブシステム
ProASIC3 FPGA	IGLOO FPGA
・最大35Kロジック素子の低消費電力デバイス	・CPLDの交換に最適な低密度FPGA
・CPLDの交換に最適	・最大35Kロジック素子の低消費電力デバイス
・Flash * Freezeテクノロジーで利用可能	・Flash * Freezeテクノロジー
・ARM対応のプロセッサコアで提供	・ARM Cortex-M1プロセッサコアで利用可能

- フラッシュベースのFPGA。電源落としても消えない。
- SRAMベースのFPGAと異なり、プログラム用外ROM不要。
- 低消費電力、高信頼性が売り。(高集積、高速、大規模というキーワードは出てこない。)

FPGAラインアップ[®] その3

SoC(CPU搭載型)

System On Chip FPGA

SmartFusion2 SoC FPGA

- ・最高の信頼性、最も安全で低消費電力のFPGA
- ・5MビットSRAMおよび4.5MビットNVMで最大150K LE
- ・命令キャッシュ付き166 MHz ARM Cortex-M3マイクロプロセッサ
- ・5Gbps SERDES、PCIe、XAUI / XGXS +ネイティブSERDES
- ・SECDED付きハード667 mbps DDR2 / 3コントローラ

SmartFusion SoC FPGA

- ・ハード32ビットARM Cortex-M3マイクロコントローラコアを搭載した業界唯一のFPGA
- ・アナログ/デジタルコンバータ（ADC）、電圧/電流/温度モニタ、デジタル/アナログコンバータ（DAC）、コンパレータ、およびアナログ演算エンジン（ACE）を備えたプログラマブルアナログ
- ・最大500Kのゲートと204のアナログおよびデジタルI/Oを備えた実績のあるProASIC3 FPGAファブリック

ARM Cortex-M3搭載

SmartFusionはADC,DAC搭載

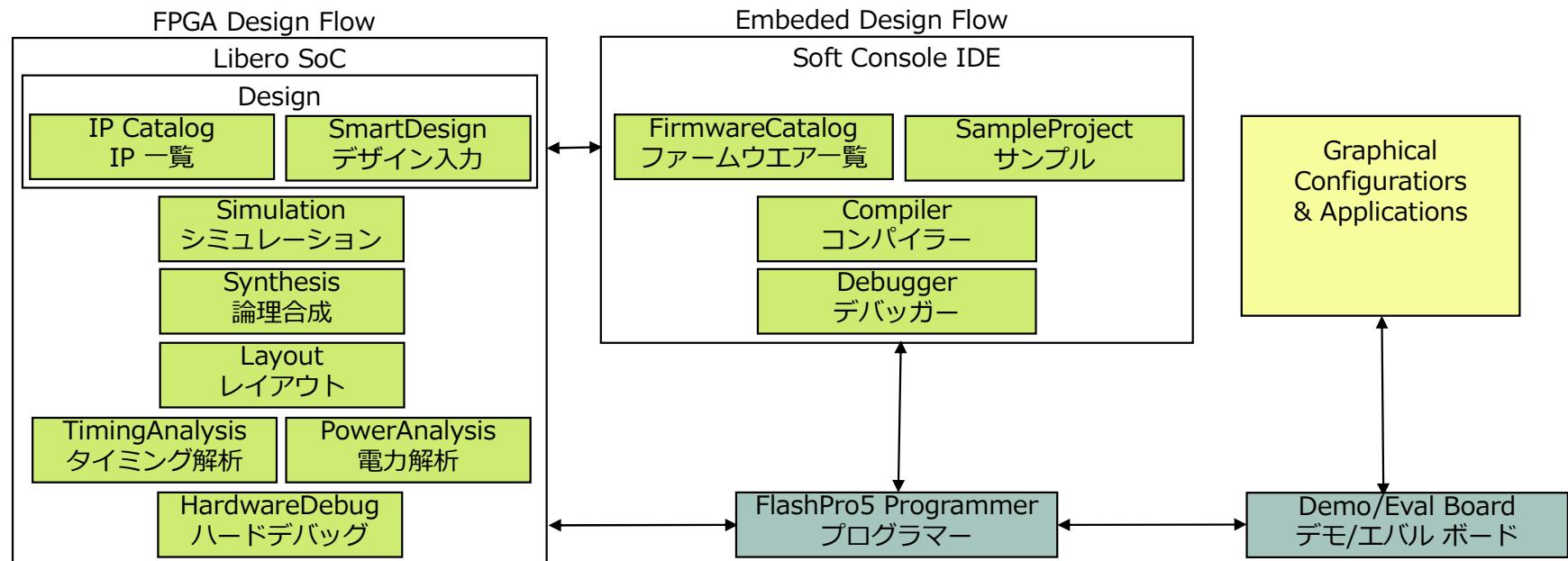
(2010年 ESC AwardとかBest Electronic Design award とか受賞)

@shibatchii

開発ツール

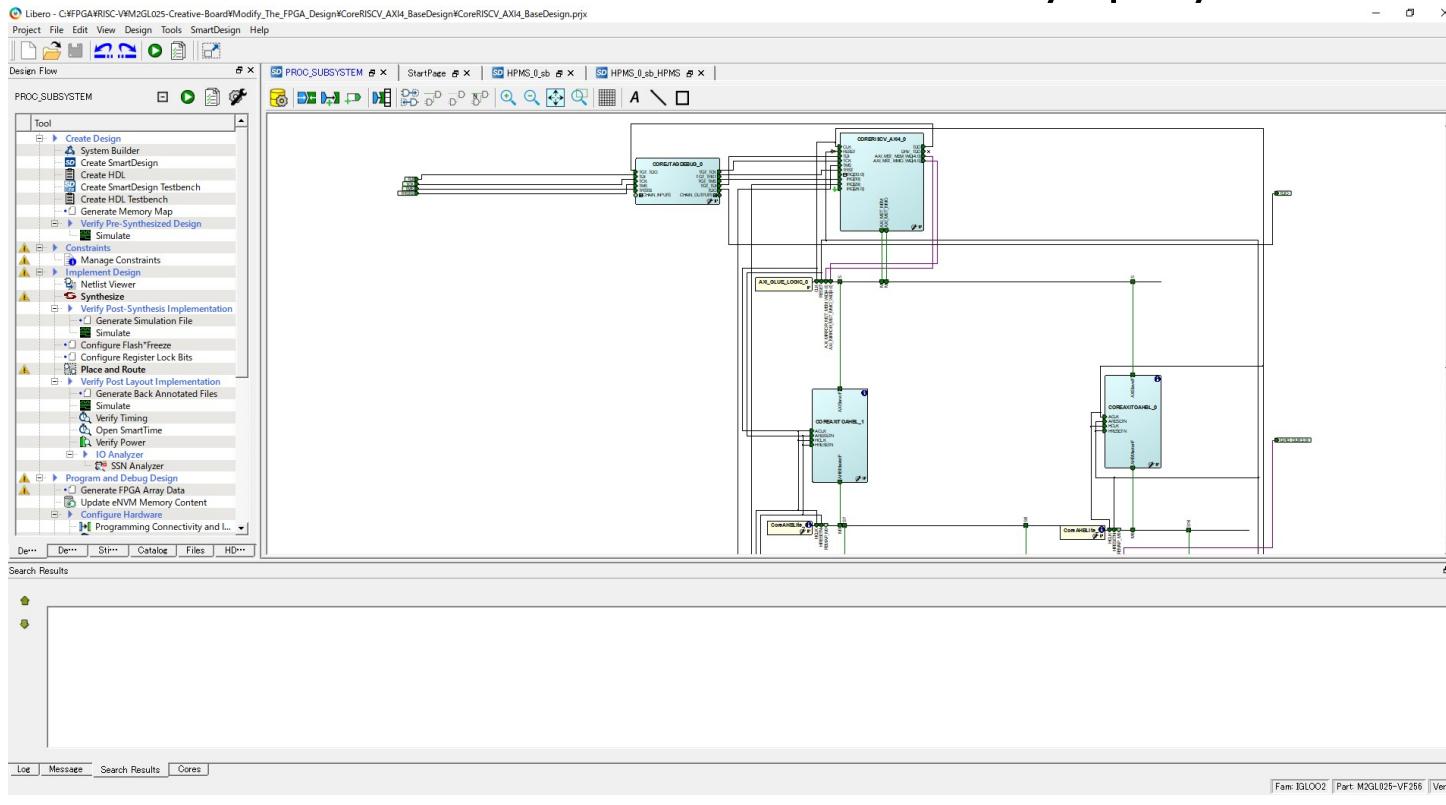
FPGA部開発用のLiberSoCとファーム開発用のSoftConsole

■ デザインフロー



開発ツール LiebroSoc

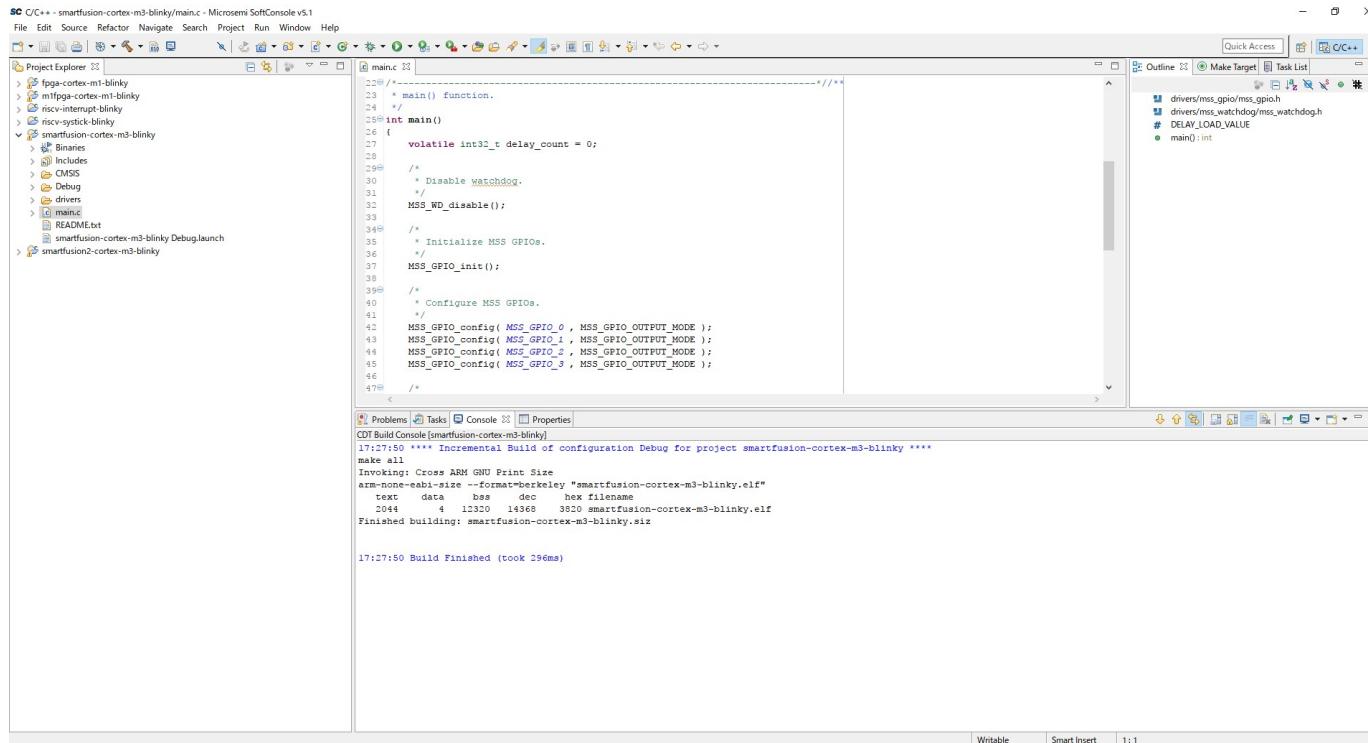
- IGLOO2, SmartFusion2, RTG4, SmartFusion, IGLOO, ProASIC3 等用のFPGA 統合開発ツール。無料版あり。Windows,Linux対応。
 - シミュレータ：Modelsim ME、合成:Synplify Pro ME



@shibatchii

開発ツール SoftConsole

- SmartFusion2とかのCortex-M3ファーム開発やIgloo2とかのRISC-Vファーム開発用開発が同じ環境で行える。Windows,Linux対応。
- 移植やツール操作の習得が2度手間にならない。
 - ベースはEclipse



@shibatchii

評価ボード

Igloo

- AGLN-NANO-KIT (Microsemi,DigiKey) \$100

Igloo2

- Creative Development Board(Future Electronics) \$99.95 ~~オススメ~~

SmartFusion

- SmartFusion Starter Kit (EmCraft) \$179
- SmartFusion Eval Kit Guide(microsemi,Digikey) \$106.25

ディスコン

Smartfusion2

- M2S-FG484 SOM Starter Kit (EmCraft) \$179
- Creative Development Board(Future Electronics) \$99.95 **オススメ**

※ Creative Development Board は品番で Igloo2 と Smartfusion2 を区別

※ 書き込み器 FlashPro5 \$50位 が必要なボード有

@shibatchii

FPGAボード



@shibatchii

マイナーなボードを買うと苦労するよ

Future Electronics

FUTUREM2GL-EVB

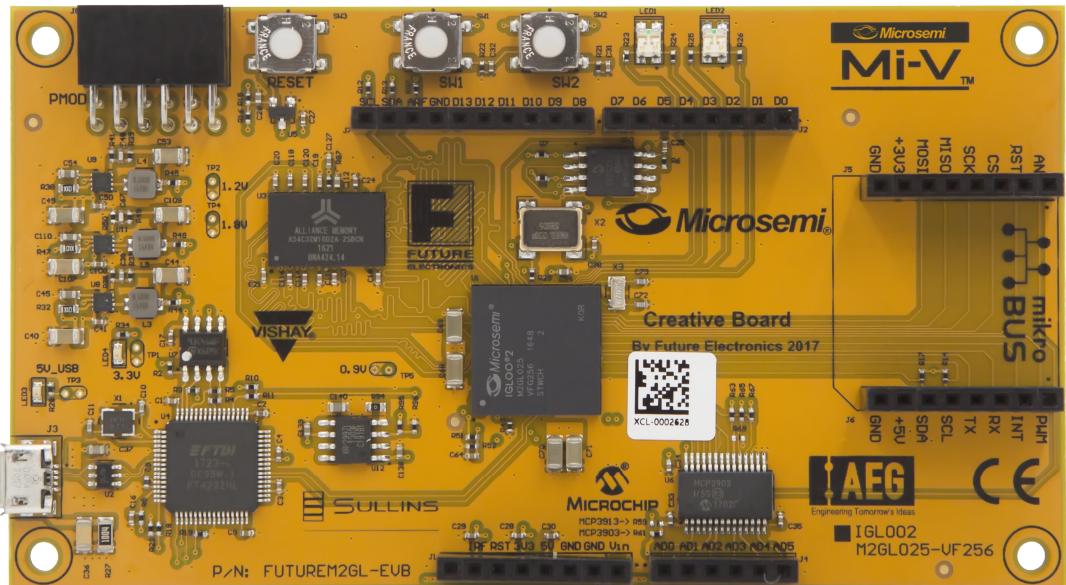
お値段 : \$99.95 約12,000円

送料 : \$50 Fedex 約6,000円

送料高 !

壊れることも想定して2枚発注

注 : クレジットカードの引き落としは「海外でのショッピング枠」
海外なんか行かんし、と限度額¥0にしてたので落とせんよ～
と連絡来了。



@shibatchii

ボード入手、だが

1週間ほどで到着。
Lチカとかサンプル動かして遊ぶ



2週間してFedexからなんか届く
何？ 「請求書」だと！！？



手数料と消費税で¥1,500払え！ えー、泣く泣く支払う。
結局 ¥12,000×2+¥6,000+¥1500 で ¥31,500位。orz

@shibatchii

Microsemi FPGAへの おさそい

■ MicroSemiFPGA ここがいい！ (他のFPGAでも同じじゃん というのは置いておく)

- 1万円台でボードが買える
- ボードを輸入するときのドキドキ感が味わえる
- 開発ツールのサイズがコンパクト。ディスクにやさしい
- ○○警察が来ない
- FPGAがあっちっちにならない
- 英語のマニュアルが読みやすい
- 他の人がやってない、先頭に立つ爽快感がある

■ MicroSemiFPGA ここがよろしくない

- 上記の裏返し
- 日本語の情報が無い。(立花エレテック社にあるようだが出してもらえない)
- ボードは個人輸入。妙な緊張感有り。(ものによってはDigiKeyやChip1Stopから買える)
- 使っている人いない。全部自分でやらなきゃ！。かまってもらえない。
- 高速、大容量を期待すると失望する

@shibatchii

Microsemi FPGA RISC-V 対応状況

@shibatchii

なぜMicrosemiでRISC-V?

- ☑ 2016年11月頃、 **RISC-V始めました**

「米Microsemi社は、 オープンソースの命令セット「RISC-V」を実装したCPUコア（RISC-Vコア）として「RV32IM」を開発し、 同社のFPGA向けに提供を始める。このCPUコアは、 米SiFive社と共同開発したもの。」

というようなニュースが流れた。

- ☑ しばらくするとGitHubにFPGAのプロジェクトファイル等が提供された。
- ☑ おもしろそうだ、 やってみるか！
- ☑ ちなみに、 MicrosemiはRISC-VのGoldメンバー なのですが無くなることもないだろうと。

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

- ☑ 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。

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

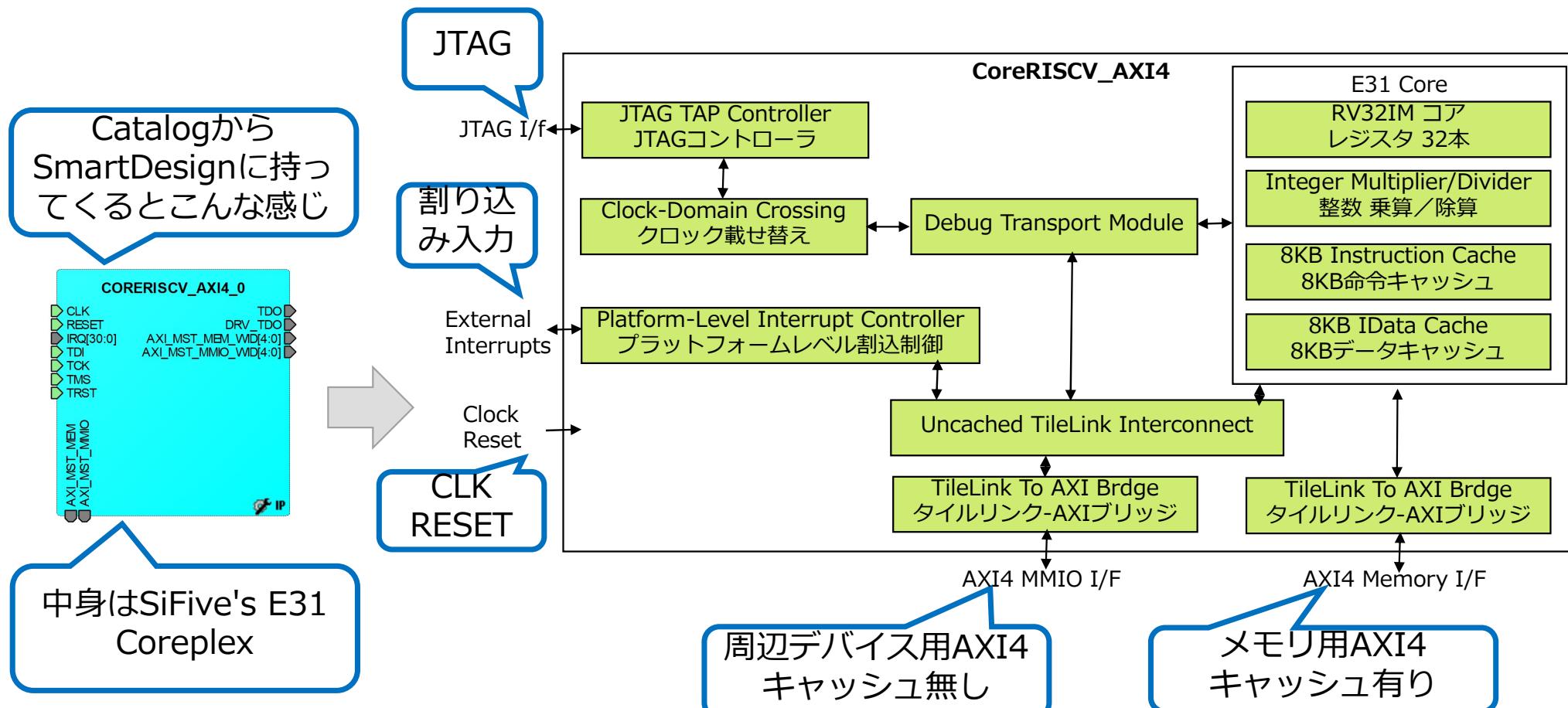
■ RV32I:基本命令 ■ M:乗除算命令 ■ A:アトミック命令 ■ C:圧縮
■ F:単精度小数点 ■ D:倍精度小数点 ■ Q:4倍精度小数点 ■ 他にも L,B,J,T,P,V,Nなど有

ここまで必要ないなら、例えばRV32IMACだけでもOK

MicrosemiのRISC-Vコア仕様

- 低消費電力のASICマイクロコントローラおよびFPGAソフトコアのインプリメンテーション用に設計・統合された8Kバイトの命令キャッシュと8Kバイトのデータキャッシュ
- プラットフォームレベルの割り込みコントローラ(PLIC)は単一優先度レベル
- RISCV標準のRV32IM ISAをサポート・JTAGインターフェースを備えたオンチップデバッグユニット
- IOおよびメモリ用の2つの外部AXIインターフェイス

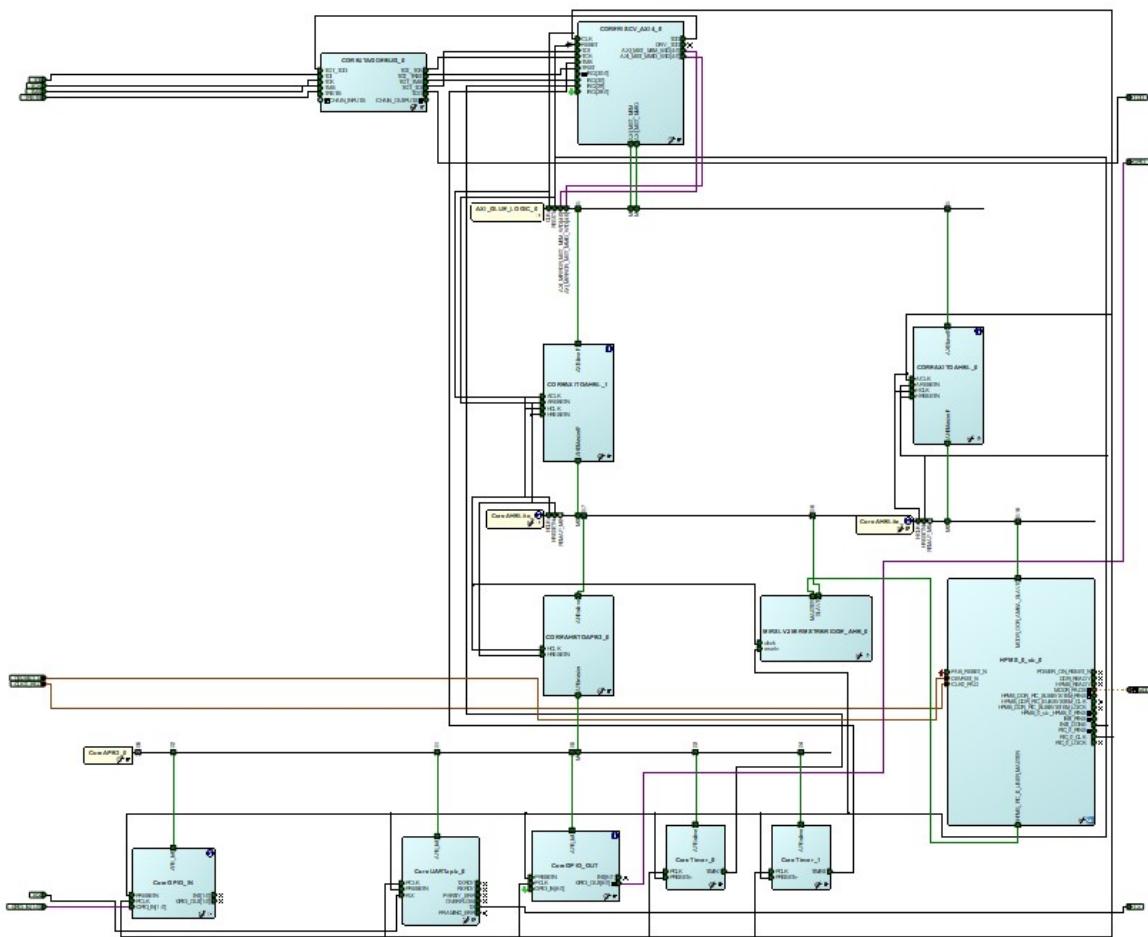
Microsemiはどう実装したか



@shibatchii

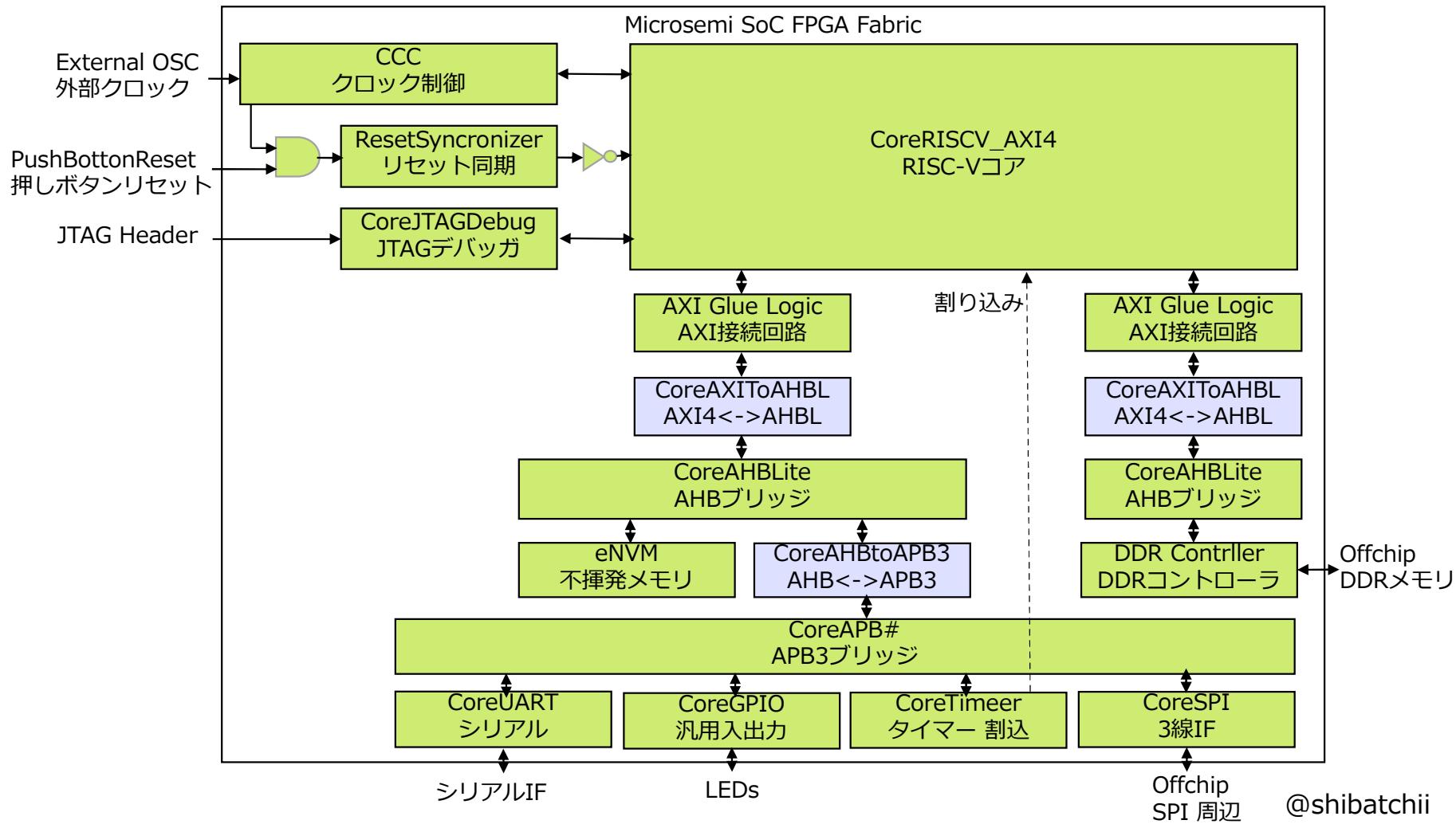
Lチカ用全体構成

Microsemiのサンプル



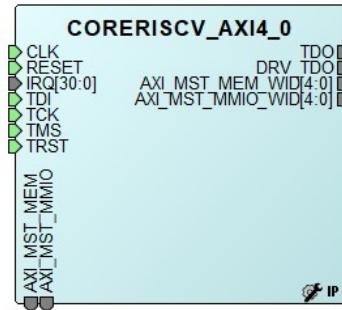
@shibatchii

全体構成 ブロック図

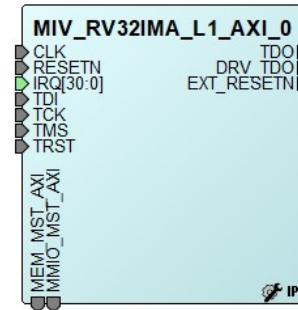


バリエーション

- さすがにAXI x 2の構成はゴージャス過ぎ
 - AHBx2バス化
- 命令セットを充実して高性能化
 - A:Atomic命令、F:単精度浮動小数点命令



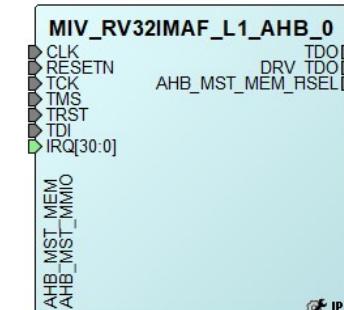
RV32IM
AXI



RV32IMA
AXI



RV32IMA
AHB



RV32IMAF
AHB

ソフトウェア構成 1

riscv-systick-blinky

```
.cproject      // SoftConsole用プロジェクトファイル Cの環境に関するもの
.project       // SoftConsole用プロジェクトファイル Softconsole自身の環境に関するもの
hw_platform.h // ベースアドレスや定数がdefineされている
main.c         // Cプログラムメイン。ここから動き始める。解析もここから。
README.txt     // このプログラムの概要。結構重要なことが書いてある。
riscv-systick-blinky Debug.launch // デバッグ時の設定ファイル? よくわからない

drivers        // 各ハードIP用のドライバファイル
  └─CoreGPIO    // GPIO 汎用入出力用
    coregpio_regs.h // GPIOが持っているレジスタのオフセットアドレス
    core_gpio.c   // GPIOへ出力したり、入力したりする関数
    core_gpio.h   // ヘッダファイル。ビットマスク等の設定値defineとprototype宣言

  └─CoreUARTapb // UART シリアル用
    coreuartapb_regs.h // UARTが持っているレジスタのオフセットアドレス
    core_uart_apb.c   // UARTへ出力したり、入力したりする関数
    core_uart_apb.h   // ヘッダファイル。ビットマスク等の設定値defineとprototype宣言
```

@shibatchii

ソフトウェア構成 2

```
—hal          // Hardware Abstraction Layer ハードウェア抽象化階層
  cpu_types.h // typedef 数個
  hal.h        // レジスタアクセスとかをHW依存のから共通のアクセスに変換
  hal_assert.h // HAL_ASSERT がdefineしてある
  hal_irq.c   // 割り込みの抽象化
  hw_macros.h // ハードウェアレジスタアクセス define
  hw_reg_access.c // レジスタアクセス関数
  hw_reg_access.h // レジスタアクセス関数のプロトタイプ宣言

—riscv_hal    // RISC-V用 Hardware Abstraction Layer ハードウェア抽象化階層
  encoding.h   // ステータスや割り込みレジスタのビット位置define等
  entry.S       // csrアクセスアセンブラー
  init.c        // 領域コピー関数
  microsemi-riscv-ram.ld // SRAMのメモリスペース設定
  riscv_CoreplexE31.h // 割り込み制御、設定
  riscv_hal.c   // 割り込みハンドラ
  riscv_hal.h   // 割り込みハンドラのプロトタイプ宣言
  riscv_hal_stubs.c // 該当関数が定義されていないときのスタブ関数
  sample_hw_platform.h // ベースアドレスや定数設定
  syscall.c     // UART_APBアクセス関数
```

プログラム抜粋 (main.c)

```
// riscvの抽象化階層、必要な定数等の定義、関数ヘッダ 呼び出し
#include "riscv_hal.h"
#include "hw_platform.h"
#include "core_gpio.h"
// GPIOでin,outの変数宣言。
// core_gpio.hで宣言
gpio_instance_t g_gpio_in; gpio_instance_t g_gpio_out;
// ステートカウンタ宣言。初期値は1
uint32_t g_state = 1;

// 割り込みハンドラ関数。
void SysTick_Handler(void) {
    uint32_t stable;
    uint32_t gpout;
    // GPIOのデータを読み出す
    stable = GPIO_get_inputs(&g_gpio_in);
    // 反転して、マスクをかけている。
    gpout = ~stable & 0x000000F0;
    // g_stateを右1ビットシフト。つまり2倍にしている
    g_state = g_state << 1;
    // 4より大きくなったら1にする。
    if (g_state > 4) {
        g_state = 0x01;
    }
    // [7:4]がそのままもちこし、[3:0]がステートカウント値
    gpout = gpout | g_state;
    // GPIOに出力
    GPIO_set_outputs(&g_gpio_out, gpout);
}
```

```
// おなじみ、Cのメイン
int main(int argc, char **argv) {

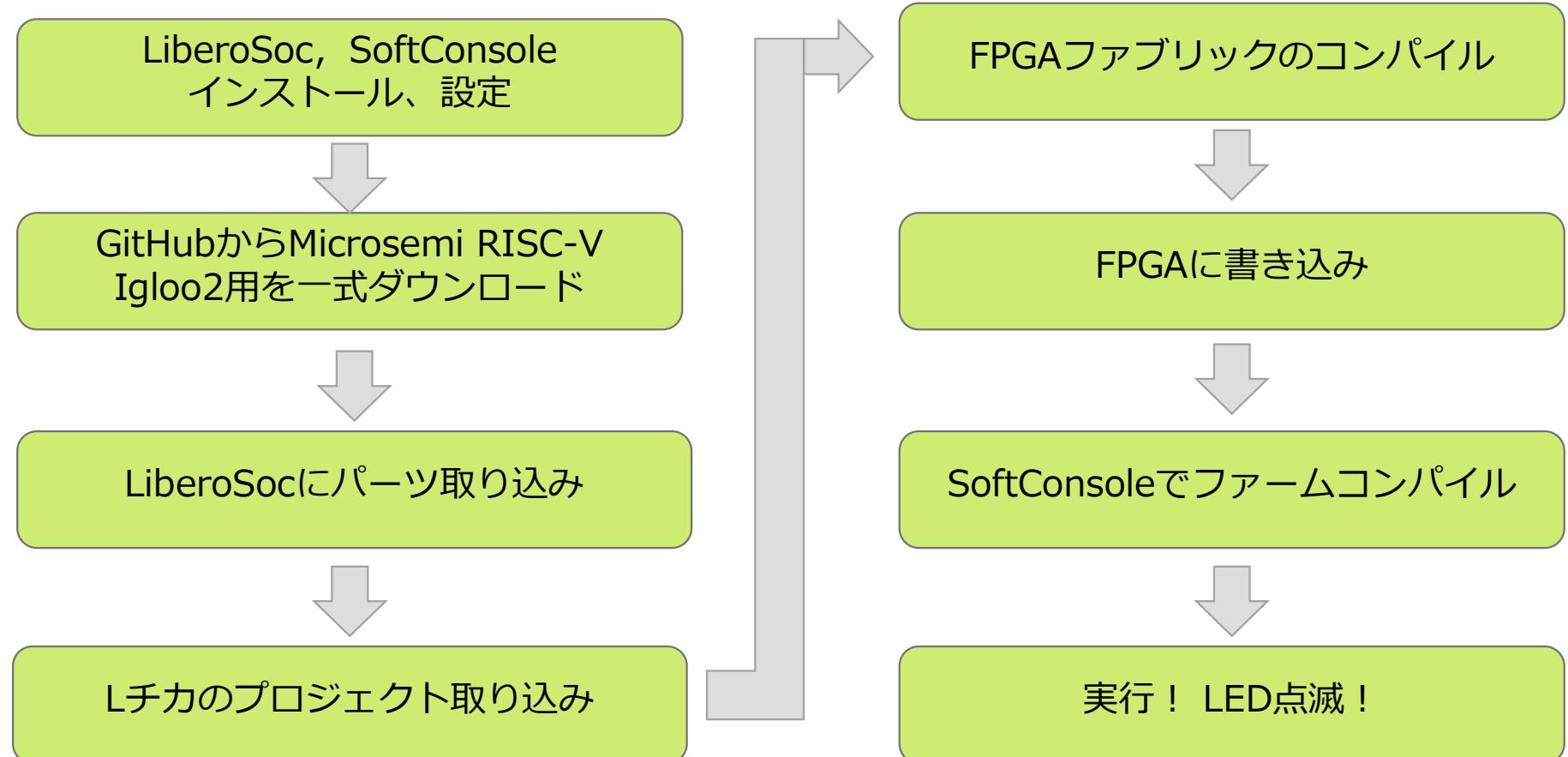
    // PLICって Platform Level Interrupt Controller の略
    // この関数は riscv_CoreplexE31.hで定義してある
    PLIC_init();

    // GPIOの初期化。ここで入力でつかうとかの設定。INとOUTでアドレスが違う
    // core_gpio.hで宣言
    // COREGPIO_IN_BASE_ADDR とかは hw_platform.h で定義
    GPIO_init(&g_gpio_in, COREGPIO_IN_BASE_ADDR, GPIO_APB_32_BITS_BUS);
    GPIO_init(&g_gpio_out, COREGPIO_OUT_BASE_ADDR, GPIO_APB_32_BITS_BUS);

    // タイマー設定 riscv_hal.h で定義されている。
    // SYS_CLK_FREQ は hw_platform.h で定義。83000000UL なので 83MHz
    SysTick_Config(SYS_CLK_FREQ / 2);

    return 0;
}
```

動かしてみよう



まとめ

- ☑ Microsemiは何を考えているか。（@shibatchii主観）
 - ▶ いままでは、IglooにCortex-M3載せたSmartfusionやSmartfusion2でCortex-M3、そこからCPU除けたIgloo2などをリリースしてきた。
 - ▶ だが最新のPolaFireではCPU入りのはでてきなさそう。
 - ▶ XilinxのMicrobreeze, IntelのNIOS IIのようなメインラインのCPUを持ってない。
 - ▶ ハードCPUを載せるのはちょっと荷が重い。
 - ▶ そうだ！ RISC-Vで自由に構成してもらおう！
 - ▶ ベンダーもユーザーもWin&Win！
- ☑ 今後予定
 - ▶ RISC-Vをさらに調査しオリジナルなものを作ってみる。（最小構成のRISC-Vとか）

RISC-Vマニュアル日本語訳のポイント

突然ですが、アンケート

- RISC-VのISAマニュアル読んだことある？
 - 読んだ。しっかり理解できている
 - 読んでみたけど、なんかよく分からない
 - 読んでない

きっかけ

ふと思った。RISC-V動いた～とかやっているけど、
Cでプログラム書いていたら
“Cortex-M3でもRISC-Vでもほとんど変わらない”

HAL(Hardware Abstraction Layer)やデバイスドライバが良くて差を吸収している。

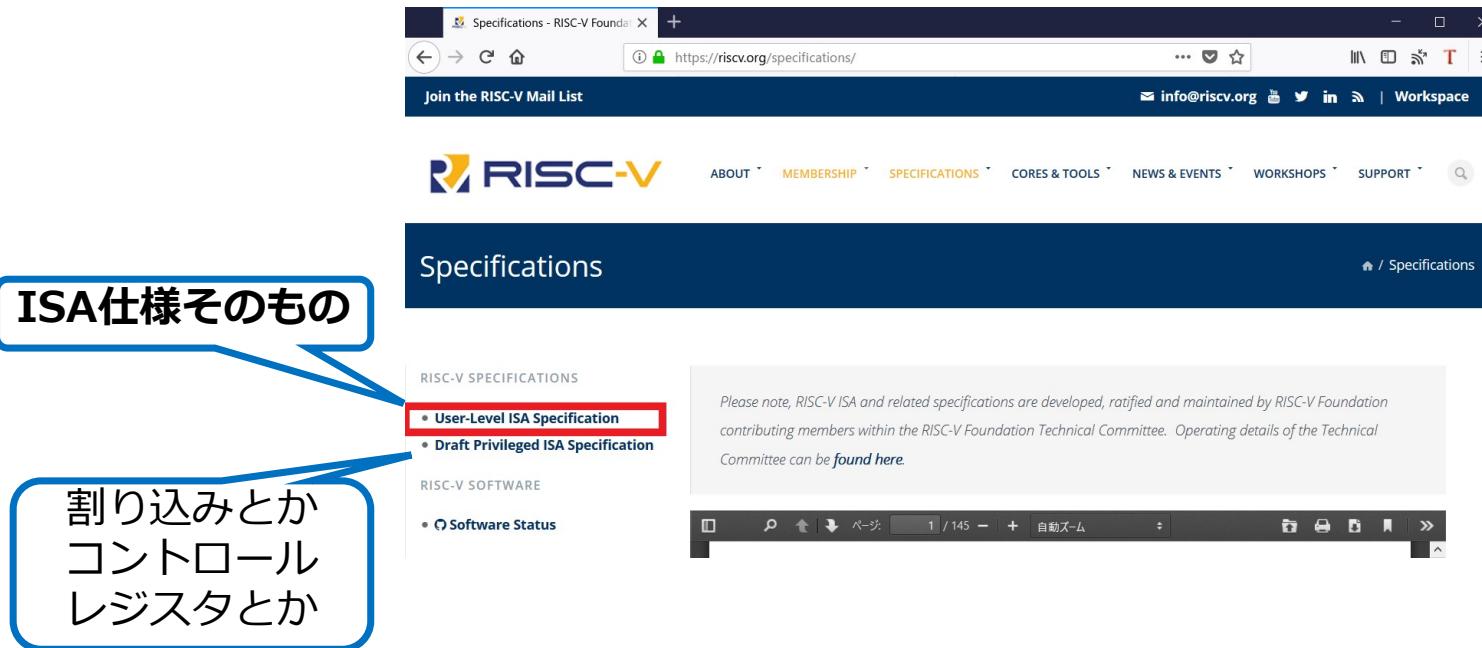
これでええんか？

よくない。全然RISC-Vっぽく無い！

＼ツカッテルッテ イエナイダロ／
@shibatchii

どうする

もっとRISC-Vの中身を知ろう！
そのためには原点に戻り、RISC-Vの仕様書を読んでみよう。<https://riscv.org/specifications/>



@shibatchii

中をみてみると。 . .

riscv-spec-v2.2.pdf

This chapter describes version 2.0 of the RV32I base integer instruction set. Much of the commentary also applies to the RV64I variant.

RV32I was designed to be sufficient to form a compiler target and to support modern operating system environments. The ISA was also designed to reduce the hardware required in a minimal implementation. RV32I contains 47 unique instructions, though a simple implementation might cover the eight SCALL/SBREAK/CSRR instructions with a single SYSTEM hardware instruction that always traps and might be able to implement the FENCE and FENCE.I instructions as NOPs, reducing hardware instruction count to 38 total. RV32I can emulate almost any other ISA extension (except the A extension, which requires additional hardware support for atomicity).*

英語だよ！ 超苦手(泣)

@shibatchii

どうしよう (*'д`)

毎回、訳しながら読むと毎回辞書や翻訳ソフト使う事になってめんどう。

そうだ、Google翻訳でPDF一括指定して全部訳せばいいじゃん。簡単。**ちょろいぜ！**



@shibatchii

やってみた

- ・ レイアウトがぐちゃぐちゃ。
 - ・ 表は枠線までは考慮してくれない。中身だけになる。
 - ・ 図は無視される。置いてくれない。
- ・ 翻訳がまともにできていない。おかしい！？
 - ・ かかりうけが合ってない。意味わからない。
 - ・ Google翻訳って結構賢いはずなのに。(泣)
- ・ 例)
Vハーツと外部デバイスまたはコプロセッサの両方をサポートします。デバイス入力 (I) 、デバイス出力 (O) 、メモリリード (R) 、およびメモライト (W) は、任意の組み合わせ同じの。非公式には、他のRISC-Vハートまたは外部デバイスが操作を観察することはできません
フェンスに続く後続セットでは、前のセットでの操作の前に
フェンス。実行環境は、どのI / O操作が可能であるかを定義し、特に、
どのロード命令およびストア命令は、デバイス入力およびデバイス出力として扱われ、順序付けられてもよい
メモリの読み出しと書き込みではなく、たとえば、メモリマップドI / O
デバイスは通常、IおよびOを使用して注文されたキャッシュされていないロードおよびストアでアクセスされます
RビットとWビットではなく、命令セット拡張は、新しいコプロセッサ
フェンスのIとOビットを使用して順序付けられるI / O命令。

この時点でやめときやよかったです。

@shibatchii

結局どうした

先頭から145頁愚直に訳すことにした。

1. 1センテンス～1段落読んでこんな意味かなあと当たりをつける。
2. エディターにコピペ。
3. 中途半端な改行をピリオド(.)のところで改行するように成形。
4. ff,fiなどが消えているので文を読みつつ追加。(どうもff,fiとかが落ちる)
例) o set -> offset oat -> float de ne -> define
5. Google翻訳、Bing翻訳にコピペし訳す。(最初は色々な翻訳を試した。単語はweblioが良い)
6. Google翻訳からdocにコピペする。
7. 常体になっているところを敬体に直す。(混在している)
8. Bing翻訳見ながら文章確認。日本語的におかしいところを直す。
9. どうしてもおかしいところは文節を切りなおしてもう一度。
10. 以上を繰り返し。
※ どうしてもわからないところは朱書きでコメント。

@shibatchii

例

- まずはエディタにコピペ

All branch instructions use the B-type instruction format. The 12-bit B-immediate encodes signed offsets in multiples of 2, and is added to the current pc to give the target address. The conditional branch range is ±4 KiB.

sets in multiples of 2, and is added to the current pc to give the target address. The conditional branch range is _x0006_4 KiB.

- offsetを正しく。改行をピリオド位置で改行。±の文字化けを一旦削除。

All branch instructions use the B-type instruction format.

The 12-bit B-immediate encodes signed **offsets** in multiples of 2, and is added to the current pc to give the target address.

The conditional branch range is 4 KiB.

- Google翻訳にかけコピペ。

すべての分岐命令は、B型命令フォーマットを使用します。

12ビットのB-immediateは符号付きオフセットを2の倍数で符号化し、現在のpcに加算してターゲットアドレスを与える。

条件付き分岐範囲は4KiBです。

- 常体を敬体へ。訳の不足している部分を調整。±を付加。

すべての分岐命令は、B型命令フォーマットを使用します。

12ビットのB-**即値**は符号付きオフセットを2の倍数で符号化し、現在のpcに加算してターゲットアドレスを**与えます**。

条件付き分岐範囲は**±4KiB**です。

Google翻訳、(Bing翻訳)のクセ

- ・ ただの改行を文の区切りと判断する。(Google)
- ・ 常体と敬体が混在する。
- ・ 文章が長くなると精度が落ちる。 200文字超えると悪くなる。
 - まともに訳さない。述語がなくなる等。
 - 文章を勝手に省略する。ここの一文どこいった？
 - 全くそこに使われていない単語を持ってくる。
 - 否定形に弱い。時々全く逆の意味になる。
 - 対策：カンマの所で改行、文の区切りを考えて(andとかwhereとか)の所で改行して翻訳にかける。

Manual 翻訳、理解のポイント

- このマニュアルはデータシートとかユーザーズマニュアルとして読んでは(とりかかっては)いけない。
- 論文とか学術書とか歴史書の雰囲気がある。
- 特にライン引いて字下げしてある箇所は検討した経緯とか、歴史とか、他のISAのディスリとか、感想とかが書いてあり、かつ1文が非常に長い。心が折れるのでまずは読み飛ばしてよし。1章も同様。

ここを重点的に

The JAL and JALR instructions will generate a misaligned instruction fetch exception if the target address is not aligned to a four-byte boundary.

こちらは後回し

Instruction fetch misaligned exceptions are not possible on machines that support extensions with 16-bit aligned instructions, such as the compressed instruction set extension, C.

まずは読むべき箇所

- 第2章 RV32I 基本整数命令セット “I”
 - どのRISC-Vにも必ず入っているので重要
- 第19章 RV32/64G命令セットリスト
 - 一覧形式になっておりビット配列がわかりやすい
- 第20章 アセンブリ プログラマーズ ハンドブック
 - どういう命令になるかはここが参考になる
 - アセンブラーに慣れている人は心が落ち着く
- 次は“M”, “A”, “F”, “D” ただ“A”はちょっと難しい

日本語訳しても著作権とか大丈夫？

- ・ マニュアルに以下の文あり。
- ・ 「This document is released under a Creative Commons Attribution 4.0 International License.」
- ・ つまり、CC BY 4.0 なので「適切なクレジットを表示すればOK。あとは自由に」という事。
- ・ GitHub、日本語訳にクレジットを表示した。
- ・ 日本語訳も踏襲して、CC BY 4.0

日本語訳の場所

- GitHub
 - <https://github.com/shibatchii/RISC-V>
 - RISC-V_spec_manual_v2.2_jp.odt (Libre Office)
 - RISC-V_spec_manual_v2.2_jp.pdf (Acrobat)
 - RISC-V_spec_manual_v2.2_jp.docx (MS Office)

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（アセンブラー疑似命令 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

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>
- プログラマのためのFPGAによるRISC-Vマイコンの作り方
堀江 徹也さん Kindle
 - 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
 - <http://msyksphinz.hatenablog.com/>

今後の予定

- ・ 訳してみたもののまだまだ意味不明な個所や日本語がおかしい箇所があるのでさらにブラッシュアップ。協力者絶賛募集中。
- ・ Volume II: Privileged Architecture の訳
 - ・ こちらを先に行う予定

質問タイム

なにか質問あればどうぞ。
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。(^^\)/