Trinita 1stage Core

注意

- **本ソースコードの使用にあたり ライセンス に同意頂く必要がございます。**
  - **本リポジトリの一部または全部を Clone / Download した時点で、ライセンス条項に同意したものとみなします。**
- **本ドキュメントの pdf 版は こちら です。**
- **ソフトウェアを Flash Memory に格納し、Bootloader を使用して起動する方法は下記ドキュメントを参照してください。**
  - **Bootloader によるソフトウェアの起動**

概要

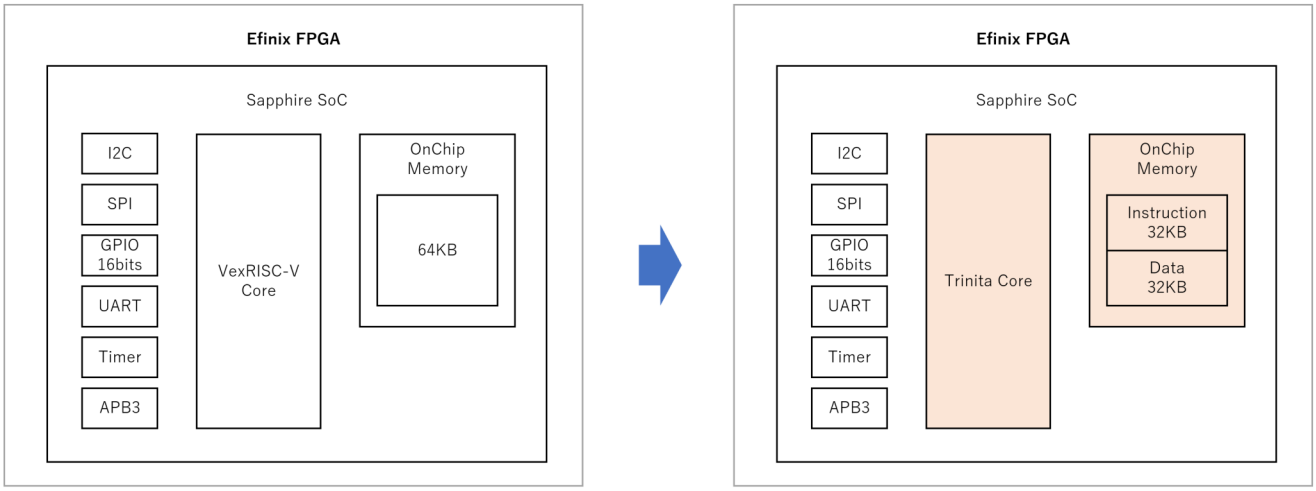
このコアは Efinix FPGA Sapphire SoC 向けの 1ステージ RISC-V コアです。

※2024年3月現在 1 ステージコアの性能向上にともない、2 ステージコアは提供終了となりました。

Efinix Sapphire SoC の Vex RISC-V コアを差し替えて使います。

Efinity の コンパイルパラメータ (VerilogHDL マクロ) 定義によって、下記のオプションを使用できます。

- CPU レジスタ実装方法 : Block RAM または FF の選択



動作環境

ハードウェア

- Efinix 社 Trion FPGA または Titanium FPGA

※後述の Example Design では Trion シリーズ T8 / T20, Titanium シリーズ Ti60 だけに言及していますが、同シリーズの他のデバイスにも実装可能です。

ソフトウェア

- Efinix 社 Efinity IDE 2023.2~

- Efinix 社 RISC-V IDE 2023.1~
- Python 実行環境

サンプル

Trinita Core の性能評価用として、各種評価ボード向けの Example Design をご用意しました。

下記フォルダのデザインをダウンロードしてお使いください。

評価ボードですぐに動作確認できる、コンパイル済みのバイナリ (.hex) も同梱しています。

- T8 Xyloni 用 Example Design
- T20 BGA256 Development Board 用 Example Design
- Ti60 F225 Development Board 用 Example Design

制約事項

項目	内容
動作時間	無償評価版のみ 1 時間
動作周波数	Trion T8 : 10 MHz Trion T20 : 25 MHz Titanium シリーズ : 75 MHz
オンチップメモリ容量	Trion T8 : 8KB (imem 4KB + dmem 4KB) Trion T20 : 64KB (imem 32KB + dmem 32KB) Titanium シリーズ : 64KB (imem 32KB + dmem 32KB)
メモリ先頭アドレス	imem : 0xF900_0000 dmem : 0xF908_0000

性能参考値

※ Titanium Ti60 Development Board による測定結果
※ 動作周波数は 75 MHz
※ オンチップメモリ容量は 64KB (imem 32KB + dmem 32KB)

Efinix Sapphire SoC の動作周波数は 20 ~ 400MHz ですが、Trinita Core は実行効率が向上しているため動作周波数を下げています。

動作周波数やメモリ容量のカスタマイズはご相談ください。

DMIPS/MHz

※ 当社比の数値です。実装条件によって値は変動します。

コア	キャッシュ	DMIPS/MHz
Efinix VexRiscv	無し	1.08
Efinix VexRiscv	有り	1.34
Uno Lab Trinita 1stage CPUレジスタBRAM	無し	1.79

Efinix FPGA リソース使用量

※ 当社比の数値です。実装条件によって値は変動します。

※ 下表は CPU コアのみの数値であり、実際に使う場合は IMEM, DMEM, Peripheral 等の使用量が追加されます。

※ Ti60 におけるリソース使用量です

コア	キャッシュ	CPUレジスタ	FFs	LTUs	RAMs
Efinix VexRiscv	無し	FF	1180	1954	4
Efinix VexRiscv	有り	FF	1502	2382	16
Uno Lab. Trinita	無し	BRAM	685	2219	4
Uno Lab. Trinita	無し	FF	1675	3803	0

Trinita 1stage Core 実装 (差し替え) 手順

※ この手順は、下記の前提条件で説明を進めます。

- 既存デザインに Sapphire SoC が実装済みである
- Sapphire SoC のインスタンス名が sap である

※ 置き換え手順は YouTube でも公開しています。あわせて参照下さい。

1. テンプレートをコピーする

- template フォルダに格納されているファイル・フォルダを、プロジェクトフォルダにコピーします。
- Sapphire SoC のインスタンス名が sap 以外である場合、templete/embedded_sw/sap 配下のフォルダを、お使いのインスタンス名のフォルダ配下にコピーしてください。

2. Sapphire SoC ソースコードの VexRiscV コアの Trinita コアに置き換える

1. templete
2. ./ip/sap フォルダの sap.v を ./convtrinita フォルダにコピーします。
3. コマンドプロンプトを開き ./convtrinita フォルダに移動します。
4. 下記コマンドを実行します。このコマンドによって、sap.v の VexRiscV コアが Trinita コアに置き換わります。

```
python sap2tri.py sap.v
```

5. sap.v を ./ip/sap フォルダにコピーします。(上書き)

3. トップデザインにクロックを追加する

1. Efinity を起動して、プロジェクトを開きます。

- Trinita コアは、クロックを 3 本使用します。
    - io_systemClk : メインクロック
    - io_systemClk2 : メインクロックと同周期の位相をシフトしたクロック
    - io_systemClk3 : メインクロックと同周期の位相をシフトしたクロック
- 位相のシフト順序が io_systemClk → io_systemClk3 → io_systemClk2 となるように、PLL の設定を行います。
    - Trion では PLL の位相を細かく設定できないため、Example Design のように VerilogHDL による not 反転記述でクロックを生成しています。
    - Efinix の FF や RAM 等の Primitive には極性 (Polarity) 反転機能が実装されているため、クロックの not 反転記述を使うと、それが有効になります。
        - そのため、not 反転記述を使ってもクロックスキューは増えません。

2. トップデザインの入力ポートとして io_systemClk2, io_systemClk3 を追加します。
3. SoC のインスタンスに io_systemClk2, io_systemClk3 を接続します。
4. Efinity Interface Designer を開き、PLL に io_systemClk2, io_systemClk3 を追加します。

- io_systemClk2, io_systemClk3 の推奨位相は下記のとおりです。
    - Trion T20 25 MHz
        - io_systemClk : 0 deg
        - io_systemClk2 : 270 deg (io_systemClk3 の反転クロック)
        - io_systemClk3 : 90 deg
    - Titanium Ti60 75MHz
        - io_systemClk : 0 deg
        - io_systemClk2 : 225 deg
        - io_systemClk3 : 90 deg

5. constraint.sdc に io_systemClk2, io_systemClk3 の定義を追加します。

- お使いのデザイン規模・動作周波数に合わせて、io_systemClk - io_systemClk3 - io_systemClk2 間のデータ受け渡しタイミングがミートするように位相を決定します。
- 何度かコンパイルし、timing report (slack) をチェックして、位相を決定します。

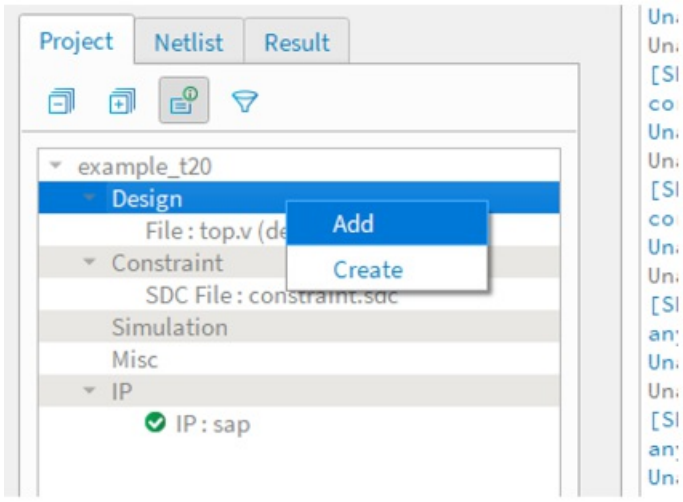
4. マクロ定義ファイル(trinita_define.vh) を作成する

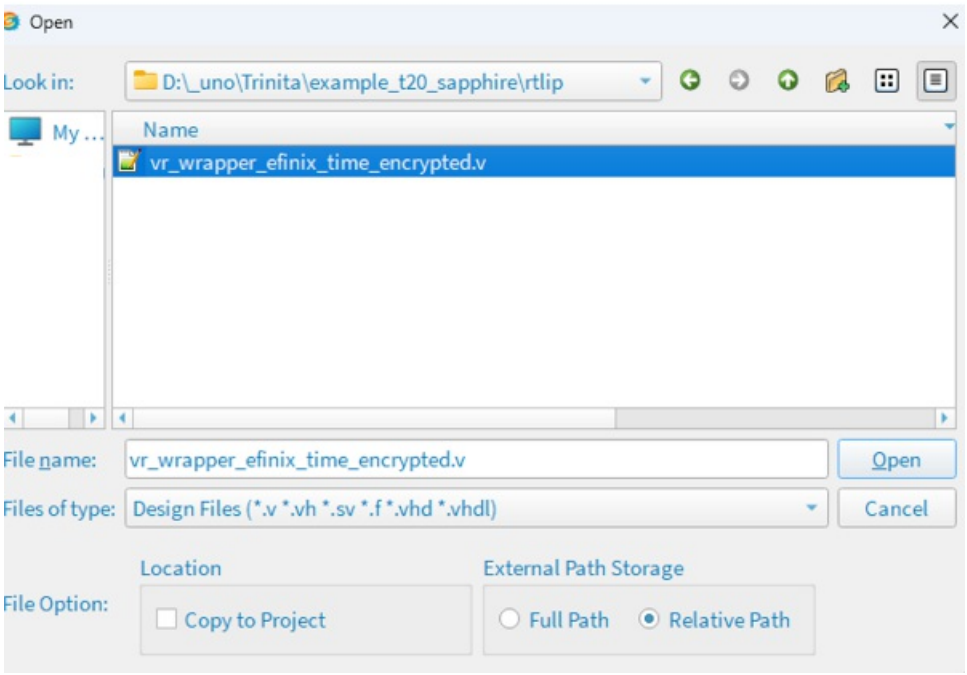
1. trinita_define.vh をエディタで開き、下記の通りマクロを定義します。

```
`define EFINIX 1
`define FREQ 25
`define SAPPHIRE 1
`define START_ADDRESS 32'hF9000000
`define IMEM_AWIDTH 15
`define DMEM_AWIDTH 15
`define FILE_IMEM  "./romdata/imem.hex"
`define FILE_IMEM0 "./romdata/imem0.hex"
`define FILE_IMEM1 "./romdata/imem1.hex"
`define FILE_IMEM2 "./romdata/imem2.hex"
`define FILE_IMEM3 "./romdata/imem3.hex"
`define FILE_DMEM  "./romdata/dmem.hex"
`define FILE_DMEM0 "./romdata/dmem0.hex"
`define FILE_DMEM1 "./romdata/dmem1.hex"
`define FILE_DMEM2 "./romdata/dmem2.hex"
`define FILE_DMEM3 "./romdata/dmem3.hex"

//デバイスの選択。どちらかを有効にする。
`define TRION 1
//`define TITANIUM 1
```

2. プロジェクトに ./rtlip/vr_wrapper_efinix_time_encrypted.v を追加します。



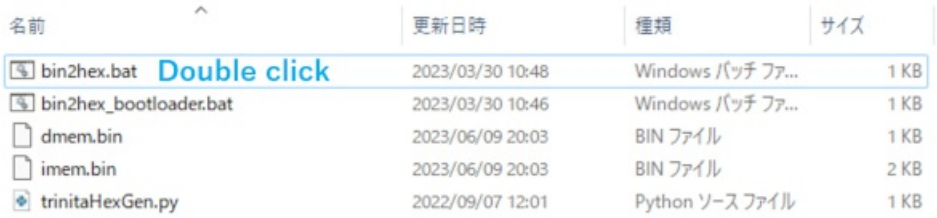


3. トップデザインに下記の 1 行を追加します。

```
`include "trinita_define.vh"
```

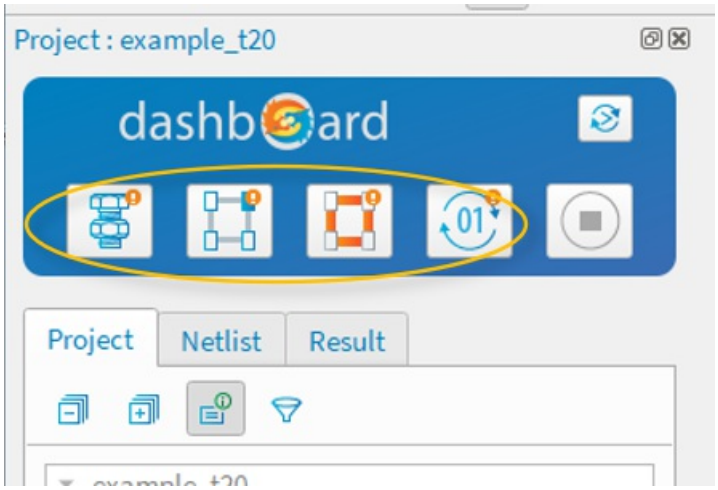
5. RISC-V IDE でソフトウェアを Trinita 用にビルドする

1. RISC-V IDE を起動したら、ワークスペースとして ./embedded_sw/sap を指定します。
2. gpioDemo_trinita プロジェクトをインポートします。
3. gpioDemo_trinita/build 配下に imem.bin と dmem.bin が出力されます。
4. imem.bin と dmem.bin を ./romdata にコピーします。
5. コマンドプロンプトを開き、./romdata に移動します。
6. bin2hex.bat をダブルクリックして実行します。



6. Efinity でコンパイルする

Efinity でコンパイルを実行します。



問い合わせ先

	リンク先
Trinita IP コア 開発元・技術問い合わせ	株式会社ウーノラボ
Efinix FPGA / 評価ボードのオンライン購入	コアスタッフ Efinix製品ページ
Efinix FPGA 取扱代理店	加賀デバイス株式会社