Efinity Debugger
(Logic Analyzer編)

株式会社レスターエレクトロニクス

EfinityのDebugger

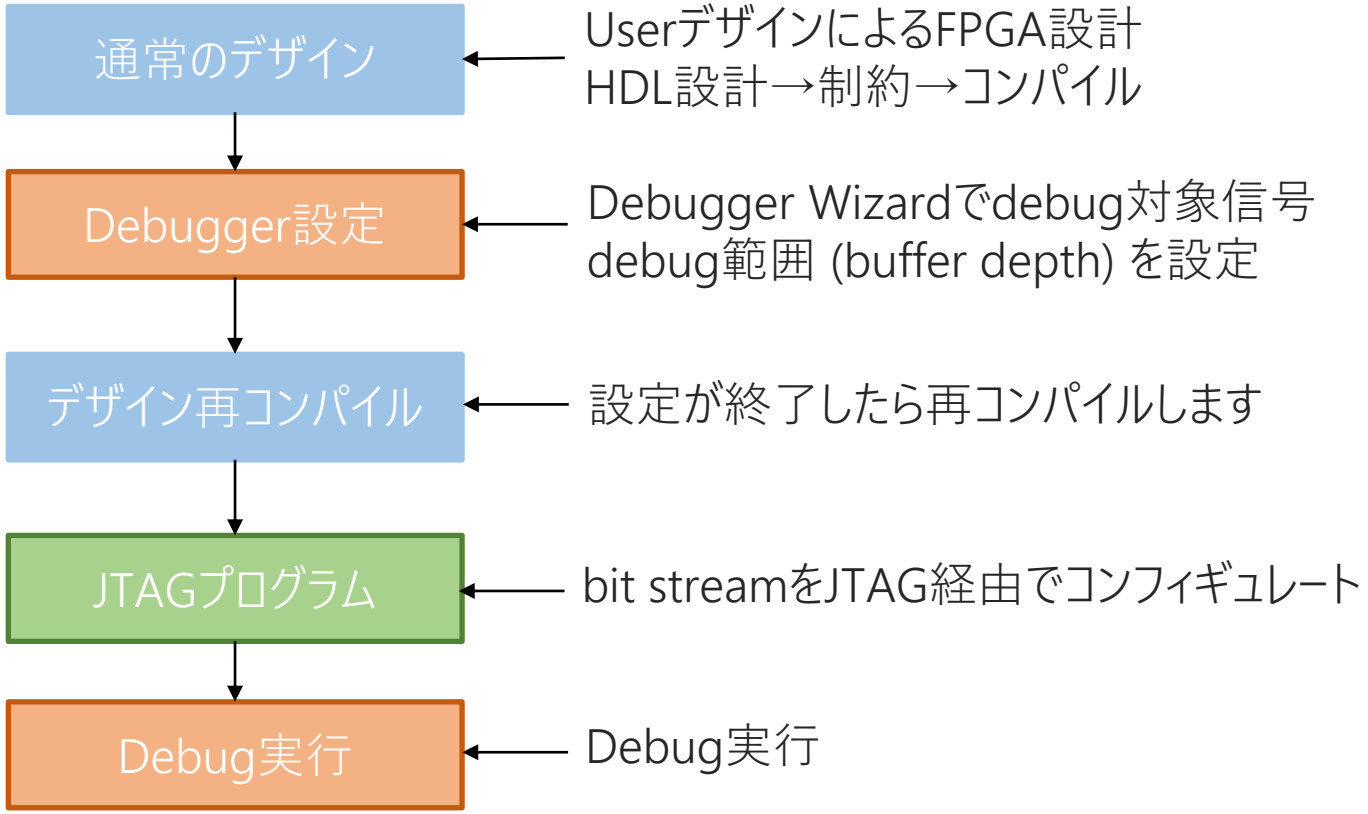
Efinityにはハードウエアデバッガの機能が実装されており、FPGA内部の信号の状態を確認できます。
モニタの手法としてLogic AnalyzerとVirtual IOがあります。

実現できること

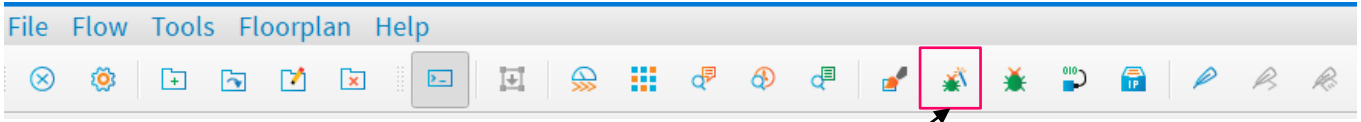
- JTAG I/Fを用いたハードウエアデバッグ
- Virtual IOを用い信号のモニタとドライブ
- Logic Analyzerでトリガ条件を指定しての信号キャプチャ

本書ではハードウエアデバッガのLogic Analyzerに関して説明いたします

Debuggerの準備



Debugger Wizard - Logic Analyzer 準備

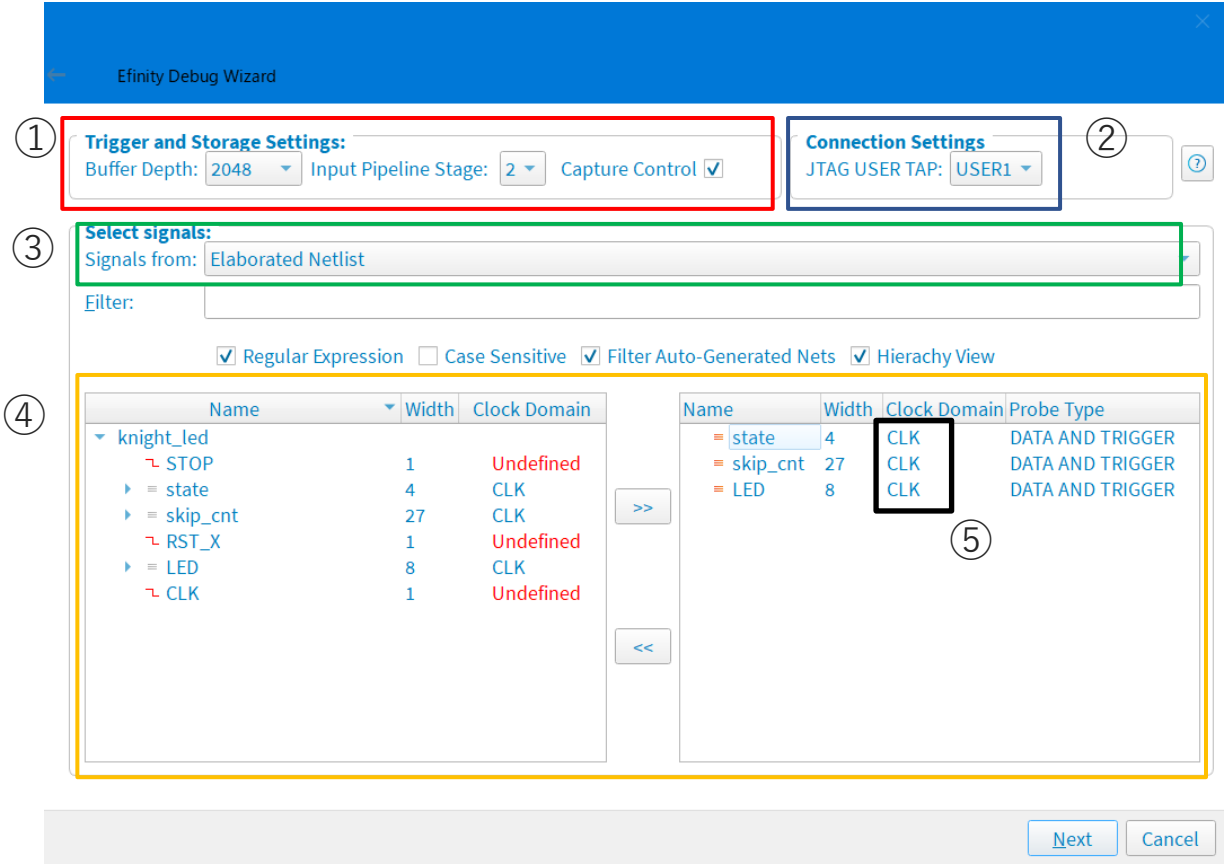


Logic Analyzerではdebugを行なう前にdebug wizardを用いて事前の準備ができます
debug wizardでは

- debug対象信号の指定
- 観測領域（buffer depth）の定義

などを設定できます

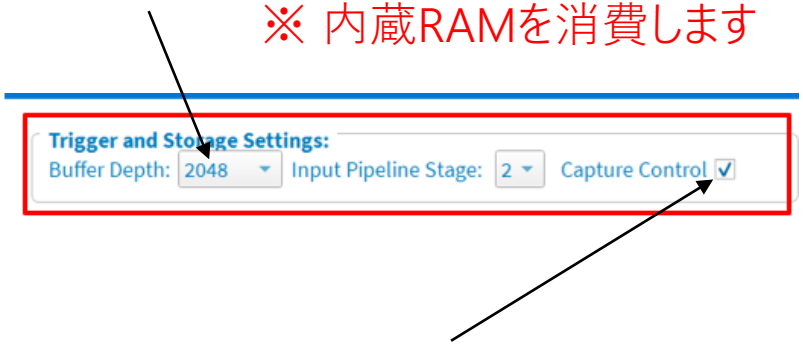
Debugger Wizard画面



Debugger Wizard ①項目設定

- ①赤枠
- Buffer Depth - 観測量（バッファの深さ）をプルダウンリストから選択

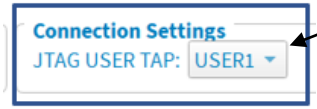
※ 内蔵RAMを消費します

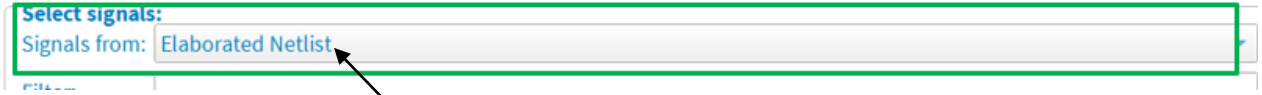


- Capture Control - checkすればcapture条件を設定可能（常にチェック！）

Debugger Wizard

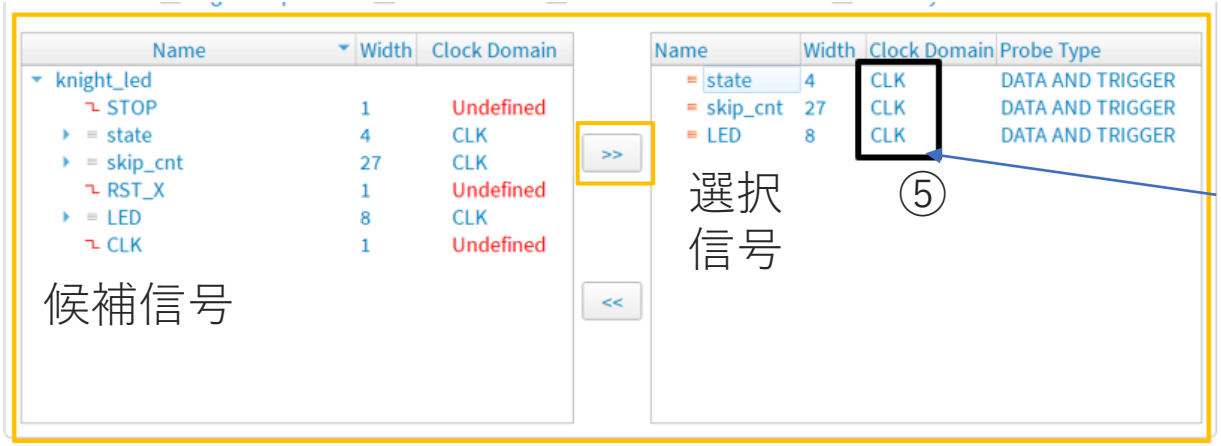
- ②青枠
  - JTAG USER TAP - USERデザインに合わせプルダウンリストから選択 (USER1,2,3,4)





- ③緑枠
  - Signals from - Post-Map, elaborated (Pre-Map) Netlistのいずれかを選択

Debugger Wizard



➢ ④橙枠

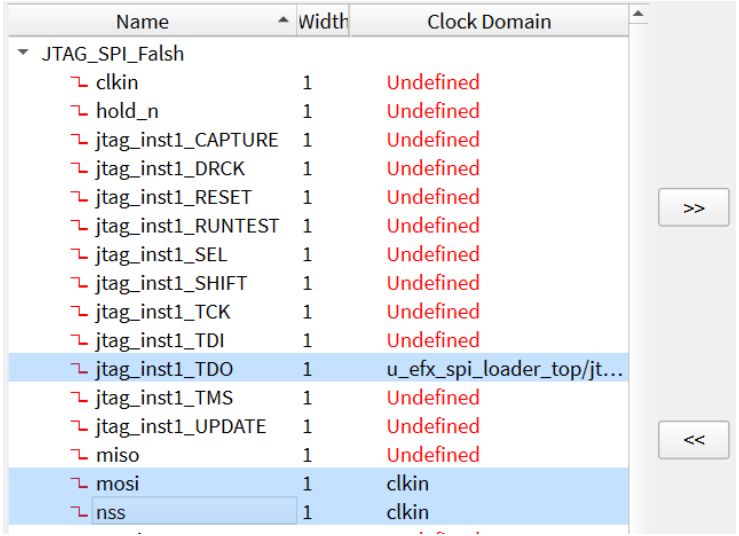
- 信号の選択 - Debugで観測する信号を選択
  左のウインドウからクリック　>>　ボタンで選択

➢ ⑤黒枠

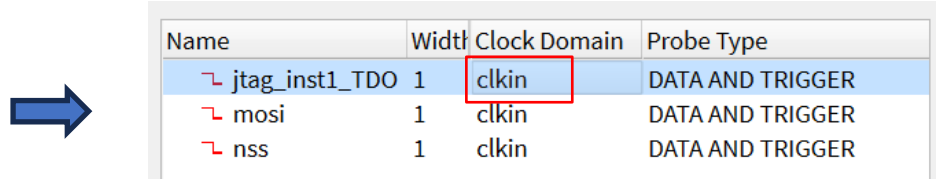
- Clock Domain – サンプリングするClock sourceを選択
  Undefinedの信号もClock sourceを選択することでDebug対象にできます

Clock Domain - 注意点

- **指定できるClock Sourceは１種類のみ**
  - Clock Sourceを複数指定するとDebuggerは動作しません
    → 駆動クロックが異なる信号がある場合、Clock sourceの変更必要



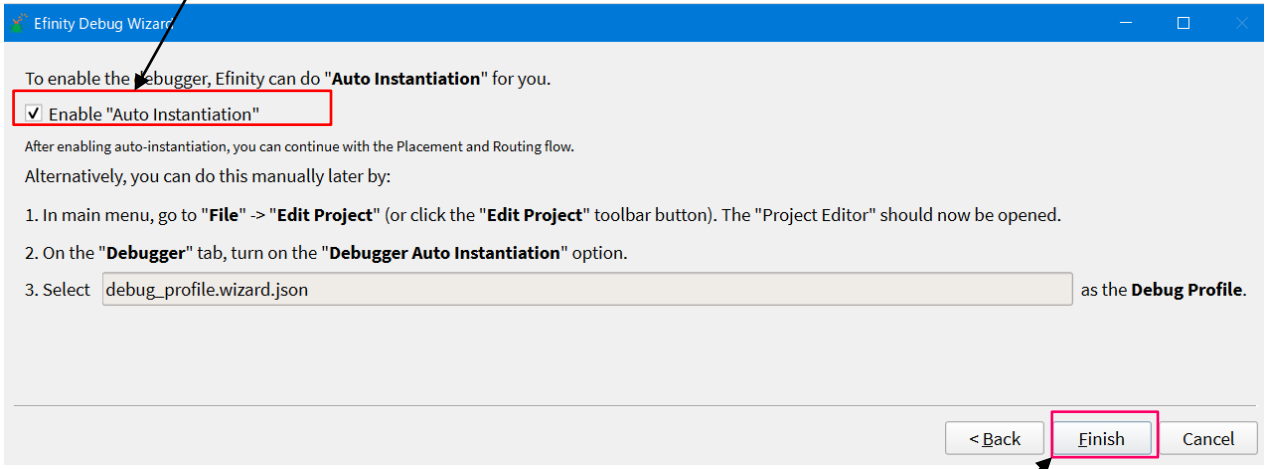
異なるClock Domainの信号を選択



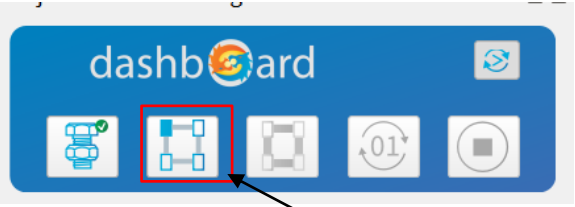
異なるClock Domainの欄をクリック
→サンプリングClockを変更し１種のみとする

Debugger Wizard

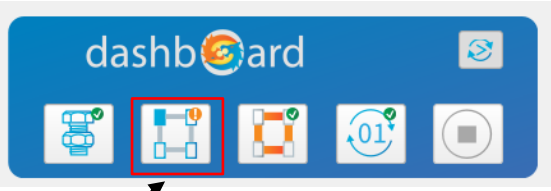
'Enable Auto Instantiation'がチェックされていることを確認！



Finishをクリック → Debug Instance生成



or

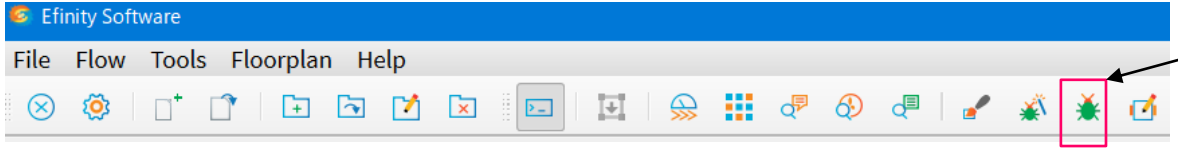


Wizardを終了すると‘Place’ボタンがグレイアウト or !マーク

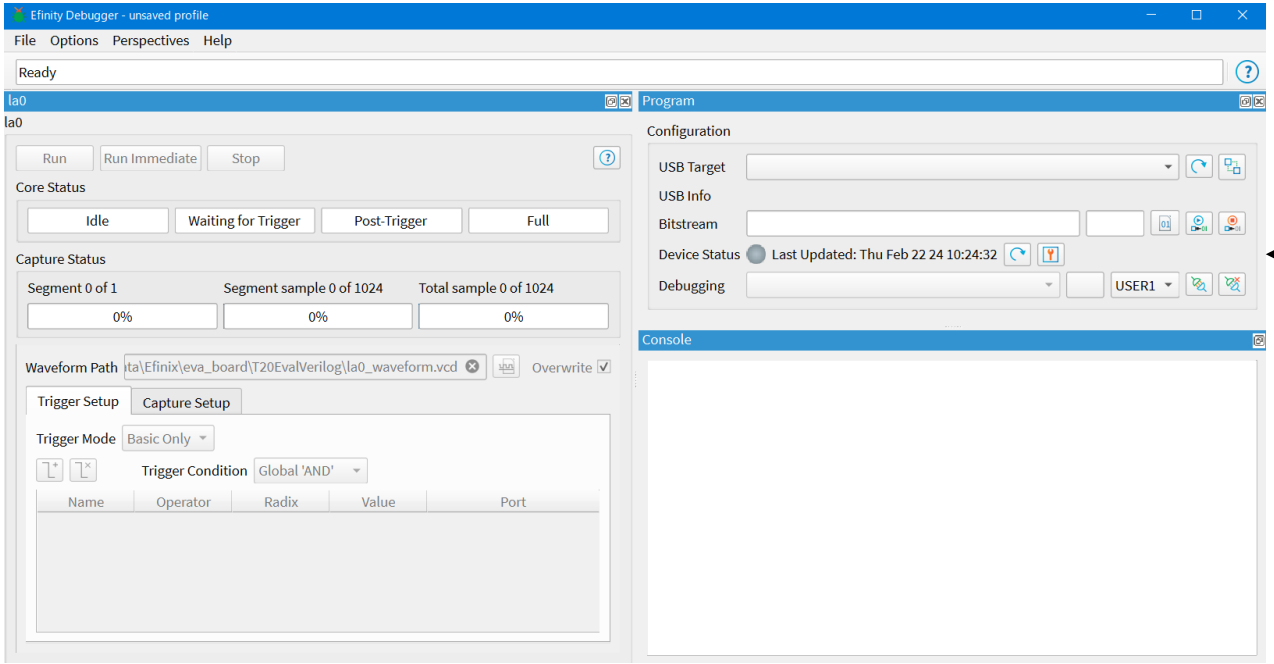
- Placeボタンを押し再コンパイル

コンパイルを終了すると Logic Analyzerによるデバッグが
可能な環境が準備できます

Debugger Logic Analyzer起動

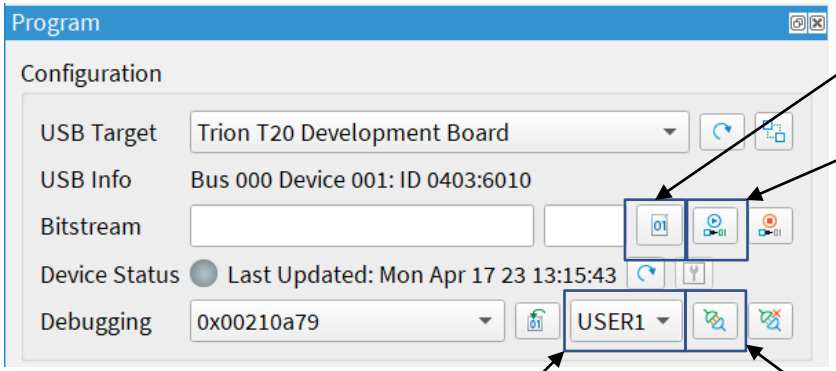


アイコンをクリック
→ Debuggerを起動



起動時のDebugger
画面

Debugger - Program



(1) Bit Streamファイル (.bit)を選択

(2) プログラム開始

(3) JTAG User TAP選択

(4) コネクトをクリックしデバッガ接続

Debuggerを起動したら → ‘Program’のWindowにて

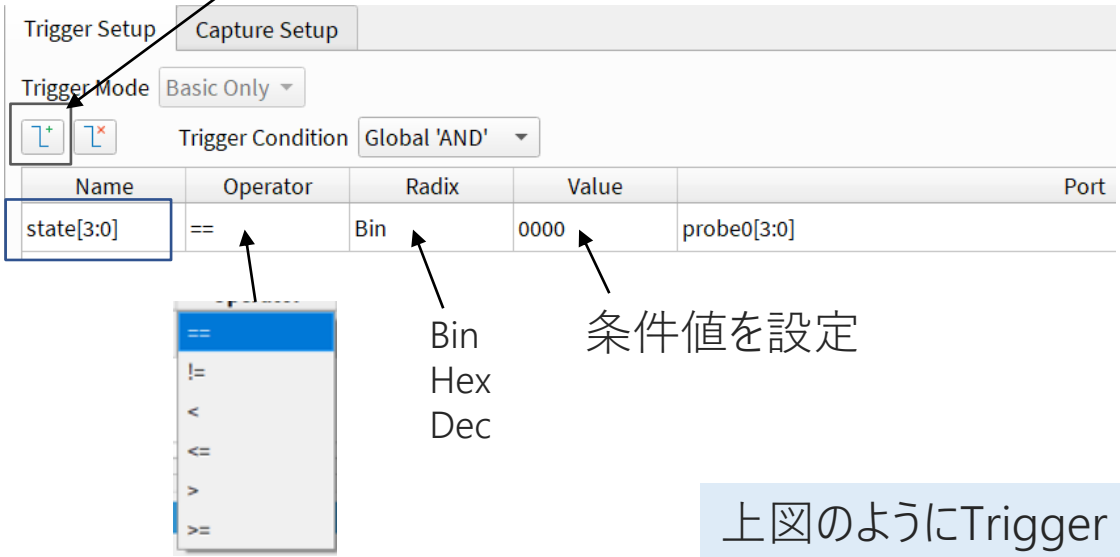
✓ (1) Bit Streamファイルを選択 → (2) プログラムを行なう

✓ (3) JTAG User TAPの選択が正しいことを確認 (P.7にて設定と一致しているか？)

➢ (4) コネクトしDebuggerを開始できる状態となります。

Debugger - Trigger Setup ->トリガ条件設定

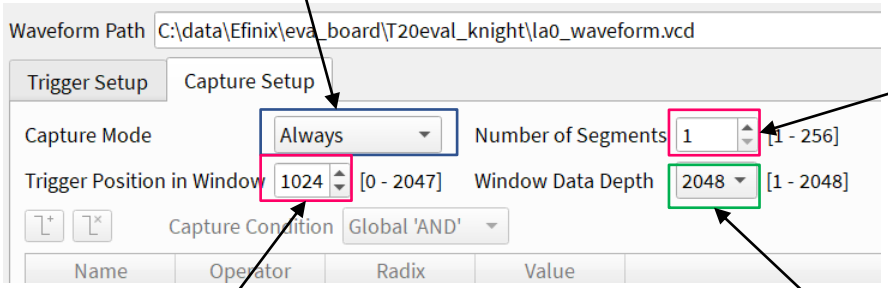
トリガー信号追加アイコン



上図のようにTrigger Setup画面で
Trigger条件を設定します。

Capture Setup

- Always - Trigger条件に関係なく波形データをキャプチャする
- Basic - Trigger条件と一致した時に波形をキャプチャ



キャプチャする回数を設定
=トリガ発生回数

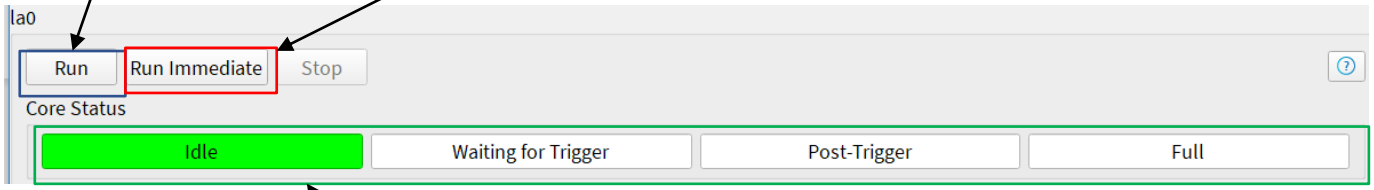
Trigger位置

キャプチャ1回分のバッファ容量
= Buffer Depth / キャプチャ回数

Debugger - Run

Debug開始
- Trigger条件満たすまで実行

Run Immediate
- Buffer Depth 1回分のRun実行



Debug実行中の状態を表示

- Runボタンを押してDebug開始 → トリガー条件を満たすとSTOPする
- Run Immediateを押すとBuffer Depth分'Run'→'STOP' （トリガ条件不要）
- Run -> Stopボタン押下でも波形ファイルは保存されます（トリガ条件不要）

Debug 結果 - 波形確認

Efinityには波形表示機能がありません。従ってサードパーティ製の波形表示ツールが必要となります。
Debugの結果は'vcd (Value Change Dump)'という形式のファイルで出力されます。
この形式をサポートする波形表示ツールを使用すればdebug結果を確認できます。

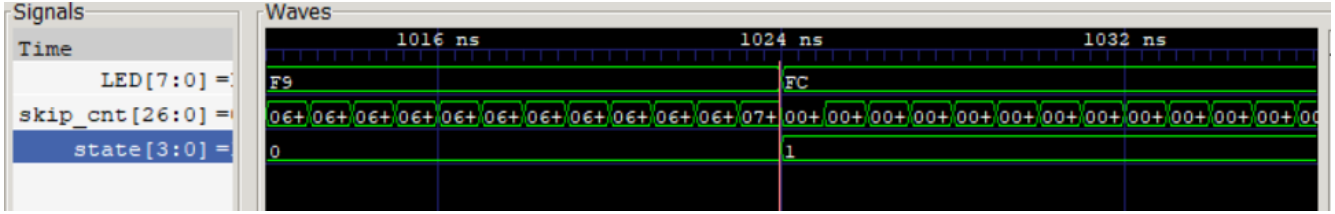
'vcd'形式をサポートする無料の波形表示ツールとして'GTKWave'が存在します。

Efinix社でも波形表示ツールとしてGTKWaveを紹介しています。
下記URLからダウンロード可能です。

gtkwave - Browse Files at SourceForge.net

表示波形

Debug結果の表示波形例を以下に示します
(GTKWaveによる表示波形)

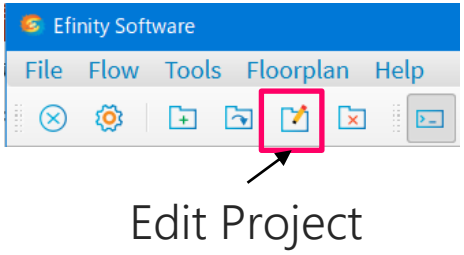


手順

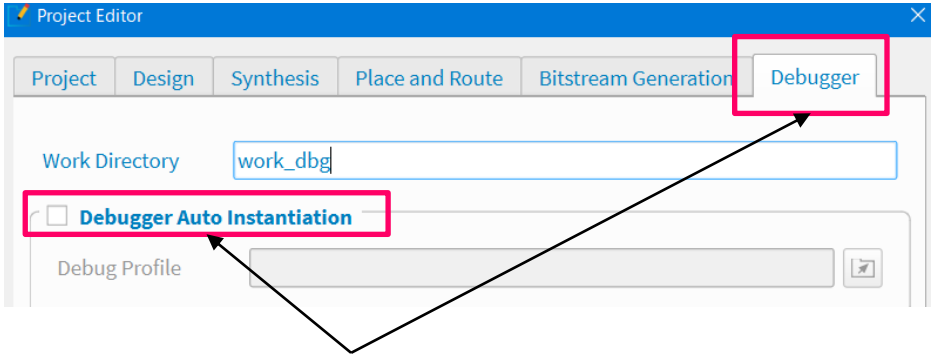
1. GTK Waveを起動
2. 波形を確認したい対象の'vcd'ファイルをGTK Waveへドラッグ＆ドロップ

Auto Instantiationデバッグコア削除方法

Debug Wizardで作成したAuto Instantiationデバッグコア（Logic Analyzer）はデバッグ終了後は冗長回路となるので削除します。
削除の方法は以下の通り



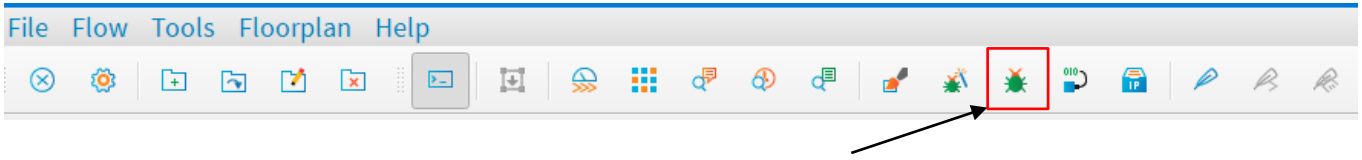
Edit Project



Debuggerタブを選択 ->
Debugger Auto InstantiationをOFFにする

再コンパイル実行

ManualでのDebug信号定義



Debug Wizard以外にもProfile EditorにてDebug対象信号を定義することができます。

Open Debuggerアイコンをクリック
→ Debuggerを起動しProfile Editorで定義します

※ ただし、デバッグモジュール (HDL) 追加の他
トップモジュールの修正も必要です
→ ManualでのDebugは手間を要するので本書での説明は割愛します。

EOF