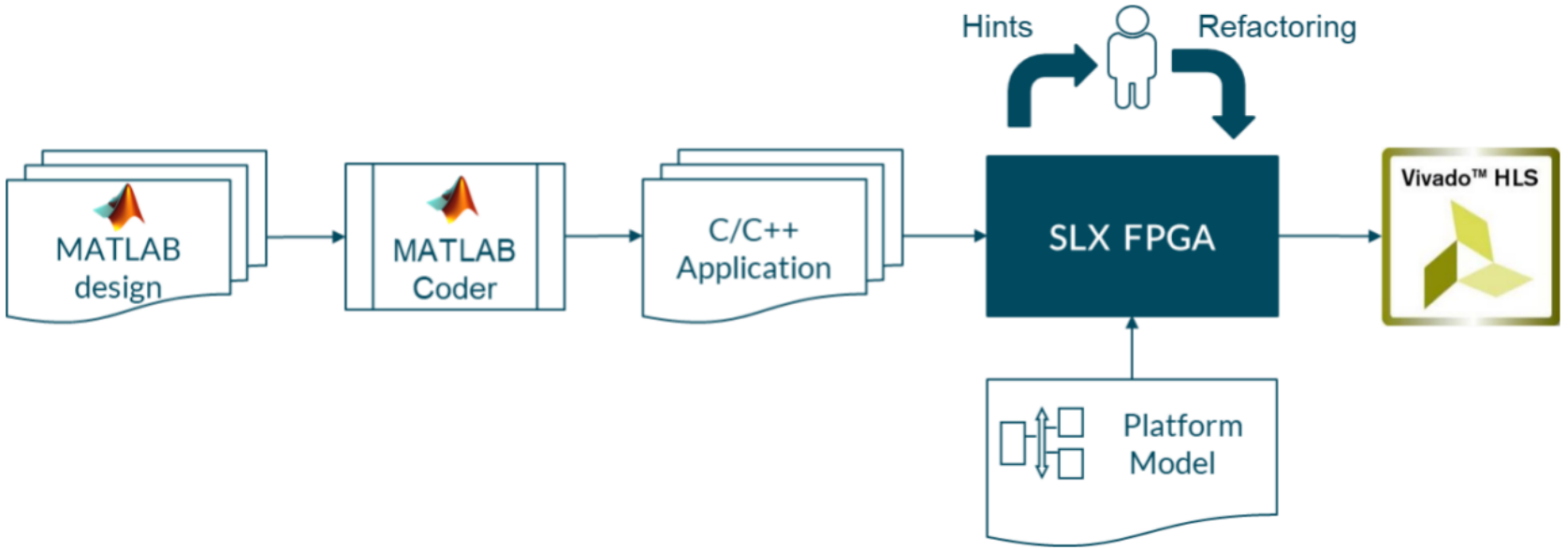
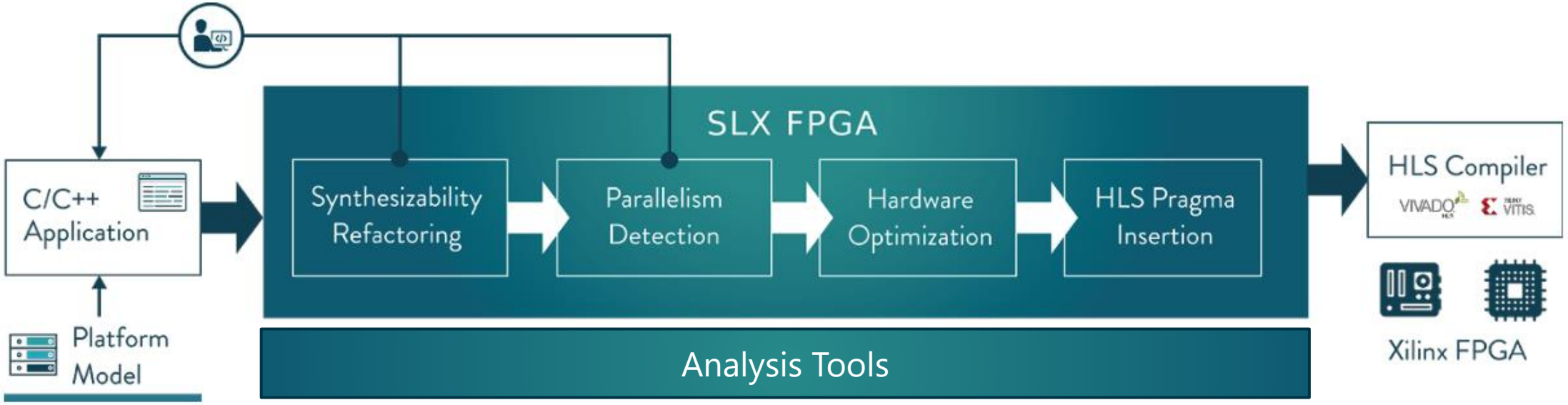
SLX FPGA with MATLAB Coder

- HDL Coderでサポートされていないコードの HW化
- HLS用の合成不可能なC/C++コードのリファクタリング
- FPGA実行効率を考慮した並列性の検出
- HW最適化されたHLSプラグマの自動挿入



今日から使えるSLX FPGA 開発フロー

C/C++のFPGA化を加速



SLX FPGAは、HLSを使用するための様々な障害を取り除く

- 合成できないC/C++コード
- 並列性の検出
- 性能とFPGAリソースの最適化

HWエンジニア：

ソースコード解析とガイダンスを取得

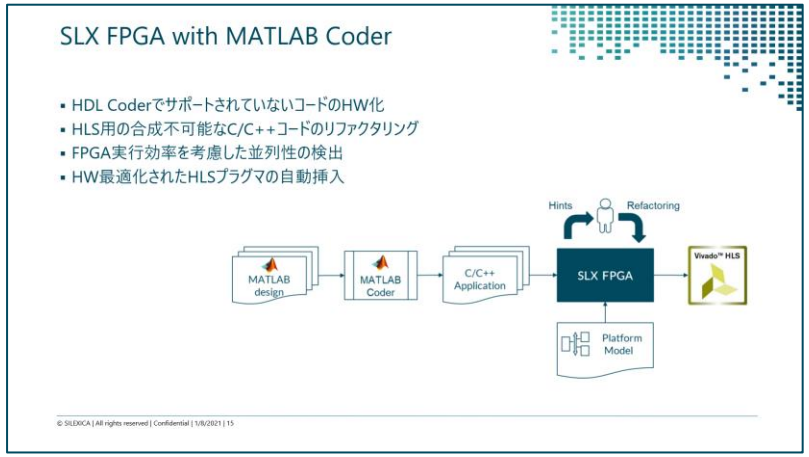
SWエンジニア：

並列処理/ハードウェア向けのガイダンスを取得

C/C++, HLS フローにおける改善ステップ

アルゴリズム設計を生かした設計手法

- C/C++ コードレビュー
  - データフロー、アルゴリズムとして作成されたコードの確認と解析
  - IP のチェック
- HW 化のためのリファクタリング
  - 実体化の必要なポインタの探索、HW 化スコープの探索
  - 境界条件設定、プラグマ挿入、合成条件設定、効果の確認
- 並列化のためのリファクタリング
  - Unroll, Pipeline 化など、パフォーマンス向上の検討
  - 阻害要因の除去(LCD、アクセス順)
- 最適化のためのリファクタリング
  - HLS ツールの結果を確認し、コードの表現を再検討
  - データのライフタイム調整、構造化などを反映し、再合成
    - Array の分割、変数のスコープ変更

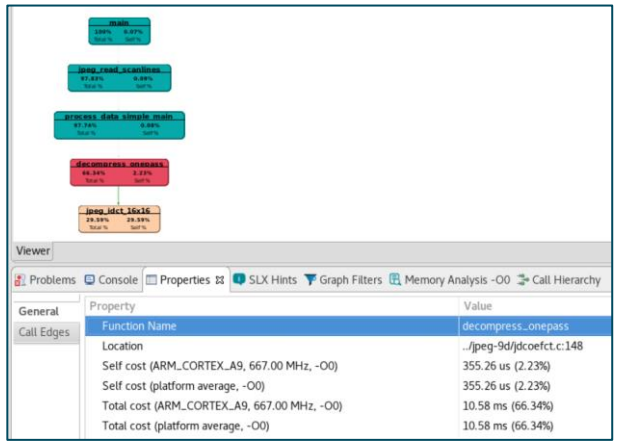


LCD : Loop carried dependency

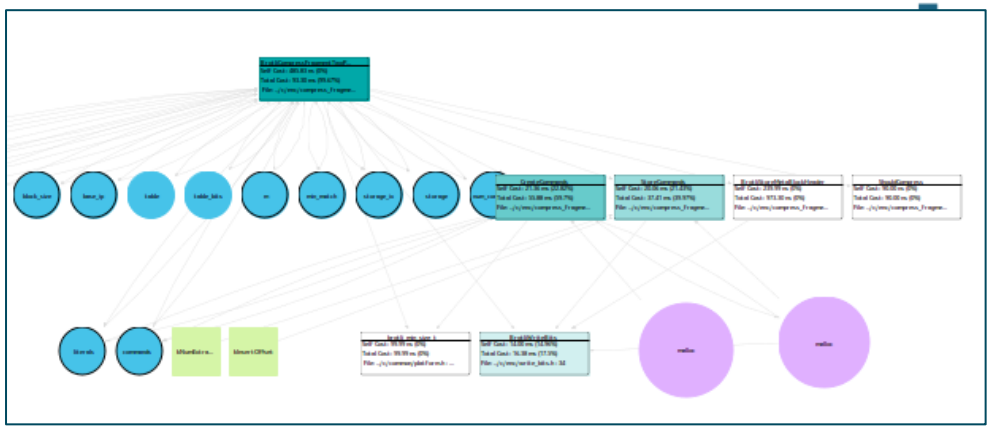
すぐに！簡単に！使える SLX FPGAツール

フェーズ2：分析とコード変換

- **分析：**
  - Dynamic 解析の表示 ***Code Insights Section*** ツールバーのボタン を利用して、アプリケーションとソースコード構造を解析



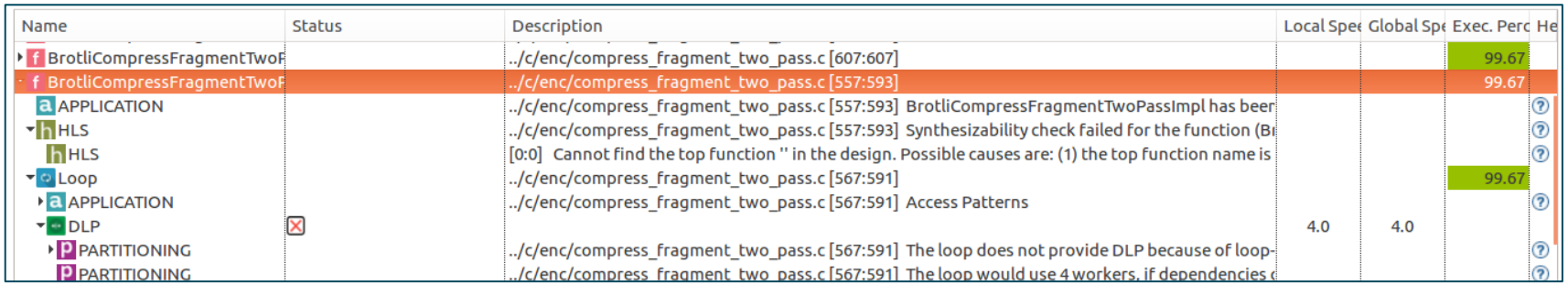
コールグラフ（*ホットスポット分析*）



コード分析グラフ
（*ホットスポット構成要素とメモリの依存関係*）

- **コードリファクタリング：**
  - new を使った初期化などハード化できないコードを事前に修正

（例：コード分析グラフに表示されている紫のmallocノードなど）



ヒントの表示
（*並列処理、並列性阻害要因、HW不相応*）

すぐに！簡単に！使える SLX FPGAツール

フェーズ3：最適化とHW化

- **HW化：**
  - HLS コマンドをバックグラウンドで実行
  - SLX上で、ユーザーへ結果を表示
- **最適化：**
  - HW化の問題を解決
  - 並列阻害要因の排除
  - 最適化発見的手法の実行
  - プラグマを挿入したコードを生成（ユーザー制御可）