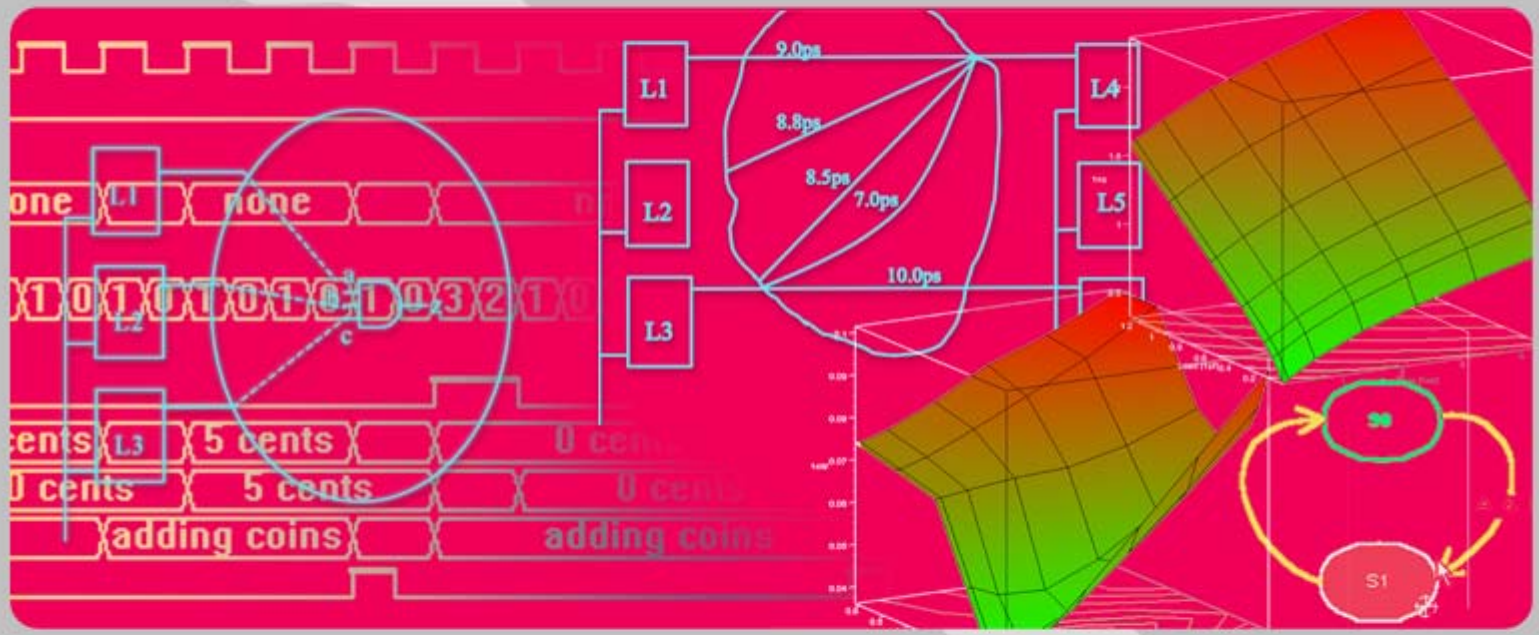
# AccuCore –
## STA機能を持ち、SPICE の精度を保持する
## コア･キャラクタライゼーション･ツール




シルバコ･ジャパン　テクノロジー･セミナー

Spring 2007

Kaoru Kashimura

SILVACO

2007/4/17

# AccuCore の必要性、使用用途について

- AccuCore は、ブロック・レベル・デザインにおけるスタティック・タイミング解析機能を持つキャラクタライゼーション・ツールです
- 90ナノ、65ナノ以下のディープ・サブミクロン・テクノロジーにおいて、数百万ゲート規模の複雑でかつ、多数のスタティック / ダイナミック回路の組み合わせで構築しているデザインの設計者に、AccuCoreを使用する必要性を見出すことができます

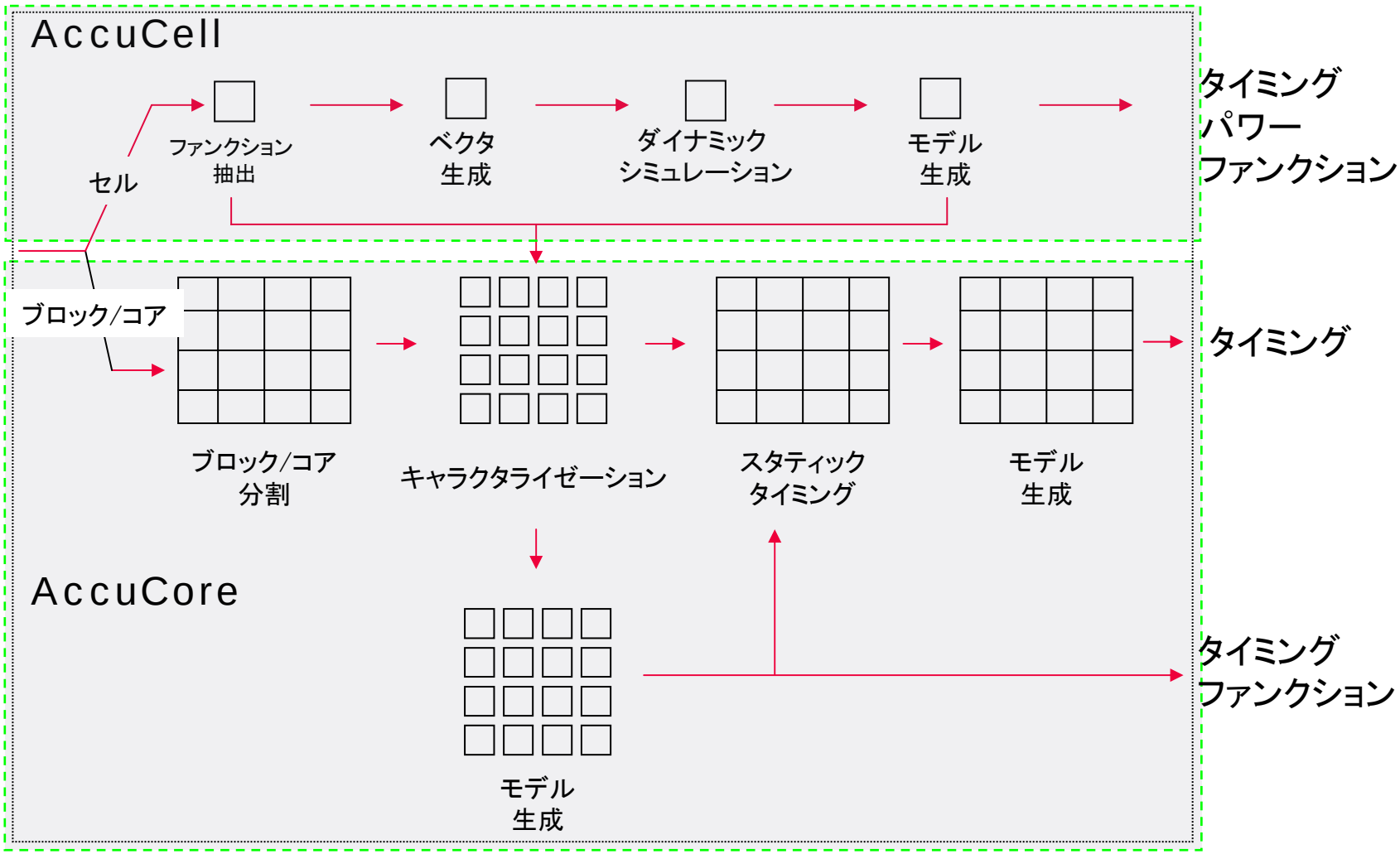
- スタティック、ダイナミックデザインに対応
- 多階層もしくはフラットのRCデザイン
- 分割、ファンクション認識、ベクタ生成を自動で実行
- 状態依存のタイミングファイル .lib & TLF
- フォルスパスの除去を自動で実行

- クリティカルパスをSPICEネットリストに出力
- コンプレス・モデルの生成
- 統合されたSmartSpiceのAPI機能
- Firebird DB – インクリメンタル手法
- 最新のSPICEモデルに対応
- 他社のSPICEエンジンに対応



SILVACO

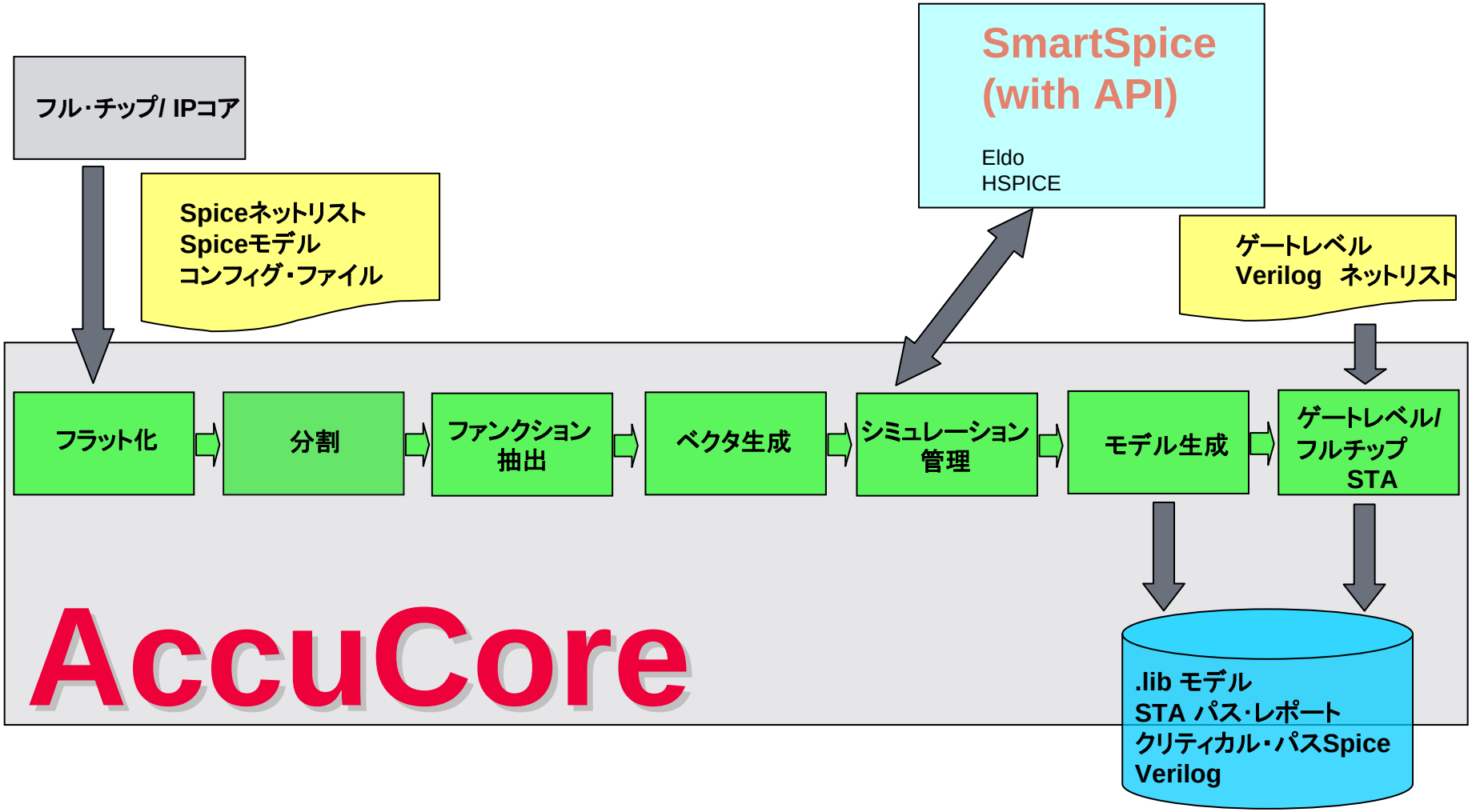
# AccuCell and AccuCore キャラクタライゼーション・フロー






SILVACO

# AccuCore デザイン･フロー

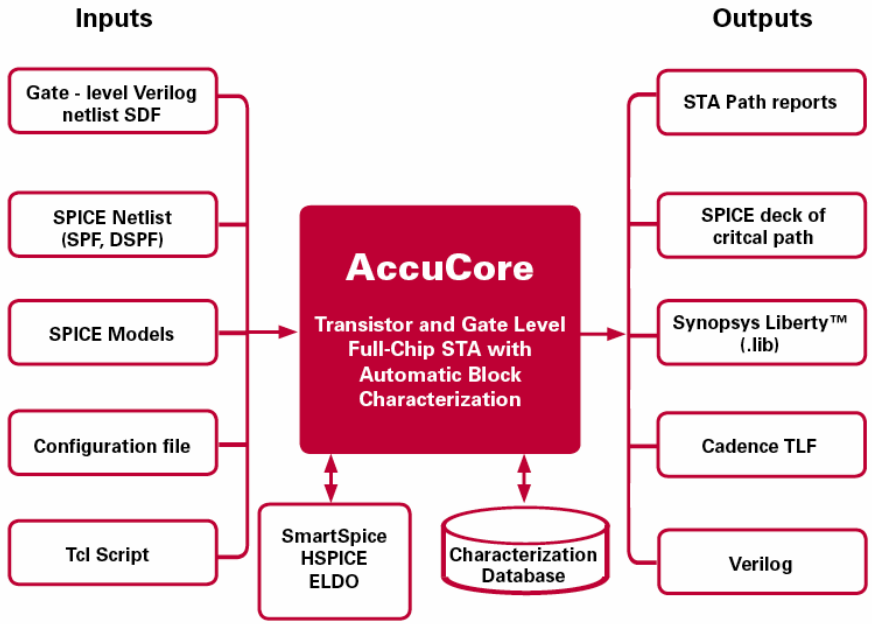





SILVACO

# AccuCore – ハイ・パフォーマンス・タイミング・エンジン

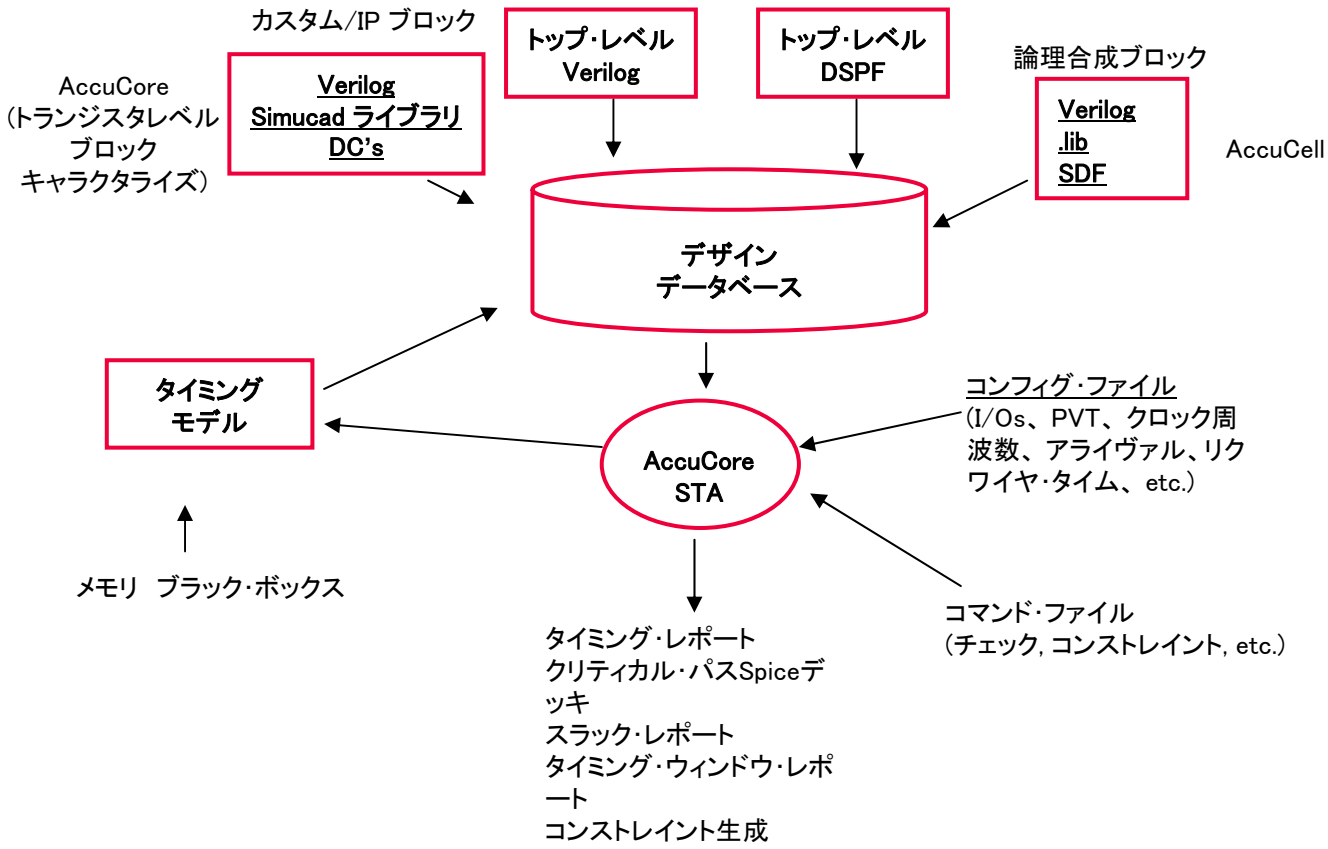
- 業界標準フォーマットの入出力に対応
- Firebird データベースによりインクリメンタル・キャラクタライゼーション
- ゲート・レベル・タイミング解析 – ほとんどのSPICEモデルに対応
- ブロックレベルでのタイミング・モデル・オプションにより、階層化されたフルチップのタイミング解析を迅速に実行
- スタティックまたはダイナミック回路に対応、Tcl スクリプトを用いて自動実行






SILVACO

# AccuCore スタティック･タイミング解析　フルチップ･フロー




## AccuCore スタティック タイミング解析

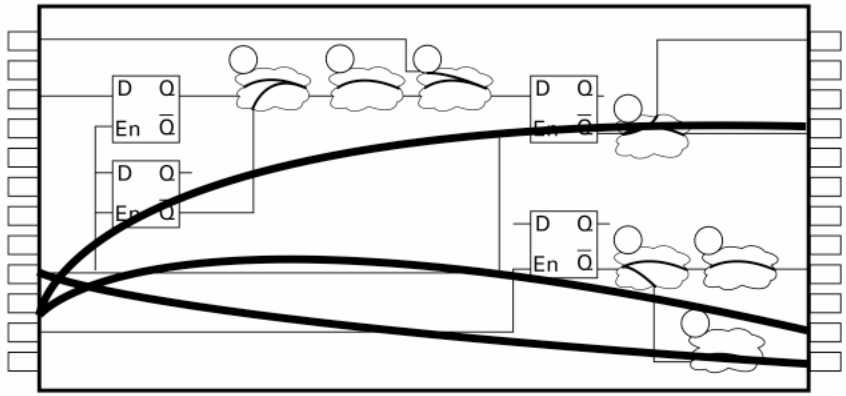
1. すべてのパス･モデルを読み込み
2. ブロックレベルのタイミング･チェックを実行
3. ブロック･レベルのクリティカル･パス解析を実行
4. クリティカル･パスのSpiceデッキを作成
5. ブロック･レベルのコンプレス･モデルを生成

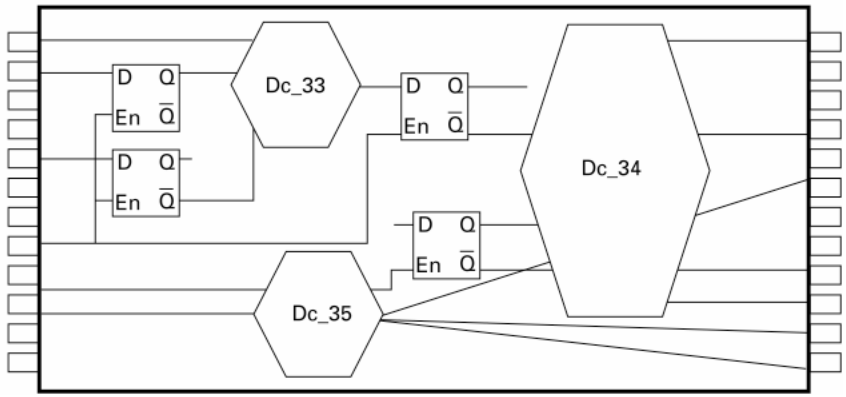


SILVACO

# AccuCore スタティック･タイミング解析　タイミング･モデル

- すべてのパス･モデル(all path)
  - デザイン上のすべてのパス
- コンプレス･モデル (N-level)
  - コンビネーションデバイスはモデル化しない
  - デバッグ･パス情報
  - ディレイとスロープ･テーブルをサポート
- ブラックボックス･モデル
  - デバッグ･パス情報
  - ディレイとスロープ･テーブルをサポート
- インターフェース･リング･モデル
  - インターフェースロジックをそのままモデル化
- インターフェース･コンプレス･モデル
  - インターフェースデバイスをそのままモデル化
  - 残りのコンビネーションロジックは、モデル化しません
- ブロック･コンストレイント生成
  - ASICブロック用にブロックバウンダリを生成
- クロック･スキュー･解析






SILVACO

# Step 1: KEEP_SUBCKT, FIND_SUBCKTを用いてコンフィグファイルを作成

分割を実行する際に考えられるケースとして、ある回路構成をそのまま一つの機能ブロックとしてAccuCoreが機能分割をして欲しいという要求があるかと思います。その場合、もし階層構造で記述されているSpiceネットリストであり、かつ、回路構成が下位階層に一つの.subckt として存在している場合は、KEEP_SUBCKT コマンドをコンフィグファイルに記述します。

KEEP_SUBCKT のシンタックスは下記のようになります:

KEEP_SUBCKT <subckt_name> <inputs> <outputs> <bidirs>¥
<clocks> <optional_table_filename>

FIND_SUBCKT のコマンドは、フラットのSpiceネットリスト時に使用します。KEEP_SUBCKT のコマンドと同様に、ある回路構成をそのまま一つの機能ブロックとして認識させます。このコマンドを使用する場合は回路の構成をSpiceネットリスト形式でサンプルとして与える必要があります。

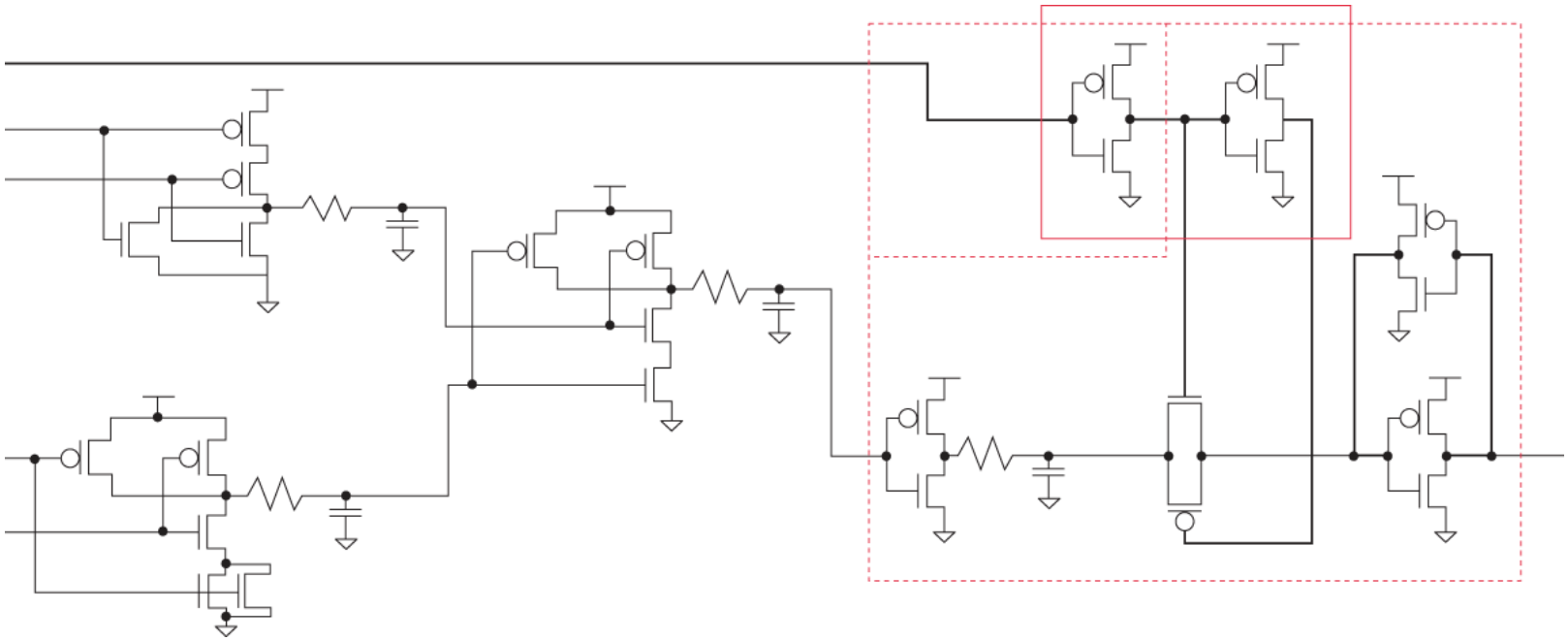
FIND_SUBCKT のシンタックスは下記のようになります:

FIND_SUBCKT <pattern_name> <pattern_spice_netlist> <powers> <grounds> ¥ <inputs> <outputs> <bidirs> <clocks> <optional_table_filename>

# Step 2: パラレル･デバイスのマージ、クロック伝播、ラッチの認識

- 不必要なデバイスを除去してフラット化
- カップリング･キャパシタンスを形成
- 接続されている抵抗をトレース
- VDD, GND、トランジスタのデバイスタイプを付加
- クロックの属性を割り当て
- パラレルデバイス、RCをマージ

- ノード、ドライバをスタティック、もしくはダイナミックのコンディションに振り分け
- KEEP_SUBCKT、FIND_SUBCKT.を使用して機能を特定
- 伝播クロックの認識
- インバータ、ゲーティング･ロジックの認識
- ラッチ･ノード、ラッチ･デバイスの認識

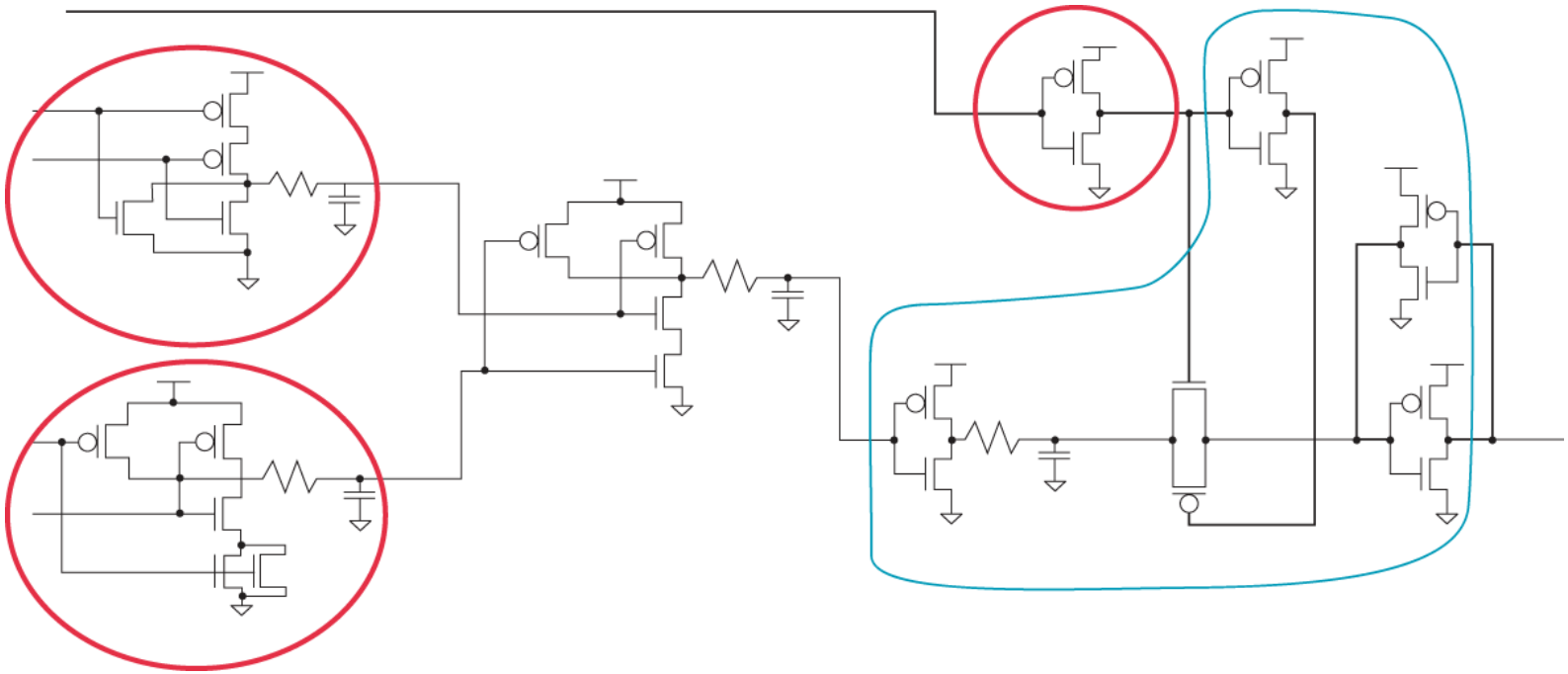




SILVACO

# Step 3:　ネットリストをデザイン･クラスタ (DCs) に分割

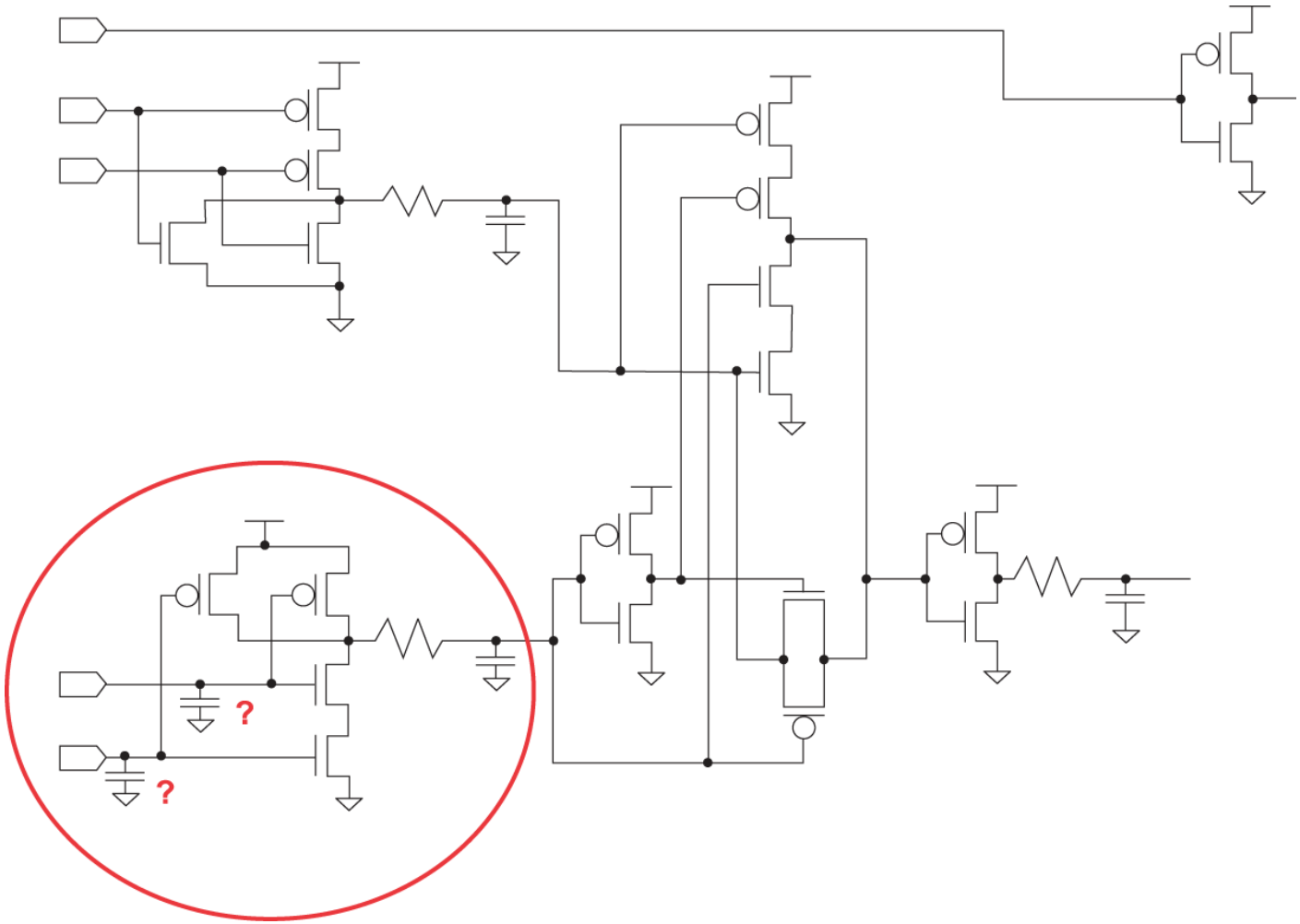
- ネットリストよりデザイン･クラスタ(DCs)に分割します
- チャネル･コネクト･コンポーネント (CCC's) –　ソース･ドレイン間に接続されているノードとぶら下がっているトランジスタを一つのコンポーネントとして認識
- フィードバックループ形態および強固に関連性のあるリージョンをマージします
- マルチプレクサは、一つのまとまった回路として認識されます





SILVACO

# Step 4: 最上位のデザイン･クラスタにおける入力ピンキャパシタンスをキャラクタライズ

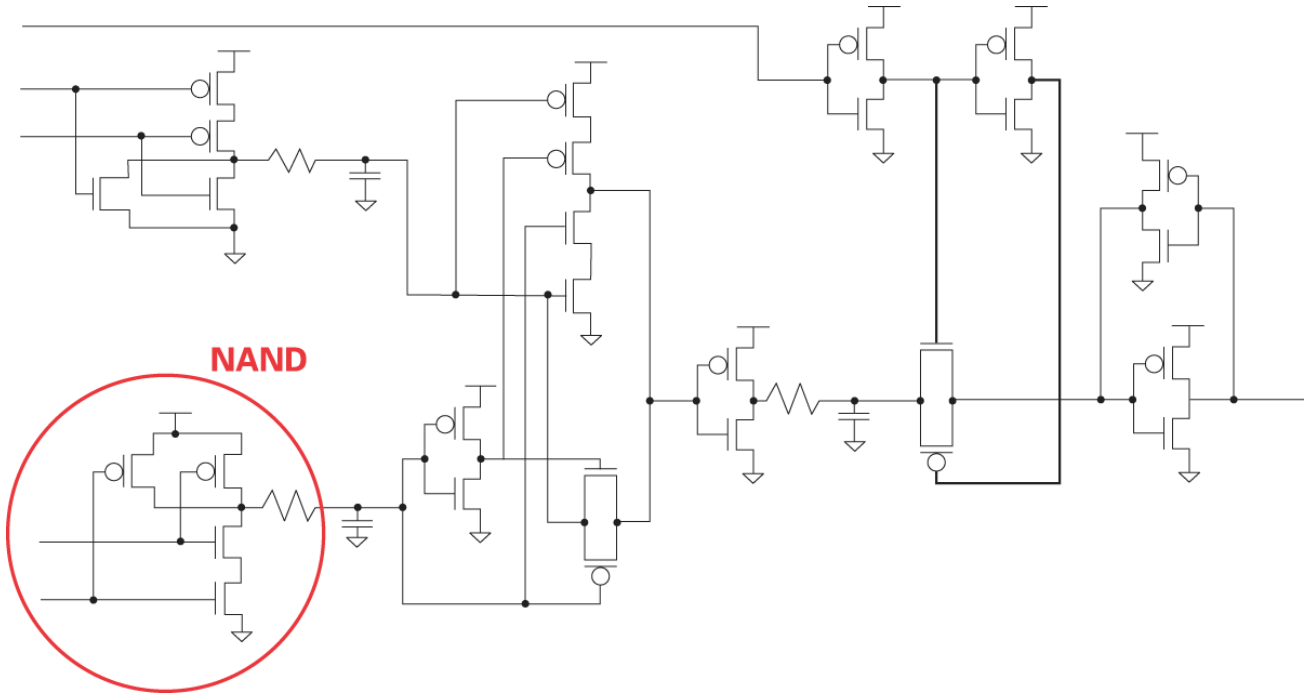





SILVACO

# Step 5: デザイン･クラスタのファンクション抽出

- AccuCoreは、独特のBDDをベースとしたアルゴリズムを用いて、デザイン･クラスタのファンクションを決定します。
- AccuCoreは、独特のアルゴリズムによりデザイン･クラスタのキャラクタライズ用ベクタを最小限の構成になるように生成します。
- フォルス･パスの除去 – デザイン･クラスタ内におけるフォルス･パスを、論理的に可能な限り除去します。





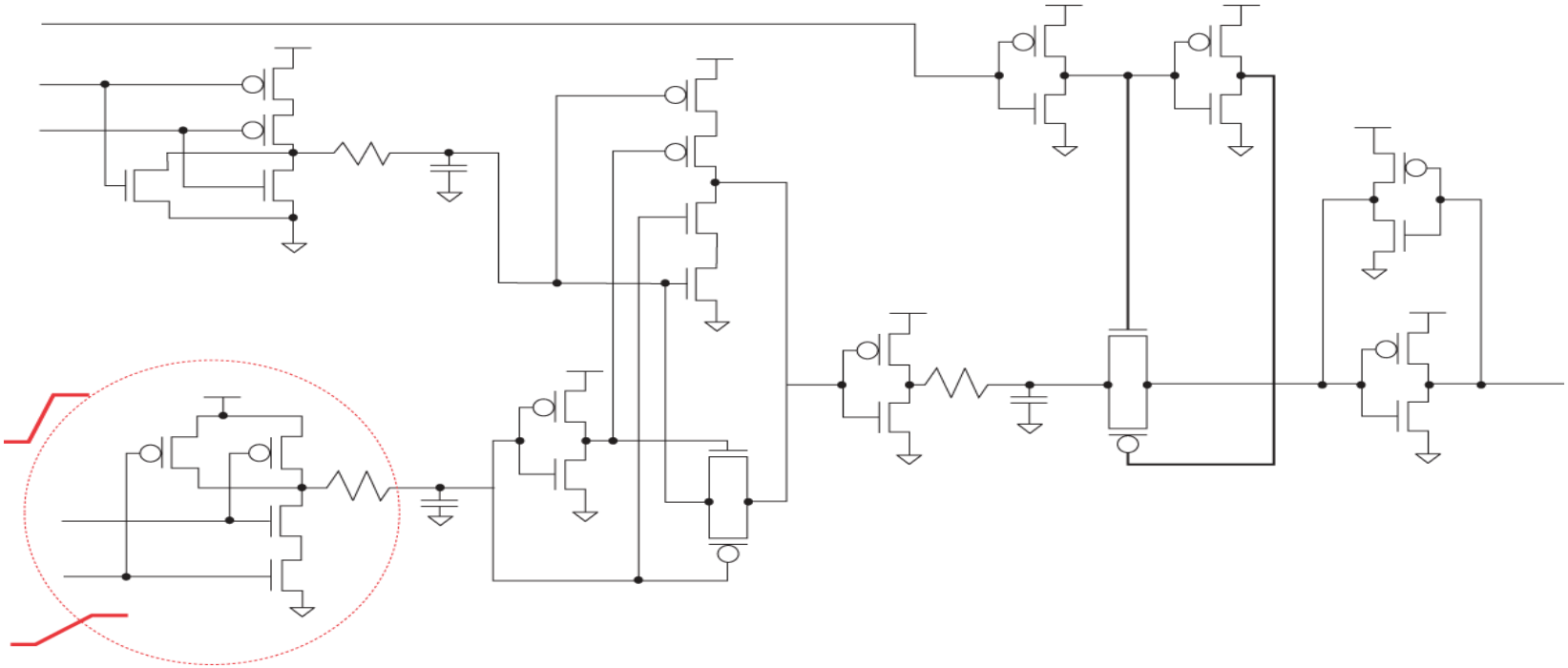

SILVACO

# Step 6: 入力の傾きを伝播し、実際の回路における負荷をデザイン･クラスタの出力に付加し、キャラクタライズ

- キャラクタライズの要求事項と入力の傾きは、次段のデザイン･クラスタに伝播されます。
- 実際の回路におけるデザイン･クラスタの出力の負荷は、ダイナミックシミュレーションを使用したキャラクタライゼーション用に自動で認識し、整形されます。
- バイパスシミュレーションをベースにしたキャラクタライゼーションを実行する場合、前段の結果がストアされているデータベースよりルックアップテーブルの値を派生してデザイン･クラスタに付加し

キャラクタライゼーションを 実行します

- ユーザーがコンフィグレーションファイルに定義するパラメータの許容範囲をベースに決定します。

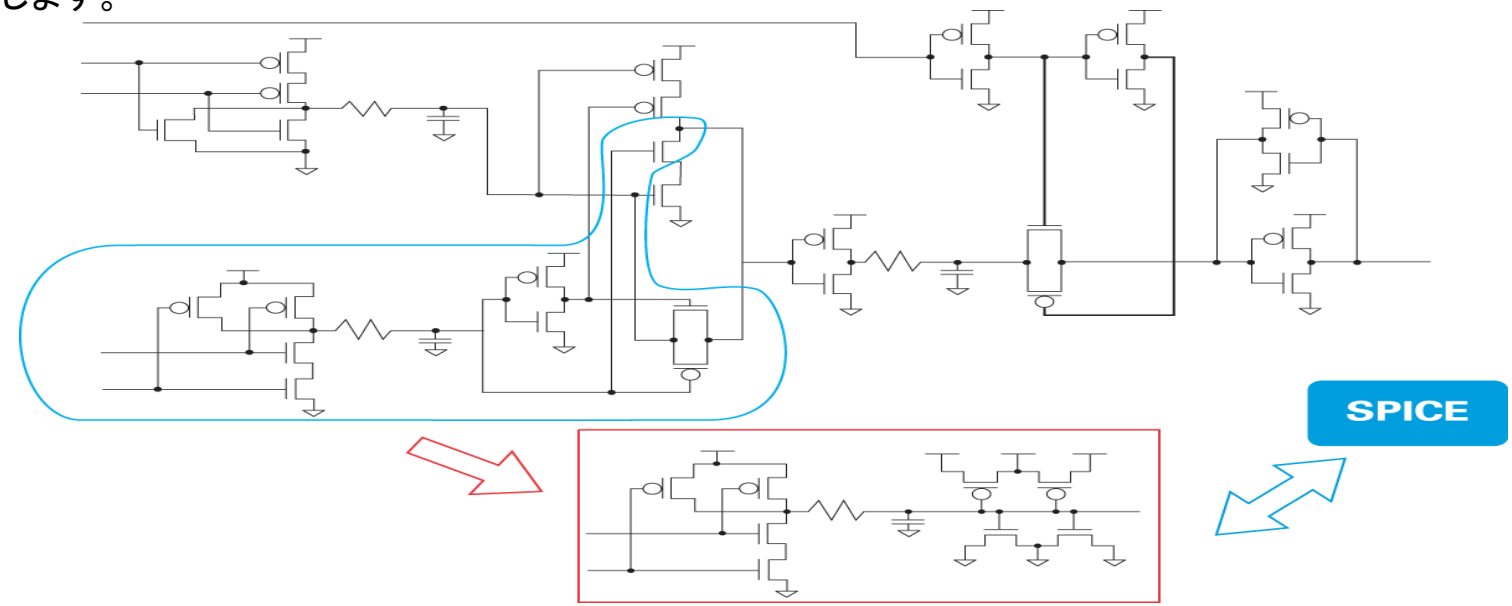




SILVACO

# Step 7: デザイン･クラスタをダイナミック･シミュレータを用いてキャラクタライズ、結果をストア

- “fold”されていないデザイン･クラスタのキャラクタライゼーション用シミュレーション･デッキを生成します
- 伝播遅延及びトランジション･レートのキャラクタライズを先に実行します。その後にセットアップ･ホールド、リカバリ･リムーバブル等のキャラクタライズを実行します。
- SPICE シミュレータとリンクし、シミュレーション･デッキを渡します。SmartSpice の API modeでは、シミュレーション･デッキを直接メモリにストアし、読み込みにかかる時間を短縮し、実行します。
- API機能を用いて結果をFirebirdデータベースにストアし、入出力ファイルのトランスレーションに費やす時間を短縮します。
- API機能を用いない場合と、他社のSPICEシミュレーションを使用した場合は、パフォーマンスに莫大な差が生じます。



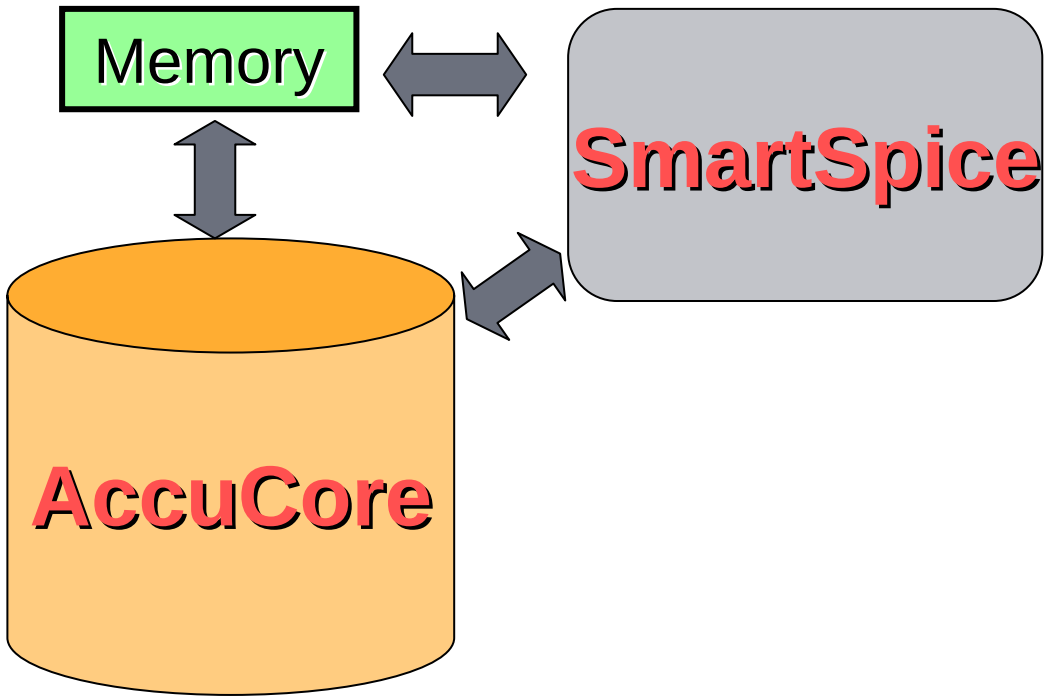



SILVACO

# SmartSpice API 機能

SmartSpice API機能

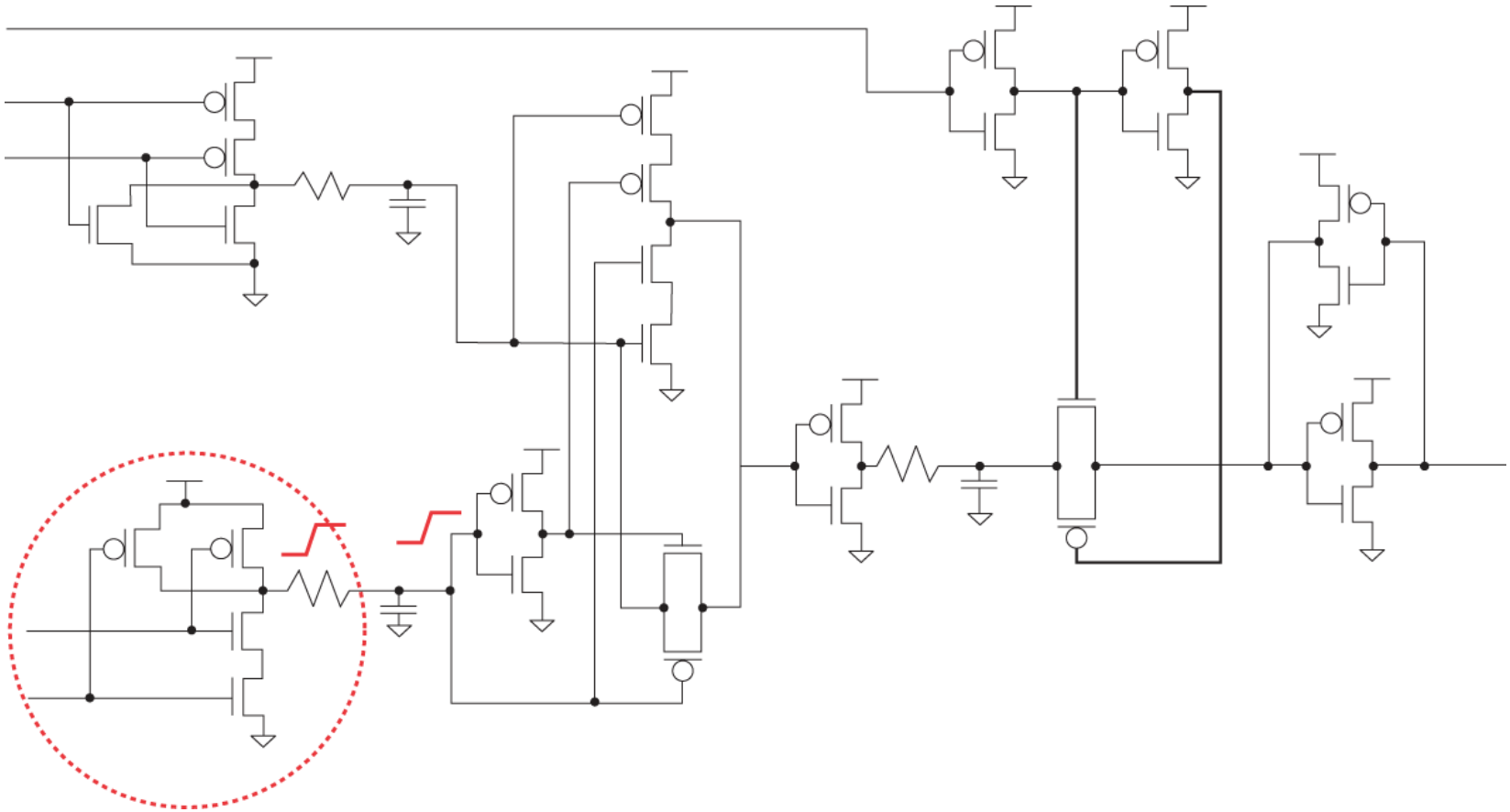
- SmartSpice API機能を使用して、パラメータの変更を直接メモリー上で実行
- 再読み込みの時間が短縮され、さらなるスピードアップを実現！






SILVACO

# Step 8: 出力の傾きを次段のデザイン･クラスタに伝播させ、入力の傾きとしてキャラクタライズを実行





SILVACO

# サマリー

- **正確性**
  - ダイナミック・シミュレーション
  - デザイン上のスロープテーブルを伝播
- **使いやすさ – セットアップ、メンテナンス**
  - シンプル TCL script–コンフィグ・ファイルベース
  - 自動ファンクション抽出
  - ダイナミック・シミュレーション用にベクタを自動生成
  - 自動フォルス・パス除去
- **アグレッシブ・デザインをサポート**
  - ハイ・パフォーマンス・デザイン – ダイナミック・ロジック
- **複雑なミックスト・レベルに対応する内蔵スタティック・タイミング解析ツール**
  - クリティカル・パス、サブ・クリティカル・パス、タイミング・チェック、スラック・レポート
  - クリティカル・パスのSpiceデッキ生成
  - 階層化デザイン、フルチップデザインのSTA用のモデル生成

**迅速なタイミングの収束 – インクリメンタル・キャラクタライゼーション**

**デザイン・サイクルの減少**

**デザイン・クオリティの改善**



SILVACO