はじめに

ディープ・ラーニング、自動運転車、 _ モバイル・ネットワークなどのアプリケーションにより、 の継続的な集積度向上への期待が高まっています。また、デザインの複雑化、設計期間の短縮、プロセスの進歩、そして検証に対する要求の高まりに伴い、これまで以上に迅速かつ効率的なフィジカル検証フローが求められています。しかも、現在最先端の / 、 _、_ プロセスは驚くほど複雑な技術の上に成り立っています。 メーカーはこれまでリソグラフィの限界に挑戦し、より多くのトランジスタを詰め込むことを「法則」としてきました。たとえば1 、 世代では、一般的な 0 2ダイに約20億個のトランジスタが集積されていますが、 、 世代になると同じダイ・サイズに集積されるトランジスタの数が120億個を超えます。ファウンドリは複雑な

フィジカル検証の生産性を向上

フィジカル検証の生産性を高めるアプローチには、以下の　つがあります。

- およびブロック・レベル設計の段階で早期に検証を実行し、「こまめにきれいにする」こと。
- フルチップ検証を効率よく実行し、テープアウト可能なクリーンなデザインに短時間で収束すること。
- 使用する　の数を増やしてサインオフ検証の時間を短縮すること。

IP およびブロック・レベル設計の段階でシームレスに検証を実行

「こまめにきれいにする」という考え方は、日常生活のさまざまな効率化にも役立つアプローチです (料理をしながら使い終わった調理道具を洗うなど)。当然、デザイン作成中の節目ごとに　を実行していけば、それだけでテープアウト直前の混乱とcal スケジュール遅れを防ぐことができます。　の　により、デザイン・ルール・チェック (　)、 (　) チェック、タイミング考慮フィル、プログラマブル・エレクトリカル・ルール・チェック (　) など先進のフローと語稷規佯窖箕まヌÀ屾層度椓屾緣洞みⓝÒ、ズ夰火ヨ脆ヱ◤遂桀ブみ曼楢儨庭層椵虠■µ金ラ謦帉魘椹咠ほ襯楀■ 庨

フルチップ・デザインを早期に効率よく検証し、テープアウト可能なクリーンなデザインに短時間で収束

の　は、フルチップ・デザインのステータスを迅速に評価し、問題の修正に役立つ具体的なフィードバックを返すように設計されています。現在の大規模なチップは、配置配線ブロック、アナログ・セル、メモリー、サードパーティ 、 セルなど数百個ものブロックで構成されています。それぞれのブロックは、設計時に個別に検証されているかもしれ

使用する CPU リソースを増やしてサインオフ検証の時間を短縮する

テープアウトのスケジュール終盤になると、最後の検証ジョブをなるべく短時間で完了できるように、多数の　_　コアを確保する作業が発生します。最大規模の_

リーの利用効率が最大となるようにシステムを調整します。そして重要なのは、これらの機能は理想的な環境でなくても効果を発揮するように設計されているという点です。現実には、異種ホストの混在環境で1つのジョブを千個のコアを使って分散実行することもあれば、ディスクを接続するネットワークのレイテンシが問題になることもあります。こうした場合を考慮し、　　には予期しないホストの再起動、ネットワークやソケットの障害、マシンのクラッシュ、ディスク容量不足などを検出して回復する耐障害性機能があります。

　　のスケジューラは自動での動作に加え、実行中にユーザーが手動でジョブ・リソースを変更することもできます。また、ジョブ開始時には数コアのみを使用し、実行中にコア数を増やして処理速度を高めるといった弾力的な　　管理機能もあります。一般的なコンピューティング・ファームで数百コアを必要とするジョブを実行しようとしても、それだけの数のコア