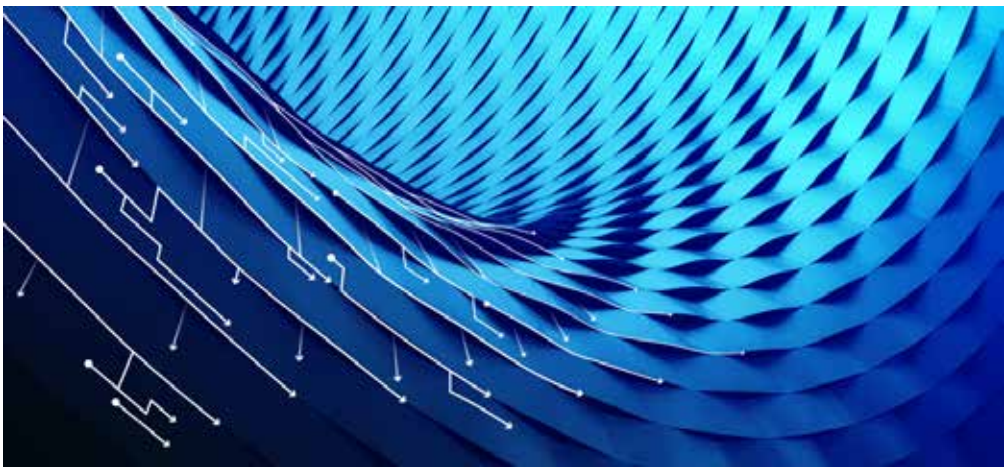
統合仮想プラットフォームの構築

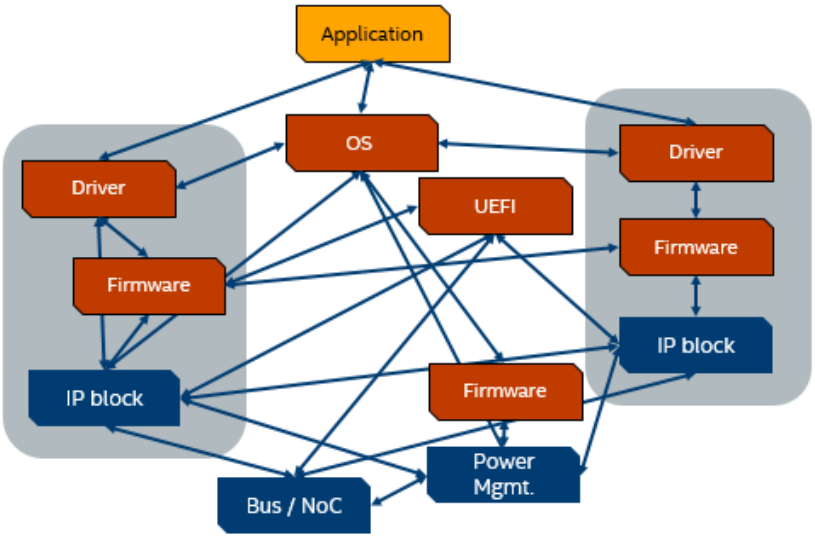
この記事は、インテル® デベロッパー・ゾーンに公開されている「The More the Merrier – Building Virtual Platforms for Integration」の日本語参考訳です。



「統合」は、システムのエンジニアリングと構築において最も難しい分野の 1 つであることは間違いありません。異なるハードウェア、ファームウェア、ソフトウェアを組み合わせてシステムを構築すると、さまざまな「興味深い」問題が発生する可能性があります。コンピューター・システムとソフトウェア・システムを構築する従来の方法では、すべての要素が魔法のように連携し動作する、「ビッグバン統合」が最後に行われてきました。この方法が適切でないことは誰もが分かっており、近年では継続的インテグレーション (英語) とボトムアップの小さなステップでの統合へと移行しています。

複雑な問題

コンピューター・チップとシステムオンチップ (SoC) の設計では、統合の問題を早期に発見し、コストのかかるシリコンの再設計を行わずに更新できるように、シリコンの前工程で統合する必要があります。この統合には、多くの要素と相互接続が含まれます。たった 2 つの IP ブロックを統合する場合であっても、次の図のように複雑になります。



各 IP ブロックには、通常、ブロックに統合するファームウェアがあります。統合された IP ブロックには、ソフトウェア・ドライバーへのインターフェイスがあり、このインターフェイスはオペレーティング・システムと統合します。いくつかのケースでは、ドライバーと OS からのアクセスはすべてファームウェアを経由し、別のケースでは、ファームウェア経由のアクセスに加えて、直接ハードウェアへアクセスすることもあります。各 IP ブロックは、直接またはネットワークオンチップ (NoC) やバス経由でほかのブロックと通信します。チップ上のすべてのブロックのアクティビティーは、電源管理ハードウェアおよびファームウェアが制御します。UEFI (Unified Extensible Firmware Interface) やその他の BIOS とシステムを起動するブートコードは、ハードウェアを確認して、起動し、そのいくつかを有効にするため、ハードウェアにアクセスする必要があります。IP ブロックのソフトウェア・ドライバーは、実際のファームウェアを IP ブロック上にロードする場合があるため、それらを起動する責任があります。

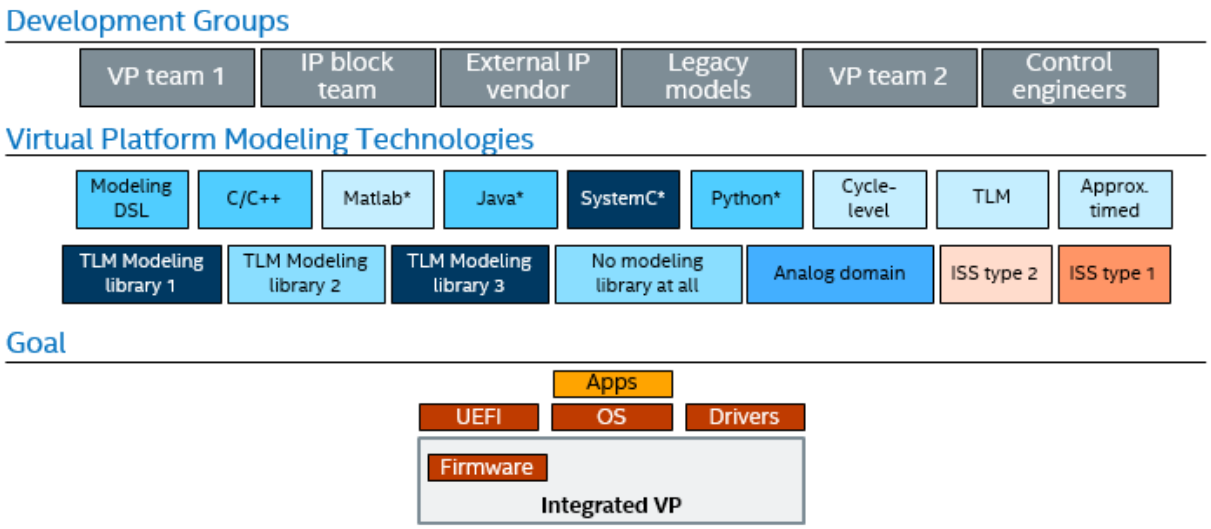
つまり、さまざまなコンポーネントのセットと多種多様なソフトウェア、ファームウェア、ハードウェアにわたる多数のシナリオとテストを検討する必要があります。

複雑さのシミュレーション

シリコンの前工程で統合するには、メイン・ソフトウェア・スタックを実行するメインコアからファームウェアを実行する IP ブロック内のプロセッサー・コアまで、すべてを含む完全なシステムを提供する仮想プラットフォームを構築する必要があります。

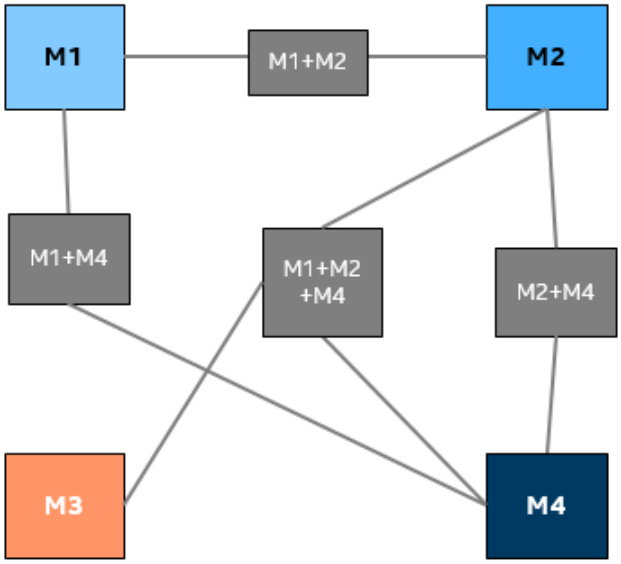
このようなモデルは、全体的な仮想プラットフォーム開発の一環として構築できますが、多くの場合、IP ブロックチームや IP ベンダーの既存モデルを使って素早く構築することができます。モデルの提供元はさまざまで、多種多様なフレームワークを使用して記述されています。一般的なケースとして、グラフィックス、メディア、ネットワークなどの関数のアクセラレーターがあり、これらのケースでは、ハードウェア設計者はシミュレーターを使用してシステムを設計する傾向にあります。コンピューター・システムのほかにも、物理学や機械工学の分野において、完全なシステムを構築するためシミュレーターが使用されています。

以下に全体像を示します。



さまざまなグループがあり、それぞれ独自のモデリング手法で構築されたモデルを提供しています。これらを 1 つのプラットフォームに統合して、実際のソフトウェアを実行し、ソフトウェアに実環境であるかのように認識させる必要があります。

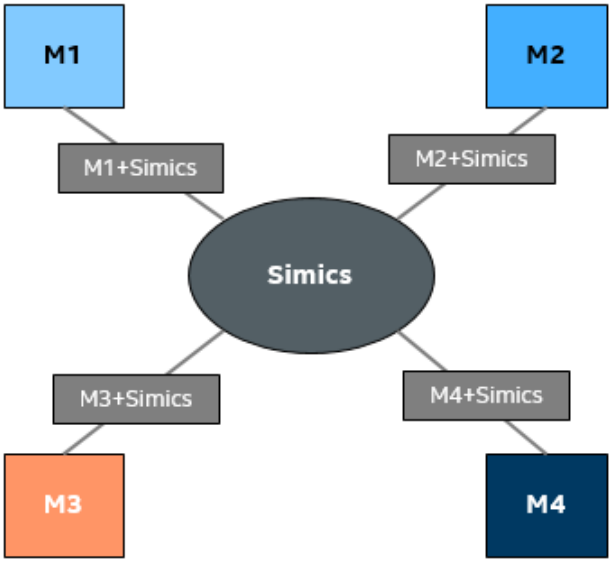
これを実現する 1 つの方法は、特定の問題に取り組むため、異なるシミュレーター間のアドホックポイントを作成します。これは、次のようにいくつかの個別の統合を作成する可能性があります。



この方法は、少数のモデルには適用できますが、モデルの数が増加すると統合の数も増加するため、すぐに利用できなくなります (理論的には、モデルを 1 つ追加するごとに、可能な統合の数は倍以上に増えます。例えば、4 モデルの場合は 11、5 モデルの場合は 26 になります)。

Simics* をベースに統合を構築する

より実践的な方法は、各モデルから Simics* のような共通のベースへのアダプターまたは統合を構築することです。



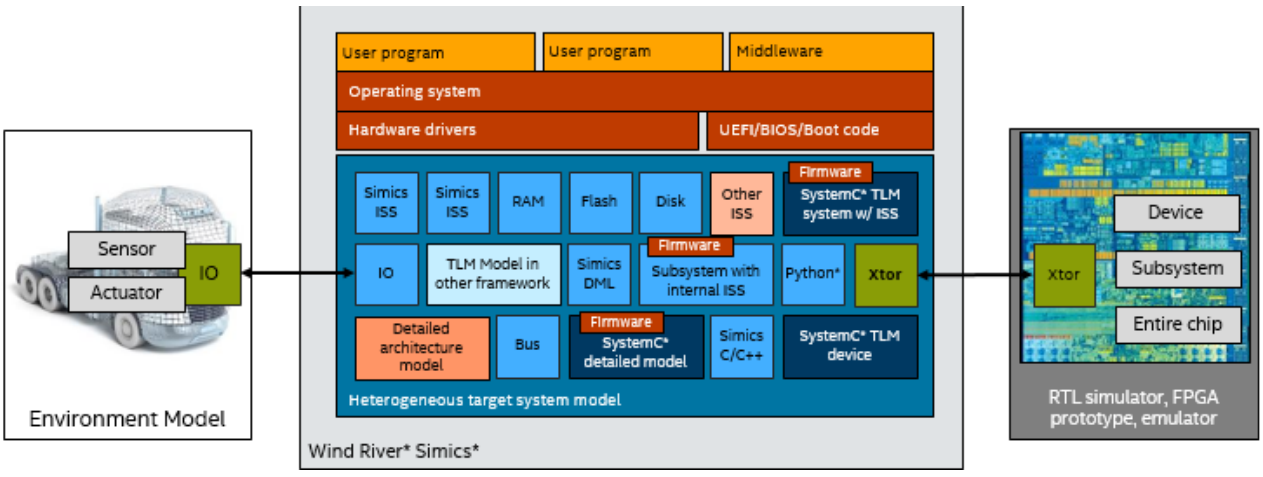
このケースでは、モデルごとに 1 つの統合を構築するだけで済み、各モデルの統合を共通のベース・シミュレーターである Simics* に統合することで、任意のモデルの組み合わせが得られます。このアプローチにより、任意のモデルの組み合わせを配備することが可能になり、ブロック間の統合テストが容易になります。

Simics* フレームワークは、誕生当初から採用されているいくつかのコアテクノロジーにより、共通のベースに適していることが実証されています。

1. Simics* は、モデリング向けに複数の言語をサポートするように設計されています。C、C++、Python*、SystemC* (英語) はすべて標準言語としてサポートされています。さらに、MATLAB*、CUDA*、Java* などのほかの言語のコードを Simics* モジュールに統合しているユーザーもいます。
2. Simics* は、モデルのリンクとモデル間のインターフェイスにホスト・プラットフォームの C レベル ABI を使用します。この ABI は、使用されるコンパイラーに関係なく同じであるため、特定のコンパイラー・バージョンでモデルをビルドする必要がありません。一般に、システムモデルは、さまざまなコンパイラーとコンパイラー・バージョンでコンパイルされたモデルを使用してビルドされます。
3. Simics* は、モデルをバイナリー・モジュールにパッケージ化します。これにより、ユーザーのシステム構成からモデルビルドを切り離します。モデルのユーザーは、ソースコードにアクセスしたり、モデルがどのようにビルドされたかを知る必要はありません。
4. モデル間の通信は Simics* インターフェイスのセットを使用して実行されます。これは、共通の規格による相互運用性を提供し、モデルが Simics* プラットフォームや PCIe* および Ethernet* ネットワーク・シミュレーションなどの Simics* 機能にアクセスできるようにします。ただし、Simics* インターフェイスは変わらないわけではありません。Simics* インターフェイスも、新しいユースケースに対応するため、必要に応じて進化してきました。
5. Simics* API は、機能が豊富であり、すべてのユーザーが利用できるため、Simics* 製品に依存することなく強力なコードを記述して、特定の機能をサポートすることができます。

Simics* を利用して構築されたヘテロジニアス仮想プラットフォーム

結果として、Simics* プラットフォームは、多種多様なコンポーネントのセットで構成されます。通常、Simics* を直接使用してモデル化されたベース・プラットフォームがあります。これは、例えば、インテル® プラットフォームのメイン・プロセッサー・コアとチップを提供します。Simics* プラットフォームは高速で、オペレーティング・システムとソフトウェア・スタック全体を起動し実行することができます。



そして、追加コンポーネントをベースに統合したり、ベース・プラットフォームのコンポーネントをより詳細なモデルに置き換えたりします。多くのケースでは、デバイスまたはサブシステムの Simics* モデルとより詳細な統合シミュレーターの両方があります。

例えば、オーディオ・サブシステムは、システムの残りの部分へのインターフェイスで高速な関数モデルとしてシミュレーションすることも、プロセッサー・コアと検出されたデバイスのモデルを含み、ドライバーとファーム

ウェア間の統合テストを行うためファームウェアを実行する、完全な「ホワイトボックス」モデルとして使用することもできます。

上の図の右側では、別の例として、プラットフォームの一部をエミュレーター、シミュレーター、または FPGA プロトタイプ上の実際の RTL (レジスター転送レベル) に置き換えています。このケースでは、トランザクターを使用して RTL をトランザクションレベル・シミュレーターに接続します。高速に実行できるように、通常 RTL は外部ハードウェア・ボックスとして実行します。

上の図の左側では、Simics* が物理ドメインの環境モデルと統合されています。このケースについては、http://jakob.engbloms.se/archives/2116 (英語) で詳しく説明しています。

パフォーマンスは?

多種多様なパーツを統合して仮想プラットフォームを構築することは、パフォーマンスに影響する可能性があります。しかし、通常、統合そのものが原因となることはありません。経験上、インターフェイス間の変換がシミュレーションのパフォーマンス全体に与える影響はわずかです。パフォーマンスの問題を引き起こす可能性がある原因は 3 つあります。

1 つ目は、統合するシミュレーターの数が増加すると、シミュレーターを実行するのに必要な作業も増加します。つまり、すべてのパーツが最適化されていても、統合システムの実行速度は低下します。1 つのプロセッサーをシミュレーションする場合と、100 のプロセッサーをシミュレーションする場合では、避けられない違いがあります。この問題を軽減するには、常に最も完全で最も複雑なプラットフォームでテストするのではなく、テストごとに適切な設定を使用します。

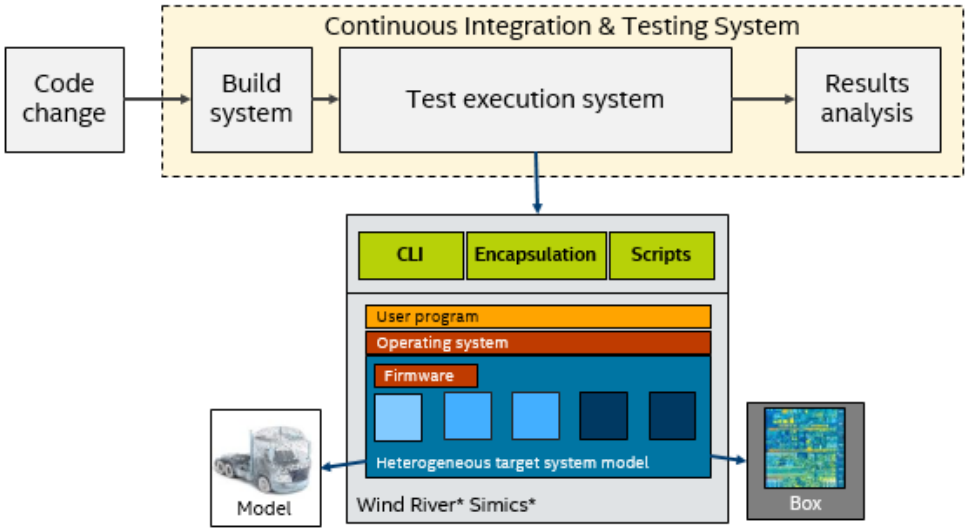
2 つ目は、一部のモデルは単体では問題なく動作し、統合すると最適に動作しない場合があります。例えば、多数のシミュレーター・イベントでモデルを実行する場合、単体では問題がなくても、大きなシステムに統合するとパフォーマンスが低下することがあります。ほとんどの場合、この問題は原因が特定できれば簡単に解決できます。

3 つ目は、統合されたモデルは設計上の理由から統合プラットフォーム全体のパフォーマンスが低下することがあります。サブシステムのマイクロアーキテクチャーを忠実にモデル化するサイクルレベルのモデルは、トランザクションレベルのモデルよりもはるかに遅く (1,000 から 10,000 倍遅く)、プラットフォーム全体のスピードに影響します。この問題を解決するには、詳細なモデルと高速なモデルを組み合わせて、詳細が不要な場合は高速なモデルを実行することで、詳細なモデルの実行時間を最小限に抑えます。

概して、多種多様なシミュレーターの統合自体は、ホモジニアスなプラットフォームを構築する場合と比較して、必ずしもパフォーマンスを低下させるわけではありません。

仮想プラットフォームを別のシステムへ統合する

一見すると明らかではない統合の別の側面として、仮想プラットフォーム自体もより高いレベルのフローに統合されます。ほとんどのケースでは (少なくともシミュレーションの実行時間と実行回数において)、VP モデルはユーザーのデスクトップでインタラクティブに実行されるのではなく、自動テストシステムやランチャーシステムから実行されます。



そのようなケースでは、すべてのモデルをカプセル化できる共通のシミュレーター・プラットフォームが非常に役立ちます。高度なシステムは、使用されるモデルの内部構成に関係なく、単一のツールを使用するように記述できます。一貫性のある自動化およびカプセル化インターフェイスを提供することで、Simics* は異なるターゲットとターゲットの異なる構成で再利用可能なテスト・インフラストラクチャーの構築を可能にします。ターゲットの仮想プラットフォームでは、実際には Simics* のネイティブモデルは必要ありません。しかし、Simics* をベースに構築されたインフラストラクチャーに適合させるため、Simics* と統合することは有益です。

そのような統合は、シリコンの前工程から後工程、配備、保守まで続きます。現代のソフトウェア配備において自動化は重要であり、ここで紹介したように仮想プラットフォームが非常に役立ちます。

まとめ

システム・インテグレーションを早期に実現するには、仮想プラットフォームとの統合をサポートする必要があります。必要なソフトウェア (特にファームウェア) をすべて実行できる完全なプラットフォームを構築するため、仮想プラットフォームは通常、多種多様な既存のモデルとパーツを統合して構築されます。この統合により、すべてのソフトウェアを実行するのに必要な詳細を備え、ほかのシステムへ一貫したパッケージを提供する仮想プラットフォームを素早く構築できます。

著者のほかの記事 (英語) もご覧ください。

コンパイラーの最適化に関する詳細は、最適化に関する注意事項を参照してください。