Bluespec SystemVerilog (BSV)を使用したタイミングクロージャ、チップ面積の最適化、ECOの効率的な実現

設計者はASICやFPGAプロジェクトにおいてVerilogからレイアウトまでの設計フローに馴染んでいます。Bluespecに初めて触れる場合、以下のような疑問を抱くでしょう。

1. ソースコードが Verilog でなく BSV を使用する場合、ECO はどうするのか？
2. BSV の設計ではタイミングクロージャはどうやって実現するのか？
3. BSV の設計でどのようにして面積の最適化を行うのか？

簡単に言うと、多くのケースでははるかに速く実現するにも関わらず、現在行っていることと大きく違うことはない、ということです。BSVからどのようにしてVerilogが生成されるのかということを設計者が理解すると、以下の実用性がわかります。

1. タイミングやエリアを改善するために異なるアーキテクチャやマイクロアーキテクチャ、機能、実現方法を採用したVerilogコードを生成するためにBSVを変更すること
2. ソースコードの対応する行を知っていて、VerilogからBSVにパスを割り当てること

例えば、論理合成ツールが生成したクリティカルパスをVerilog RTLに戻すことが困難であるということを、設計者はすでに知っています。私たちはアーキテクチャを理解し、Verilog RTLから合成されたゲートに渡される信号名から判断することによって、これを行います。例として、”dataA”という名前のレジスタから複数のゲートを通って”dataB”のレジスタにつながる長いタイミングのパスを考えます。Verilogファイルの正確なパスを容易に探すことができるようにゲート間の信号名をモジュール名に組み込むことがあります。論理合成がどのようにRTLを論理的に取り扱ってAND/ORゲートに変換するのかということを私たちが理解する必要がありますが、私たちは基本的にそれを受け入れています。20年前、ちょうど論理合成ツールが市場に出たころ、コンピュータ産業の多くは、私たちがツールの仕組みを理解する必要があるということは、RTLから論理合成することが決して実用的でないことを意味すると考えられました。カスタムレイアウトや非常にタイミングの厳しいロジックを設計しているのでなければ、テクノロジ固有のゲートレベル設計を現在行っている企業は明らかにごくわずかです。私たちはRTLからの論理合成ツールがどのように機能するのかということを理解したのです。

タイミングクロージャと機能デバッグ

私たちは、BSVにおいて同様のアプローチをとります。実際問題としてネットリストではなく、Bluespecは可読性の高いVerilogを生成するため、このプロセスは論理合成のときよりも非常に簡単です。

まず、Bluespecのコンパイラは、入力に対して一意に出力が決まるように設計されています。同じ入力が同じコンパイルオプションを使用して何度コンパイルされても私たちは同じ結果を得ます。（コンパイルオプションによって、信号線の最適化などが可能ですが、マイクロアーキテクチャなどは変更できません）

次に、コンパイラはユーザがBSVに記述していないロジックは追加しません。これは、ユーザが設計のマイクロアーキテクチャに対する究極の制御性をもっていることを意味します。例えば3つの乗算器が直列につながって、タイミングパスが長すぎる場合（これはゲート合成では最適化できません）、この問題の唯一の解決方法はパスのマイクロアーキテクチャを変更することです。Bluespecはブロックの機能を変更しません。ユーザが明示的にコードに記載しない限り、レジスタやモジュールは追加されません。機能を変更するためには、設計者が明示的にBSVコードを書き換え、Verilogを再生成する必要があります。タイミングの問題に取り組むとき、必要であれば設計者がBSVを変更してVerilogを変更します。もしロジックを分割する場合やレジスタを追加してクリティカルパスにレイテンシを加える場合、もしくはシャドウレジスタを追加する場合、BSVを使用することによって全てのアトミックアクションとコンパイラのcorrect-by-construction機能によって高速に、そしてエラーを発生させることなく変更することができます。Verilogベースの設計では、私たちはゲートレベル出力を変更せずにソースコードの方を変更します（レイアウトレベルのECOを除いて。本件は、後述します）。

さらに、BSVの信号名は生成されるVerilogに反映されます。Bluespecコンパイラは、生成されるVerilogのモジュールやレジスタ、BSVのインスタンス名を元にしたポートなどの名前をとても慎重に作成します。例えばBSVの”cfgStatus”という名前のレジスタは、Verilogでも”cfgStatus”という名前でレジスタが生成されます。BSVで”ram16k”の名称のモジュールは、Verilogコードで”ram16k”のインスタンス名のモジュールが生成されます。BSVのインタフェースから生成されるVerilogの入力と出力は一意に決まります。設計者は、制約ファイルやBSVブロックのピンのタイミングを規定するSDCファイルにおいて、タイミングレポートや生成されるVerilogコードが常に同じピン名であるということを確認することができます。レジスタ名は常に同じ名前に保たれます。

生成されるVerilogにおいて、いくつかの名前がコンパイラによって作られます。ちょうど論理合成がゲート名を作成するように、BSVでいくつかの合成変換（例えばif-elseをmuxに構成すること）によって信号名が作成されます（例えば、if条件名が使用されます）。そして、（例えばSDCファイルのパスを定義するために）BSVから特定の名前に固定することが必要であれば、BSVの定義”probe”を使用することができます。これは、BSVからVerilogに直接信号名が渡される以外には論理的な影響はありません。

コンパイラはとても可読性の高いVerilogコードを生成します。コンパイラは、一種の競合調停ロジックを生成します。そのロジックは”CAN_FIRE”や”WILL_FIRE”信号がルール（BSVコード内の”rule”に相当するロジック）の実行を制御します。これらの信号名はまた、それらが由来するルール名を含んでいます。”CAN_FIRE_rl_watchout”を見ると、ルール”watchout”の競合調停ロジックであることが分かります。一般に誰でも、生成されたVerilogからBSVコードにゲートを困難なくマッピングすることができます。

最後に、Bluespecを使用することで、設計者はタイミングや面積の要求に合わせるために機能性を損なわず、信頼性をもって大きな機能変更やアーキテクチャの変更をすることができるようになります。Bluespecの設計者が特定のブロックのレイテンシや機能を変更する場合、その変更は他のブロックには影響しません。

ECOs

BluespecがどのようにECOをサポートするのか、とよく尋ねられます。FPGA設計では、設計が頻繁に再コンパイルされるため、このことはあまり考慮されません。ASIC設計では、ゲートはワイヤ負荷モデルのみを使用する論理合成ツールを使用する場合、設計の実際のレイアウト段階においてレイアウトの金属配線のみでタイミング変更が行われるように要求するかもしれません（これによって数時間から何日間ものチップの再配線の時間短縮ができます）。生成されたVerilogの機能を変更せずに大きなバッファをタイミングパスに加えたり、容量を軽減するためにシャドウレジスタを使用することをイメージするのが最も簡単なECOです。手修正のレイアウトネットリストと手修正の生成されたVerilogブロックを比較するためにフォーマル等価性検証ツールを使用することができ、Bluespecコンパイラが生成したVerilogとフォーマル比較を実行できます。多くの変更が設計レビューによって単純に検証することさえ可能です。多くのASICデザインハウスは現在、ブロックのタイミングを明らかにし、最終的なレイアウトに達することを確かにするため、（MagmaやSynopsys Physical Compilerのような）レイアウトと合成が統合されたツールを使用しています。

より困難なケースでは、レイアウトによってロジックが変更される場合で、合成されたゲートをVerilogに戻す必要があります。ゲートが生成される前においてもBluespecコンパイラが多くの一般的な小さなエラーを検出するため、私たちはこのことがBluespecを使用する設計において一般的でないことがわかります。もし機能が変更された場合、VerilogそしてBSVに戻したいと考えます。

Bluespecコンパイラが生成したVerilogは可読性が高いので、この機能的な変更をどこに行うべきであるか見つけることができます。これは、熟慮の上、BSVソースが機能的にどのように変更されるのかを理解して実行する必要があります。これは、Verilog設計における変更と同様のコンセプトです：あなたはどのように、なぜ変更するのかを知っている必要があります。あなたが望めば、同じメソドロジは、簡単なタイミングフィックスにも適用することもできます。設計者が行うことは以下のとおりです。

設計者はもともと2つの基本となるファイルを持っています：オリジナルのBSVファイル（これをMod.bsvとします）と、Bluespecコンパイラの生成したVerilogファイル（これをmkMod.vとします）。

1. 設計者がロジックの変更を加えるためにオリジナルのBSVファイルのコピーを作成する。コピーしたファイルにロジックの変更を加える（変更後のファイルをModNew.bsvとします）
2. 設計者がModNew.bsvのコンパイルを実行し、新規に生成されたVerilog（これをmkModNew.vとします）
3. 設計者はECOによりVerilogファイルに加えるべきロジックの変更を理解してmkMod.vのコピーに変更を加える（変更後のファイルをhandEditedMod.vとします）

設計者は現在、handEditedMod.vとmkModNew.vのロジックの比較を実行します。いったん彼らがロジックの比較をして同じであれば、それからは新規のBSV ModNew.vをソースとして使用することができます。もし将来変更が必要になった場合、新しいBSVを変更し、ソースコードとして使用し続けることができます。

お問い合わせ先：

CYBERNET

サイバネットシステム株式会社
新事業統括部
〒101-0022 東京都千代田区神田練塀町３富士ソフトビル
Tel: 03-5297-3295　Fax: 03-5297-3637

e-mail: bluespec@cybernet.co.jp
http://www.cybernet.co.jp/bluespec/