7B－2

階層設計における遅延制約情報の自動算出手法

辻　慎一、山本　孝司、黒橋　学、金子　信之、桑原　教雄、吉川　浩†
ＮＥＣソフトウェア北陸、　†ＮＥＣ(株)

１．はじめに

近年、ますます大規模化するＬＳＩの開発において、階層設計法は必要不可欠なものとなりつつある。この階層設計において遅延を考慮しながら論理合成[1)]を行う場合には、各階層の設計単位(以後、サブサーキットと記す)毎に遅延制約を定め、その制約内に遅延を収めるよう設計を進めることになる。

今回提案する遅延制約情報の自動算出手法は、階層表現されたネットリストと、チップレベルに展開された(以後、チップ・フラットと記す)遅延情報とから、各サブサーキット内の論理は、どの程度最適化可能かを考慮して、各サブサーキットの遅延制約を自動で作成することを特徴としている。これにより、遅延制約のメンテナンス・フリーを実現し、効率的な遅延を考慮した階層設計を実現できた。本稿ではその概要を報告する。

２．階層設計法における問題点

階層設計法にて遅延を考慮する場合、各サブサーキットは指定された遅延制約を満たすよう設計される必要がある。

遅延制約を自動で算出する従来の方法としては、上位階層の遅延解析結果から遅延制約を作成する方法が存在する。この方法は、図１．に示すフローのように、まず、チップ・フラットな論理接続情報(ネットリスト)を作成し、そのデータに対して遅延解析を行う。遅延解析にて、エラーを検出した場合、エラーとなるパスを含む下位階層に対して、チップフラットな遅延解析結果より、その下位階層に対する遅延制約を作成する。ここでの遅延制約は、サブサーキット内でどの程度の論理の最適化が可能かを考慮せず、処理対象サブサーキット以外の遅延時間を計算し、クロック周期との差と求めて、その差を処理対象サブサーキットの遅延制約としていた。この制約にもとづいて下位階層の論理を最適化する。この一連の処理を遅延エラーがなくなるまで繰り返すことになる。このように従来の方法では、下位階層の論理を変更する度にチップ・フラットな遅延解析と、遅延制約の再作成が必要である。特に、遅延エラーがなかなか収束しない場合には、上記の繰り返しに多くの時間がかかるという問題があった。

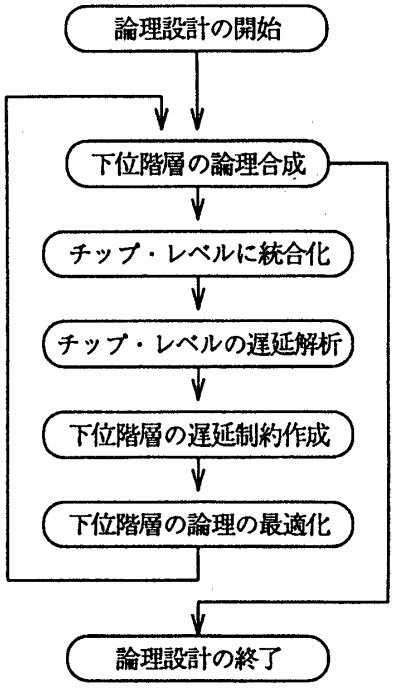

図１．論理設計フロー

３．遅延制約の算出方法

２．で述べた問題は、以下に述べる方法で遅延制約を算出することにより解決が可能である。

チップレベルの遅延解析でエラーとなるパスは、通常、複数のサブサーキットにまたがっている。この時、

A Technique for Calculating Subcircuits Timing Constraints in Hierarchical Desigen.
Shinichi TSUJI, Manabu KUROHASHI, Takashi YAMAMOTO, Nobuyuki KANEKO, Norio KUWAHARA, Ko Yoshikawa †
NEC Software Hokuriku, Ltd., †NEC Corporation

それぞれのサブサーキット内でどの程度の論理の最適化が可能かを考慮して遅延制約に緩急をつける。この遅延制約にもとづいて論理を最適化することにより、サブサーキット内で遅延が満足する限り、チップ・フラットな遅延解析と遅延制約の再作成を不要とすることができる。経験的に、ゲート段数の多い、すなわち、遅延時間の大きい論理回路ほど、論理を最適化できる可能性がある。従って処理対象サブサーキットの境界における遅延制約は、そのサブサーキットおよびそれと対をなすサブサーキットの最適化可能な論理の遅延時間の比で決まるとし、以下の式で算出する。

遅延制約：Const ＝ Dinner ／ (Dinner ＋ Douter) × Cycle

Dinner：処理対象サブサーキット内で最適化が可能な論理の遅延時間

Douter：処理対象サブサーキットと対をなすサブサーキット内で最適化が可能な論理の遅延時間

Cycle ：クロック周期

本式を用いて、以下の手順により、各サブサーキットの遅延制約を求める。

(1)処理対象サブサーキットの境界を経由するパスを列挙する。

(2)列挙した各パスについて、処理対象サブサーキット内における論理の最適化が可能な部分と不可能な部分の遅延時間を求める。

(3)処理対象サブサーキット内で論理の最適化が可能な部分の遅延時間と、処理対象サブサーキットと対をなすサブサーキット内で最適化が可能な論理の遅延時間とから、上記算出式により該当パスに関する遅延制約を算出する。

(4)上記(1)～(3)を繰り返し、各サブサーキットの遅延制約を算出する。

4．実回路での制約

実際の回路に適用するにあたっては、以下の項目について論理の最適化が不可能な部分の遅延時間として扱う必要があった。

(1) テスト回路等の論理が固定されている回路の遅延時間。

(2) 階層間のネットの配線遅延時間。本配線遅延時間はフロア・プランを変更しない限り改善できない。

(3) フリップ・フロップ回路の動作に必要なマージン(セット・アップ時間とホールド時間)。

(4) フリップ・フロップ回路間のクロックの到着時間差(クロック・スキュー値)。

5．評価結果

本手法を小規模なＬＳＩに適用した。従来、遅延制約はチップレベルで遅延エラーを検出する度に作成していたが、本手法により、その工数を大幅に削減することができた。また、4．実回路での制約で挙げた点を考慮することにより、各階層に対して無理のない遅延制約を定義することもできた。

6．おわりに

本稿では、遅延制約の自動算出手法について述べた。実際の回路に対して適用した結果、本手法は、後戻り処理の少ない階層設計を可能にした。ただし、必要なデータとしてチップ・フラットな論理接続情報が必要なため、かなり後工程にならないと遅延制約を自動算出できないという問題が残る。しかし、各下位階層の遅延制約の算出以後は、下位階層内で処理が閉じるため、チップレベルの遅延制約の再作成処理の回数を大幅に削減することが可能となる。なお、本手法はＥＷＳ４８００等のＬＳＩ開発に用いられ、多大な効果をあげている。

参考文献

1) Ko Yoshikawa, et al., "Timing Optimization on Mapped Circuits",
28th ACM／IEEEE Design Automation Conference, pp112-117, 1991