遅延値が変更可能なコンパイル方式シミュレーション

井上, 恒司 / INOUE, Koji / 若林, 哲 / 檀, 良 / DANG, Ryo / WAKABAYAHI, Satoshi

(出版者 / Publisher)
法政大学工学部
(雑誌名 / Journal or Publication Title)
法政大学工学部研究集報 / 法政大学工学部研究集報
(巻 / Volume)
32
(開始ページ / Start Page)
7
(終了ページ / End Page)
12
(発行年 / Year)
1996-03
(URL)
https://doi.org/10.15002/00003810

遅延値が変更可能なコンパイル方式シミュレーション

井上　恒司*・若林　哲**・檀　良***

A Compiled-Code Logic Simulation Using Selectable Delay Values

Koji INOUE, Satoshi WAKABAYASHI and Ryo DANG

Abstract

In this paper we describe a new management method for delay condition in compiled-code logic simulation. In a conventional logic simulation, the delay condition is uniformly managed for all devices in the circuit. However our management is for every device using extended delay table and mode parameters. The users can implement various simulations without recompiling.

§1 はじめに

半導体プロセスの微細化によってLSIの集積度は年々増加し、また動作も高速になってきている。この物理的な作用は論理回路における遅延値に大きく影響する。このため回路の検証においてはタイミングに関してより高精度なものが必要となる。

従来の論理シミュレーションの遅延に関する条件はベストケース、ワーストケースなどをシミュレーションの実行ごとに回路全体で一律に管理していた。我々はこの条件をインスタンスごとに管理する事で、様々なケースのシミュレーションを可能とした。シミュレーションはCコンパイル方式であるが、遅延値の変更に際しての再コンパイルの必要はない。

§2 シミュレータと回路のモデル

2.1 シミュレーションモデル

シミュレーション方式はFig. 1のような流れのCコンパイル方式である。入力としてVerilogHDL

*大学院電気工学専攻
**東芝マイクロエレクトロニクス(株)
***電子情報学科

とVHDLのゲート記述のサブセットをサポートし、トランスレータを通して必要とするデータ構造にする。このC言語表現の回路とシミュレータ本体さらにライブラリをコンパイル・リンクし直接実行可能な形式にする。このプログラムの実行時にテストパタンとシミュレーション条件を与えることでそれに応じた結果を得る。

シミュレーションのアルゴリズムは以下のような一般的なイベント駆動方式［1］とした。

1. 次に起こるイベントを取り出す
2. そのイベントの結果を評価する
3. 新しいイベントが発生すれば遅延値を考慮し登録する
4. 終了でなければ 1. へ

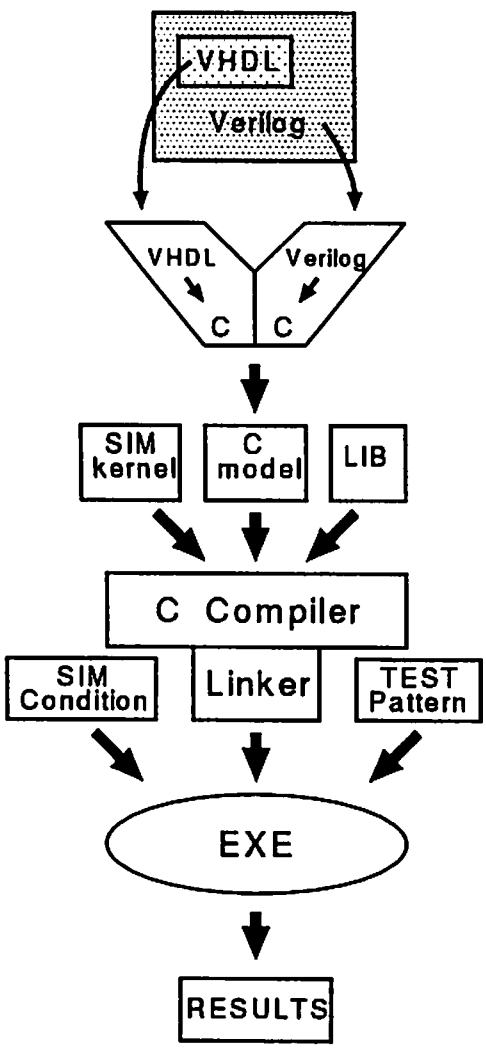

Fig. 1：全体の流れ

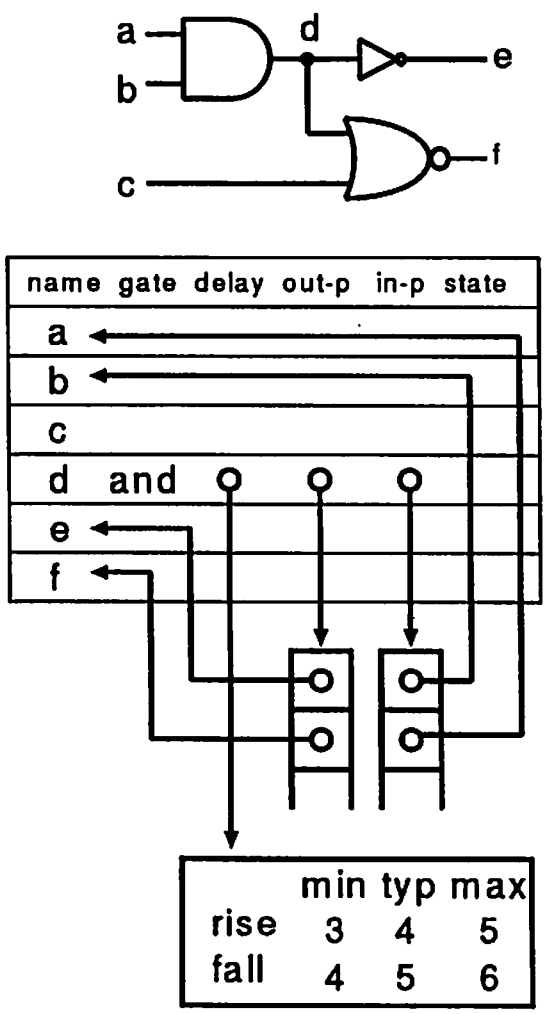

Fig. 2：回路とデータ構造

2.2　回路モデル

回路はゲートレベルを対象とし、外部入出力とゲート間の結線関係をVerilogHDLおよびVHDLで記述し、トランスレータを通すことで、我々の必要とするC言語のデータとしてFig. 2の様に構造化する。この際構造データに遅延のモードを表すパラメータを加え、必要であれば遅延値の参照先の拡張を行う。このパラメータは各インスタンスがそれぞれの値を持ち、シミュレーションの条件がこの値に反映される。

§3　遅延値参照先の拡張とモードパラメータ

	min	typ	max
rise	3	4	5
fall	4	5	6

Extension

	min	• • • •	typ	• • • •	max
rise	3	3.2 3.4 3.6 3.8	4	4.2 4.4 4.6 4.8	5
fall	4	4.2 4.4 4.6 4.8	5	5.2 5.4 5.6 5.8	6
PAR	0	1 2 3 4	5	6 7 8 9	10

Fig．3：　データの拡張とパラメータによる参照

C言語のデータを作成する際の参照先テーブルの拡張の様子をFig. 3に示す。この例では最小・標準・最大の各遅延が与えられた場合である。最小と標準、標準と最大の各間を4点で線形に補間している。もし標準遅延の1値のみ与えられた場合は、最小値・最大値の仮定値をを標準値のそれぞれ50%と150%として間の4点を補間する。なお最小値・最大値の仮定値と補間する点数はトランスレータで変更可能である。この例のように各間を4点で補間した場合、はじめは標準遅延を指すようにパラメータの初期値を5にしておく。また、補間によって有効桁数が増えるときは、整数位になるまでスケーリングしておき、シミュレーション時にそれを考慮する。

遅延のモードパラメータはインスタンスごとに値を持っているので、遅延のテーブルの参照時にこの値によってインスタンスごとに異なる条件下の値を従来よりも細かい指示で取る事が可能となる。

§4　パラメータのコントロール

シミュレーション条件をインスタンスごとに管理するという事は、インスタンスごとのパラメータ値を1つずつ制御する事にほかならない。すなわち、どのシミュレーション時刻にどのインスタンス

のパラメータをどの値にセットするかをシミュレーション条件として実行時に与える。実際にはこれらを一括して条件ファイルに記述しておき、シミュレーションの実行時に入力する。シミュレータはシミュレーションの開始前にすべてのパラメータ変化時間をスケジューリングし、シミュレーション中の該当時間にパラメータの値を変化させる。パラメータ変化のコマンドと信号変化のイベントが同時刻に起こる場合、パラメータの変化を先に評価する。パラメータ制御を考慮したアルゴリズムは以下のようになる。

1. 次に起こるイベントかコマンドを取り出す
2. 次に起こるのがコマンドなら、その処理をして1. へ
   イベントなら3. へ
3. そのイベントの結果を評価する
4. 新しいイベントが発生すれば遅延値を考慮し登録する
5. 終了でなければ 1. へ

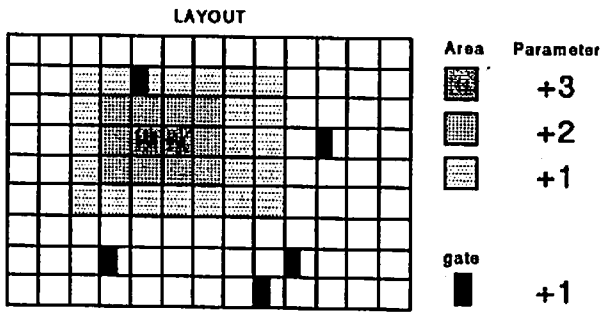

Fig. 4　パラメータ変更の例

このパラメータ制御の指示はユーザが目的に応じて与えることができるが、その一例はたとえばFig. 4に示すように、レイアウトの配線の混雑度等の物理的な要因による遅延値への影響を考慮したシミュレーションの場合やあるゲートの設計変更によるチップ上のそのゲートすべての遅延値の変更の場合である。

§5　シミュレーション結果

シミュレーションの結果を比較した項目は次のものである。

1. 本シミュレータ独自の機能を取り去った通常のシミュレーション
2. 本シミュレータ独自の機能を組み込んだが、全く使用しない場合
3. 本シミュレータ独自の機能を組み込み、使用した場合

パラメータの制御はシミュレーション中に全パラメータを3回変えるようにした。
なお、比較項目としてCPU時間×メモリ使用量を計算コストとした。

［実験1］

シミュレーションの実験回路として4ビットコンパレータ(25ゲート)を用いた。テストパターンは入力8本の全ての組合せである256パターンとした。

	Events	CPUtime(s)	Mem(KB)	CPU×Mem
1.	2836	0.708	127.0	1.00
2.	2836	0.846	127.0	1.19
3.	2484	0.873	127.0	1.23
3.	2934	0.883	127.0	1.24

これらの結果から、機能を新たに組み込むことにより約19%のパフォーマンスの低下がみられた。さらに、機能を使用した場合は、組み込まなかった場合に比較し約24%のパフォーマンスの低下がみられた。 3. でイベント数が変わる場合というのは遅延値を変更することにより、これまで合っていたタイミングがずれることにより新たにイベントが発生する場合と慣性遅延のために無くなる場合である。たとえばFig. 5に示すように、これまで同じだったn0とn1の遅延が異なる場合で、経路1と経路2のタイミングがずれる場合がある。

［実験2］

4ビットコンパレータを10個詰め込んだ回路(250ゲート)で、テストパターンは実験1の場合と同じものを用いた。

	Events	CPUtime(s)	Mem(KB)	CPU×Mem
1.	28360	6.616	214.4	1.00
2.	28360	6.822	227.0	1.09
3.	27042	8.038	230.8	1.31

この結果から250ゲートの場合では機能の組み込みにより約9%の計算コストの増加がみられる。機能を組み込んで使用した場合は組み込まなかった場合に比較して約31%の計算コストの増加がみられた。

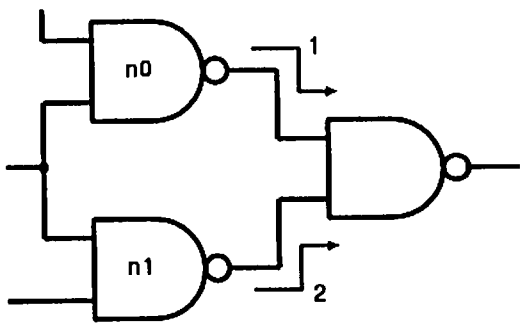

Fig. 5　タイミングがずれる場合

実験1、2ともパラメータの操作はシミュレーション中に全素子のパラメータを3回変更している。パラメータの操作が多い場合には計算コストの増加が多いと考えられるが、今回のようにシミュレーションでパラメータの操作は、全素子に対して数回程度、またはある特定のパラメータを細かい時間ごとに変更する場合が実際的で、全素子のパラメータを細かい時間ごとに変更する場合はあまりないと想定している。

［実験3］

・変換速度とコンパイル速度の比較

実験回路が1個の場合と10個の場合で、VerilogHDLからCへの変換に要する時間とシミュレータを作成するためのコンパイル時間の測定結果を示す。

Gates	Translate	C Compile
25	0.16(sec)	5.81(sec)
250	1.10(sec)	20.80(sec)

この変換とコンパイルは1度行って実行形式のファイルを作成すれば回路の接続等の変更が無いかぎり2度と実行する必要はないので、この時間を短縮化する努力はシミュレーションの実行速度の高速化に向けるべきである。

§6 ま と め

新たな機能の付加による計算コストの増加はあるものの、シミュレーション中における遅延値の変更による様々なシミュレーションや、遅延テーブルの拡張による詳細なシミュレーションが可能になった。テーブルの拡張とパラメータの導入により必要となるメモリの増加は導入前に比べて数パーセントである。計算コストの増加の原因は新しいアルゴリズムの2.でコマンド処理を特別扱いしている点にあると思われる。速度改善のためには、コマンドをイベントと同様に扱い、アルゴリズムを従来のシンプルな形に戻すことが有効であると思われる。

パラメータの制御の指示をシミュレーション開始時だけでなく、シミュレーションの履歴をもとにしてシミュレーション中に動的にすることでより実際の動作に近いシミュレーションが可能となる。

参 考 文 献

1) 石浦 菜岐佐，安浦 寛人，矢島 脩三："時間優先評価アルゴリズムによる論理シミュレーションの高速化"，情報処理学会論文誌，Vol. 26 , No. 3 , pp. 459－466 , May 1985