計算機アーキテクチャ　115－10
設　計　自　動　化　78－10
（１９９５．　１２．　１４）

分割処理スイッチレベルシミュレータ

影島淳　宇佐美公良　辻本順一
kageshim@dad.eec.toshiba.co.jp
東芝　半導体設計・評価技術センター
〒２１０　川崎市幸区堀川町５８０－１

タイミング検証を高速に行うものとして、スイッチレベルシミュレータがある。近年、開発回路の大規模化が進み、より高速で、より大規模な回路に適応できるシミュレータの開発が求められている。そこで、既存のシミュレータを大がかりな手直しなしに高速にする方法として、複数のネットワークで結ばれたＥＷＳを使用して分割処理を行う方法を開発し、評価を行った。その結果、従来と同じ精度を保ちつつ、実製品の455Ktr のマイクロプロセッサに適用しても約半分の時間で実行できることがわかった。

Switch-level simulation on communicating EWS's

Atsushi Kageshima , Kimiyoshi Usami and Jyun-ichi Tsujimoto
Toshiba Semiconductor DA & test engineering center
580-1,Horikawa-tyou, Saiwai-ku, Kawasaki-shi, Kanagawa, 210, Japan

This paper describes a switch-level simulation technique exploiting circuit partitioning and processing on multiple EWS's. Without modifying an existing simulator so much, this allows us to perform faster simulation for large-scale circuits. We applied this technique to a real microprocessor with 455K transistors. Simulation time has been reduced to half as much as the conventional simulator.

1　はじめに

我々は、半導体設計の際に行うスイッチレベルのタイミング検証に用いるシミュレータ開発を行っている。開発したシミュレータは実行速度がロジックシミュレータ並の速さを誇っており、社内で広く使われている。しかし、近年、開発回路の大規模化が進み、より高速でより対応回路の大きなシミュレータの開発が求められている。

抜本的なソフトの手直しによって、これを実現しようとすると多くの人手と時間、費用がかかると予想され難しいものがある。

一方、業務の効率化をはかるために、各部署でＥＷＳの導入が積極的に行なわれており、複数のマシンを所持し、それらをネットワークでつないで使用するという環境が当たり前になりつつある。そこで、既存のスイッチレベルシミュレータに大がかりな手直しをせずに実行速度を高速に、また対象回路を大きくする方法として、複数のマシンを同時に使って処理を並列に行なわせることを考えた。

本稿では、この考えを基に作成した分割処理シミュレータの紹介とその性能の評価結果を述べる。

2　従来のシミュレータの問題点

我々が開発し、使用していたスイッチレベルのタイミングシミュレータは、RSIM[1]をベースに独自に改良、機能追加を行なった高精度のイベント・ドリブン方式のシミュレータ[2]である（表１）。使用データは、spice 形式の回路記述ファイルをそのまま使え、パラメータも spice 用から変換できる。そのため、spice 利用者が簡単に使用することができ、また、spice と連動させることにより、デジタル／アナログ混載回路のシミュレーションにも使用することができる。

表１　従来のシミュレータ性能

速度	ロジック・シミュレータ並
精度	±２０%(対 spice)
使用メモリ量	160byte/Tr

このシミュレータは、広く社内で使用されているが、近年の開発回路の大規模化および、チップ全体のシミュレーションを行ないたいという要求に伴い、対象回路の大規模化とターン・アラウンド・タイムの短縮（シミュレーション時間の大幅な高速化）が求められている。

この要求水準を達成するには、使用マシンの新規購入、バージョンアップによる高速化や、プログラムの一部手直しなどによる高速化だけでは、到達することは非常に困難である。

そのため、抜本的なソフトの手直し、もしくは新規ソフトの開発が必要になるが、これらには、多くの時間と人手、費用が必要であり、更にユーザ・インターフェースの変化、蓄えられたデータの無効化という危険も含んでいるにも関わらず、必ず満足できる速度を達成できるという保証はない。

一方、業務の効率化をはかるために、各設計部署でＥＷＳの導入が積極的に行なわれてきており、複数のＥＷＳを所持し、それらをネットワークでつないで使用するという環境が当たり前になりつつある。

そこで、タイミングシミュレータのプログラムの大がかりな手直しをせずに実行速度を高速に、かつ、対象回路規模を大規模にする方法として、複数のマシンを同時に使って処理を並列に行なわせることを考えた。

つまり、対象回路を複数個の部分回路に分割し、それぞれの部分回路を異なるＣＰＵを用いて同期をとりながら並列にシミュレーションの実行をさせる方法である。

3　分割処理シミュレータ

3.1　前提条件

開発するに当たって、性能の目標及び、従来のシミュレータとの互換性を持たせるために以下の前提条件を課した。

- 従来のシミュレータのデータを大がかりな修正、変更なく使用する。
- 大規模回路（数百万 Tr）を対象したとき、従来版に比べ実行時間を短縮する。
- 一台当たりの使用メモリ数を少なくして、大規模回路に対応する。
- 従来のシミュレータと同じ精度を保つ。

3.2 回路分割方法

分割処理を行う際、まず問題になるのが、回路の分割方法である。

従来のシミュレータが使用していた spice 形式記述ファイルに大幅な手直しをすることなく、分割処理シミュレータで使用するという前提条件を満たすために、回路記述で使用されている階層構造に注目した。通常、回路記述は機能毎に記述された機能ブロックを使用して、階層構造になっている（図1）。

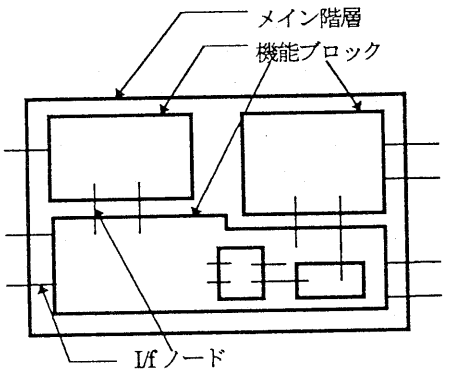

図1　回路記述構造

そのため、メインの記述は、数個の機能ブロックのみが記述されていることがほとんどである。

そこに注目し、このメイン階層にある機能ブロックをそのまま部分回路とすることにした。メイン階層の機能ブロック数が多い場合は、階層をもう一段増やして、複数の機能ブロックをグループ化することで対応できるからである。

なお、この部分回路化が終了したときに、複数の部分回路にまたがっているノード及び、外部入出力ノードを他のノードと区別してインターフェースノード（I/f ノード）と呼ぶことにする。

3.3 基本構造

前節のように分割した回路を用いて、分割処理シミュレータを実現するために、Server-Client 方式を使用した。

これは、

- Client:分割されたそれぞれの部分回路についてシミュレーションを実際に行うマシン。
- Server:Client にあてがわれた部分回路の入出力ノードとなる I/f ノードの値を調整、管理し、また、Client の同期をはかったり、実行結果をまとめて出力するマシン。

の２つの機能を持ったマシンを使用する方法である（図2）。

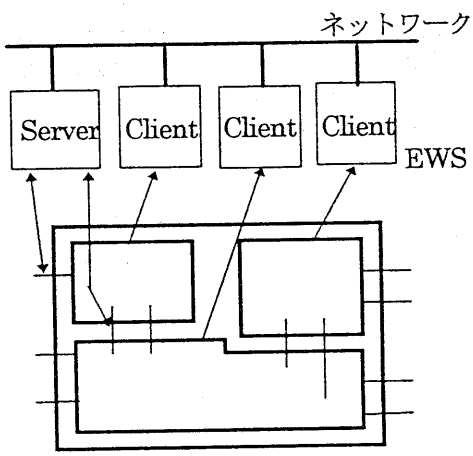

図2　EWSの分担

3.4 高速化

この方法では、Server-Client 間のデータ転送によりオーバーヘッド時間が発生する。そこで、データ転送回数を減らすために以下の工夫を行った。

- 各 Client の I/f ノード値にイベントが発生しない場合、各 Client が担当している部分回路でのイベント処理を行わずに時刻のみを進める。
- 全ノードのイベント情報がない場合、外部からの入力ノード値が変化して新たなイベントが発生するまで、Client 内の計算及び、Client-Server 間のデータ転送をやめ時刻のみを進める。

3.5 大規模対応

大規模回路対応のために、直接各 Client が自分の担当する部分回路を読み込むようにした。

これにより、各 Client のメモリを最大限利用する

ことができるため、より大規模な回路を実行することができる。

4　評価

このようにして作成した分割処理シミュレータの速度、精度、大規模回路対応について、評価する。始めに、本方式の最大のネックとなるServer-Client 間の通信による実行時間への影響を調べるため、テスト用として作成した回路を用い、その後、実製品に適応する。
実行時間は、以下の数式によって求める実行時間の比率を用いて評価する。

$$\text{実行時間の比率} = \frac{\text{分割処理シミュレータによる実行時間}}{\text{従来のシミュレータによる実行時間}} \times 100(\%)$$

4.1　テスト回路での評価

4.1.1　回路構成

今回作成した分割処理シミュレータの実行時間へは、Server-Client 間の通信時間が多大な影響を持つ。先に述べたように Server-Client 間の通信は、I/f ノードにイベントが発生すると行われる。そこで、回路内部で毎単位時間 2880 回イベントが発生するようにインバータを奇数個ループ状に接続した回路を作成した（図3）。

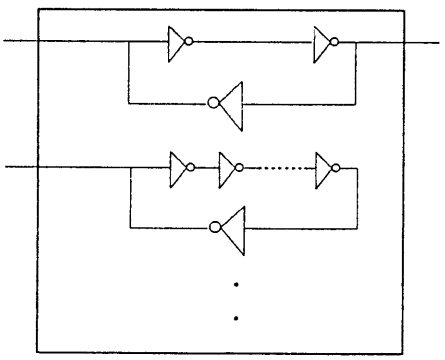

図3　テスト回路構成

この回路を部分回路に分割する際、工夫して I/f ノード全体の単位時間当たりの総イベント回数を変化させた（表2）。但し、この分割の際の各部分回路の大きさは等しくなるようにした。

この回路を、従来版のシミュレータと分割処理シミュレータで実行することにより、

- 精度。
- 実行時間と回路分割数の関係。
- 実行時間と I/f ノードのイベント数の関係。

を調べる。

表2　I/f ノードのイベント総数

分割数	2	3	6
理想分割	0	0	0
I/f ノード一定	72	72	72
I/f ノード変化	24	48	72

但し、"理想分割" は分割処理シミュレータの性能が最大に発揮される理想的な分割、"I/f ノード一定"は分割数によらず I/f ノード数を一定にした分割、"I/f ノード変化"は分割数に合わせて I/f ノード数を増やした分割である。

4.1.2　評価環境

この評価では、他のジョブの影響を受けないようにするために Sparc LT(メモリ 16M)等を使用して独立したネットワークを作成して行った。

4.1.3　評価結果

この回路での実行時間の比率を図4に示す。

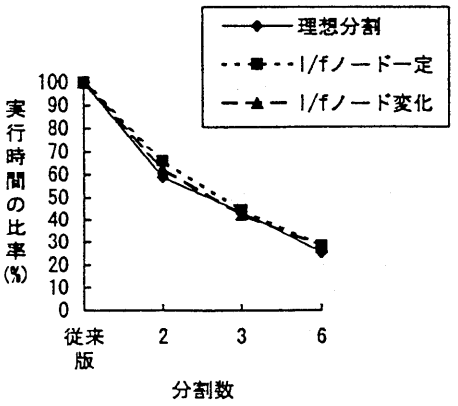

図4　分割処理シミュレータによる実行時間の比率

実行結果及び図４から以下のことが言える。

- 分割処理シミュレータの結果と従来版の出力結果が一致した。
- 理想分割をした場合、分割処理シミュレータの速度は、(1/分割数) という理想値をとり、分割数が多いほど実行時間が短くなる。
- 同じ分割数でも、I/f ノードのイベント発生数が少ないほど実行時間が短くなる。

4.2 実製品をモチーフにした評価

4．2．1 回路構成

前節の評価をふまえ、実製品に適用した時の、実行時間の評価を行う。モチーフとして用いた回路は弊社にて開発したＲ３９００[3]のコア部分であり、表３に示す６つの機能マクロからなる。

表３ Ｒ３９００構成素子

回路名	ノード数	N-channel 数	P-channel 数
全体	165999	285800	169118
部分回路１	209	304	304
部分回路２	21943	20761	20537
部分回路３	12728	20176	13162
部分回路４	35017	64343	36046
部分回路５	83759	160930	85779
部分回路６	12342	13286	13290

この回路に対して、

- 各分割回路規模をなるべく等しくする。
- I/f ノード数を最少にする。

ようにグルーピングを行いそれぞれの実行時間を測定した。グルーピング後の部分回路の大きさを表４～表９に示す。
(表中の回路名で使用する番号は表３と対応)

表４ ３分割 (回路規模優先) の部分回路規模

回路名	2,3,6	1,4	5
ノード数	46782	35288	83938
N-channel 数	60223	64647	260930
P-channel 数	46989	36350	85779

I/f ノード数＝３７８

表５ ４分割 (回路規模優先) の部分回路規模

回路名	4	5	1,2	3,4,6
ノード数	35144	83863	21922	25069
N-channel	64343	160930	27065	33462
P-channel	36046	85779	20841	26452

I/f ノード数＝７４１

表６ ５分割 (回路規模優先) の部分回路規模

回路名	2	3	4	5	1,6
ノード数	21944	12728	35017	83759	12550
N-channel	20761	20176	64343	160930	13590
P-channel	20537	13162	36046	85779	13594

I/f ノード数＝７５５

表７ ３分割 (I/f ノード数最少) の部分回路規模

回路名	4	1,3	2,5,6
ノード数	35017	13066	117788
N-channel 数	64343	20480	200977
P-channel 数	36046	13466	119606

I/f ノード数＝３３４

表８ ４分割 (I/f ノード数最少) の部分回路規模

回路名	4	5	1,3	2,6
ノード数	35144	83863	13065	33926
N-channel	64343	160930	20480	40047
P-channel	36046	85779	13466	33827

I/f ノード数＝５０９

表９ ５分割 (I/f ノード数最少) の部分回路規模

回路名	2	4	5	6	1,3
ノード数	21943	35017	83759	12342	12920
N-channel	20761	64343	160930	13286	20480
P-channel	20537	36046	85779	13290	13466

I/f ノード数＝７４４

4.2.2 評価環境

Ｒ３９００の回路規模が大きいため、使用マシンのメモリの関係で先の評価で使った独立ネットワーク上で動作させることができない。そこで既存のネットワークにつながっている AS4680(メモリ 416M)等を使用して評価する。

4.2.3　評価結果

分割後の回路規模がなるべく等しくなるように分割した回路を使用した場合と、分割後のI/fノード数が最少になるように分割した回路を使用したときの、分割処理シミュレータの実行時間の比率をグラフにしたものが図5、6である。

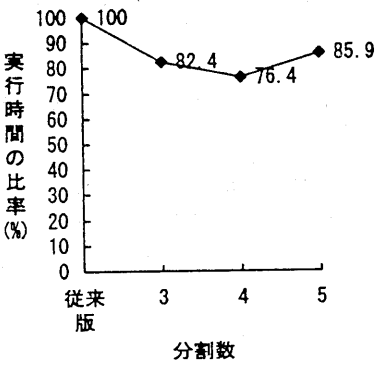

図5　分割処理シミュレータの
実行時間の比率(分割回路規模優先)

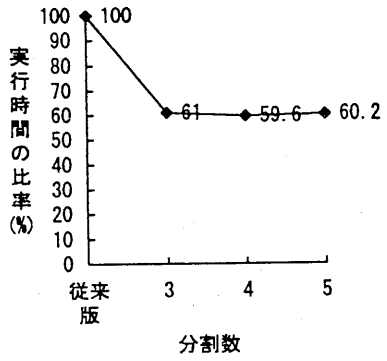

図6　分割処理シミュレータの
実行時間の比率(i/fノード最少)

このことから、以下のことがわかる。

- 分割処理シミュレータを使用することでI/fノードを最少にする手法で回路を3分割することにより、従来に比べ60%の実行時間ですむ。
- 分割後の回路規模を等しくするより、I/fノード数を少なくするように分割した方が高速になる。
- 分割数を増やすと実行時間は短くなるが、I/fノード数が増すため、ある点を境にオーバーヘッドの方が大きくなり、逆に時間がかかりすぎるようになる。

5　まとめ

既存のスイッチレベルのタイミングシミュレータの高速化手法として、ネットワークで結ばれた複数台のEWSを用いた分割処理方法を提案し、作成した。

そして、その性能を自作のテスト用回路と、実製品（R3900）を用いて評価した。

その結果、

- 分割処理化しても精度は従来のシミュレータと同じレベルを保てる。
- 理想的な分割で実行すると、小規模回路を対象にしても、実行時間を（1／回路分割数）に短縮できる。
- 既存のネットワークを使用して、実製品を評価した結果、ネットファイルへの微少な修正のみで、3分割してEWS4台で使用することにより、従来に比べ約60%の時間で実行できた。
- Client として使用するマシンの全メモリを利用できるため、大規模回路にも対応できるようになった。

以上より、本手法がシミュレータの高速化及び大規模回路対応に有効な手法であるといえる。

参考文献

[1]：C.J.Terman,"RSIM - A Logic-Level Timing Simulator", Proceedings of the IEEE International Conference on Computer Design, New York, pp437-440, Novenber 1983.

[2]：N.kimura,"Calculation of Total Dynamic Current of VLSI Using a Switch Level Timing Simulator(RSIM-FX)", IEEE 1991 Custom Integrated Circuits Conference, pp8.3.1-8.3.4

[3]：M.Nagamatsu, et al."A 150MIPS/W CMOS RISC Processor for PDA Applications", ISSCC95 Session DIGITAL DESIGN ELEMENTS, PAPER TA6.6,pp114-115