2A-5

並列計算機を用いた VLSI path delay 解析に関する基礎研究

竹田　隆太郎　鈴木　五郎

北九州市立大学大学院　国際環境工学研究科 情報工学専攻

1 はじめに

現在、VLSI の deep sub-micron 化により、配線遅延が大きくなっている。一方、Elmore ベースの STA[1]では実際の波形にくらべ、遅延が大きくなり timing error を多発させる一つの原因となっている。timing error を起した path のうち、STA によるレポートが目標値と大きく異なりかつ重要な path つまり、critical path に関しては、波形レベルの高精度な path delay 解析が必要となっている。それには Spice®を用いた解析が考えられるが、全ての path に関して解析を行うため、処理時間がかかってしまう。

我々は、Random Walk 法[2]を用いて、critical path に関してのみ波形解析を可能とする技術の開発を行っている。このうち Random Walk 法による波形解析の最初の処理である各 goal への visit 回数の booking[2]に関して、並列処理による高速化の検討を今回の目的とする。

2 Random Walk 法を用いた transient 解析

Fig.1 に示した interconnect に対応する Markov Cain Fig.2 を作成する。ここで、node 2 の電圧 $\mathbf{V}_2^0$ 及び、node 3 の電圧 $\mathbf{V}_3^0$ は各 capacitor の初期値を示し、E は driver gate の出力電圧を示す。$\mathbf{P}_1^2$ は node 2 から node 1 への遷移確率である。ここで、遷移確率は式(1)のように node 2 に接続されている、conductor の値 G, capacitor の値 C,および transient 解析を行う上での analysis interval Δt より計算を行う。

$$\mathbf{P}_1^2 = \frac{\mathbf{G}_1}{\mathbf{G}_1 + \mathbf{G}_2 + \frac{\mathbf{C}_2}{\Delta t}} \quad \cdots\cdots \quad (1)$$

以下 node 3 の電圧計算手順を示す。

(S1) node 3 において乱数を発生させ、枝の持つ遷移確率から遷移先 node を決定する。遷移先 node で再び乱数を発生させ、同様の操作を行うが、遷移先 node が goal の場合は遷移を止め、goal への visit 回数を加算する。以上の処理が game であるが、この game を十分な回数繰り返す。

(S2) 繰返し回数を K とし、node 3 に関する transient 解析 step 1 の電圧 $\mathbf{V}_3^1$ を式(2)を用いて計算する。

$$\mathbf{V}_3^1 = \frac{1}{\mathbf{K}}(\mathbf{N}_1^2\mathbf{E}^1 + \mathbf{N}_2^2\mathbf{V}_2^0 + \mathbf{N}_3^3\mathbf{V}_3^0) \quad \cdots\cdots \quad (2)$$

ここで例えば、$\mathbf{N}_1^2$ とは node 2 から goal node 1 への visit 回数を示している。

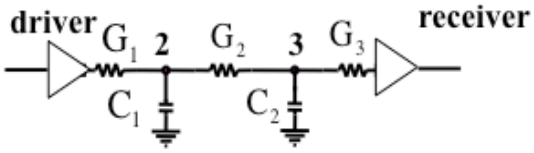

Fig.1 Interconnect circuit

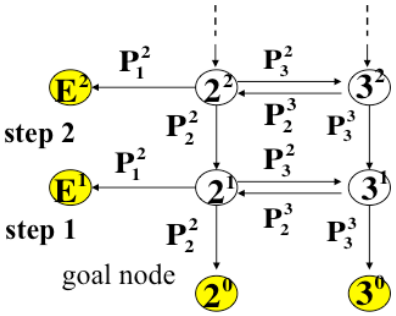

Fig.2 Markov Chain

(S3) transient 解析 step1 で求めた $\mathbf{V}_2^1$ および $\mathbf{V}_3^1$ を解析 step 2 での goal とし、式(3)を用いて解析 step 2 での該当 node の電圧 $\mathbf{V}_3^2$ を求める。

$$\mathbf{V}_3^2 = \frac{1}{\mathbf{K}}(\mathbf{N}_1^2\mathbf{E}^2 + \mathbf{N}_2^2\mathbf{V}_2^1 + \mathbf{N}_3^3\mathbf{V}_3^1) \quad \cdots\cdots \quad (3)$$

ここで、$\mathbf{N}_1^2$ $\mathbf{N}_2^2$ $\mathbf{N}_3^3$ は式(1)で用いた値を再利用している。

(S4) 解析 step 2 以降、同様の処理。

3 並列処理化

前章で説明したように、1 つの prelayout interconnect net において各 node に注目した場合、

(1) goal への visit 回数の booking

(2) transient 解析 step 1 以降の式への代入

が独立して行える。また

Interconnect Delay Analysis Using Parallel Computing
Ryutaro Takeda Goro Suzuki
University of Kitakyushu

(3) critical path 上にある prelayout interconnect net 毎の処理
も独立して行える(ただし、各 goal への visit 回数 booking のみ)。そこで、独立に計算ができる部分は Fig.3 のように並列処理化することにした。

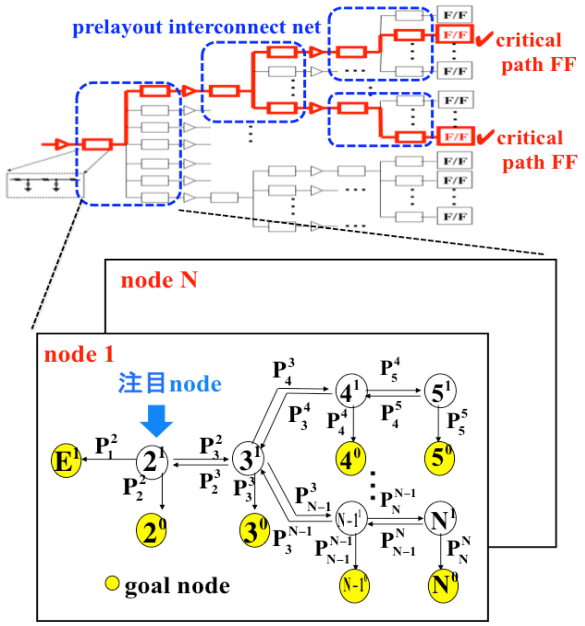

Fig.3 How to use parallel processing

この並列化は、NVIDIA 社 Kepler[3]を用いて実現した。Kepler は SMX, block, warp という 3 つの概念で並列性を制御しているが、実際にはユーザが与えた次の 2 つの情報によって並列性を決定している。2 つの情報とは、”block 数 n_b”および”block あたりに実行できる thread 数 n_s”である。n_b, n_s をどのように与えれば最適な並列性が実現できるか実験で調べた。結果、 n_b =4 および n_s =1,024 で最適な並列性つまり 4,096 並列が実現できる事が分った。実験結果を Fig.4 に示すが、これは n_b, n_s 2 つの要因に関するベクトル内積処理時間依存性を表している。

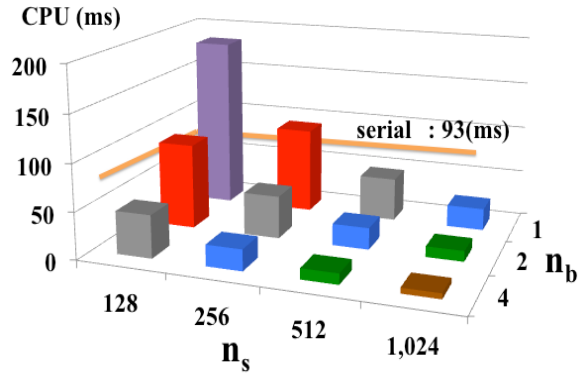

Fig.4 CPU by n_b and n_s

4 評価結果

前章での実験結果より、n_b=4, n_s=1,024 とし、Fig.3 に示す clock tree 回路を用いて、開発手法による評価を行った。goal への visit 回数を booking する処理つまり第 2 章(S1)で説明した処理に関して、serial に実行した場合と、parallel に実行した場合との処理速度の比較を行った。3 種類の evaluation circuit を用いた評価結果を Table.1 に示す。平均 7.3 倍の高速化である。

念のため、並列処理による goal visit 回数 booking 処理 degradation 有無の確認を行った。例えば evaluation circuit 1 では、Fig.5 のように対 Spice®誤差は平均 2(%),50%遅延時 1.3(%)であり、booking 処理は正しく行われている。

実験環境として、parallel には Kepler GeForce GTX650 1.1(GHz)を用いており、serial には Mac OS X Intel core i5 1.6(GHz)を使用した。

Table.1 Acceleration

	Parallel RW vs. Serial RW
Evaluation circuit 1	9 x
Evaluation circuit 2	6 x
Evaluation circuit 3	7 x

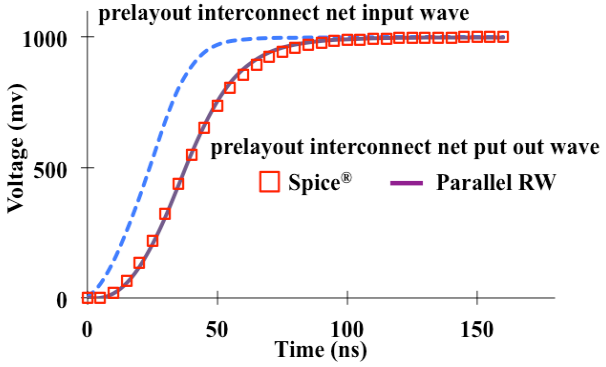

Fig.5 Prelayout interconnect net output wave

5 まとめ

Random Walk 法による波形解析の前準備[2]である、各 goal への visit 回数[2]の booking に関して、並列処理による高速化の検討を行った。並列処理を用いらなかった場合と比べ、約 7 倍の高速化を図る事ができた。

[1] 鈴木五郎,”システム LSI 設計入門”,コロナ社,2003.
[2] 竹田隆太郎,鈴木五郎,”QTAT Timing Analysis for ECO with Random Walk”,情報処理学会第 75 回全国大会,4A-6,2013
[3] Jason Sanders,Edward Kandrot,”CUDA By Example 汎用 GPU プログラミング入門”,インプレスジャパン,2011