Calculation for Strength of Signal Flow and Its Application in Circuit Simulation Algorithm[1]

信號流強度計算及其在線路模擬程式中的應用

Chun-Jung Chen
Department of Computer Science, Chinese Culture University
#55, Hwa-Gang Rd., Yang-Min Mountain, Taipei, Taiwan
email: chrischn@ms32.url.com.tw
陳俊榮
中國文化大學資訊科學系
臺北市陽明山華岡路 55 號
電子郵件: chrischn@ms32.url.com.tw

Abstract

In the circuit simulation subject, Relaxation-based algorithms have been proven to be faster and more flexible than the standard direct approach used in SPICE. Signal flow of the simulated circuit is very important in using Relaxation-based algorithm. However, there is no specific research undertaken for it. This paper gives a formal definition for the strength of signal flow (SSF), discusses how to calculate SSF, and devises techniques to utilize SSF in one of the Relaxation-base algorithms, ITA (Iterated Timing Analysis). Experimental results are given to prove the value of exploiting SSF in circuit simulation.

摘要

在線路模擬領域裏，基於鬆弛演算法已被證明較傳統的 SPICE 所使用的直接法要有效率，使用基於鬆弛法時，線路的信號流分析是很重要的，但是到目前爲止，尚未有專門針對此而做的研究。本論文對信號流強度做了數學上的詳細的定義，討論如何計算信號流強度，以及提出在基於鬆弛演算法中使用信號流強度的方法，所有提出的方法都加以實做，對於數位和類比線路的測試結果，證明了使用信號流強度於線路模擬器的價值。

Keywords: Strength of signal flow, circuit simulation, transient sensitivity, Relaxation-based

關鍵詞: 信號流強度, 線路模擬, 暫態敏感度, 基於鬆弛

I. Introduction

Circuit simulation [1] produces detailed timing waveforms of electronic circuits, which is crucial useful in circuit design process. Traditionally, people use the direct approach, which is used in SPICE, to solve circuit simulation problem. But direct approach has a big time-complexity, therefore it becomes very time-consuming in dealing with large-scale circuits. There are many methods invented to reduce the time complexity of circuit simulation, some of which improve numerical techniques [2, 3] and others trade off between accuracy and simplified computation models [4, 5]. In the former approach, various levels of numerical techniques have been improved thoroughly. To get further improvement in this approach, we have to introduce some new ideas. In this paper, we propose a new idea of using the “guidance information” to guide circuit simulation process. By using the guidance information, the circuit simulation problems have been solved with less

[1]This work was supported by National Science Concil of R.O.C under contract 90-2215-E-034-001

CPU time, while the same simulation accuracy is maintained. The guidance information used here is called the *strength of signal flow* (SSF).

Relaxation-based algorithms [1-3], which use the same numerical models for circuit devices and also generate accuracy transient waveforms of simulated circuits, have been proven to be more capable in simulating large-scale circuits than the direct approach. These algorithms partition the simulated circuit into subcircuits, simulate each subcircuit individually, and then combine their sub-solutions to form the entire solution. Relaxation-based algorithms use the same circuit formulations used by direct approach, and they can be as accuracy as the direct approach. In this paper, Iterated Timing Analysis (ITA), one of the Relaxation-based algorithms, is discussed. We choose ITA is because that ITA is the most successful Relaxation-based algorithm. It shows good performance, efficiency and robustness, in dealing with large-scale MOSFET circuits. In fact, we can find some commercial tools based on ITA in the circuit design community.

Signal Flow has been used for years [6]. Basically, a signal flow is just a flow of signal, similar as water current or electrical current. A signal flow coming from circuit variable x to y can be viewed as the influence on y caused by x. Signal flow has important impact in Relaxation-based circuit simulation algorithms. In these algorithms, signal flows are used to direct the circuit partitioning process (put strongly-coupled subcircuits together), and to decide subcircuit calculation orders (the scheduling of subcircuits), both which have crucial effects on simulation efficiency. However, most Relaxation-based algorithms only consider the "existence" of signal flow (such as that a Gate pin has signal flow to Drain pin of the same MOSFET, but not in the reverse direction). They don't consider the strength of signal flow. In this paper, we try to exploit SSF in ITA to get better simulation performance.

Following sections come in following orders. Section 2 describes the definition of SSF, and methods to calculate it. Section 3 illustrates how to use this information to enhance the simulation performance of ITA, which is followed by Section 4 describing experimental results of several MOSFET circuits, where some Relaxation-based algorithms are tested and compared. Finally, a conclusion remark is given.

II. Definition and Calculation for SSF

In this section we give SSF a formal definition, and describe the method to compute it. At the end of this section, the SSF waveform of an example circuit is shown for demonstration.

SSF is the degree of influence, caused by another circuit variable, on a circuit variable, and it is a time-varying value likes a timing waveform. So, it's appropriate to use the concept of sensitivity [6] to define it:

$$SSF(a,b) = \frac{\partial V_b}{\partial V_a} \tag{1}$$

SSF(a, b) is the strength of signal flow from node/branch a to node/branch b, where V_a and V_b are circuit variables associated with a and b respectively. We can call *SSF(a, b)* the SSF of a with respect to b. If a circuit has n circuit variables, SSF of the whole circuit can be represented as a $n \times n$ matrix. SSF is a time-varying value, and depends on input

signals. We want to see how it is calculated. The simulated circuit is described as follows:

$$F(Y(t),\dot{Y}(t),t)=0 \tag{2}$$

where Y is the vector of circuit variables, t is the time, F is a continuous function and "." means the differentiation with respect to time. Since Relaxation-based algorithms are used, (2) is partitioned into subcircuits, one of which, say a, is:

$$f(y(t),\dot{y}(t),w(t),\dot{w}(t),t)=0 \tag{3}$$

where y (a subvector of Y) is vector of circuit variables in a, w is the vector of circuit variables not in a, and f is a continuous function. We want to know all SSF of nodes in a with respect to an "input" node m in w. Equation (3) is rewritten as:

$$f(y(m,t),\dot{y}(m,t),\hat{w}(t),\dot{\hat{w}}(t),m,t)=0 \tag{4}$$

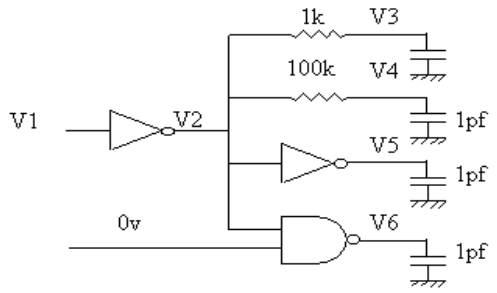
where $\hat{w}$ is equal to the resulted vector of removing m from w. Differentiating (4) with respect to m, we have:

$$f_y\frac{\partial y}{\partial m}+f_{\dot{y}}\frac{\partial \dot{y}}{\partial m}+f_m=0 \tag{5}$$

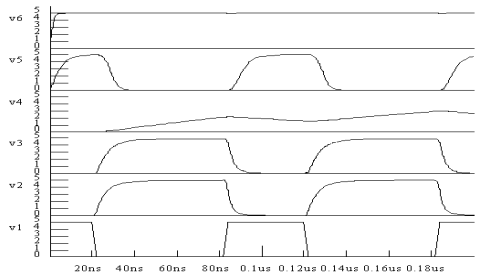
If we use Trapezoidal integration method to discretize (5), we have the following difference equation:

$$j_y s_{n+1}=(\frac{2}{h}f_{\dot{y}}s_n+f_y\dot{s}_n)-f_m=O_n-f_m=0 \tag{6}$$

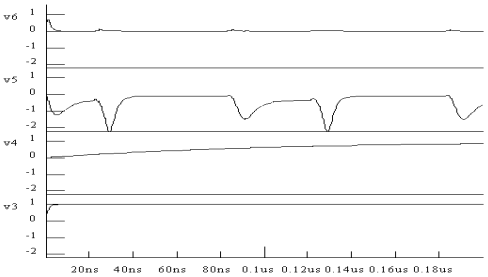
where $s_{n+1}=\frac{\partial y(t_{n+1})}{\partial m}$ is the SSF vector of all nodes in a with respect to m (t_{n+1} is the current time point), $j_y=(\frac{2}{h}f_{\dot{y}}+f_y)$ is the Jacobian of subcircuit a, O_n is the vector collecting values at previous time point t_n, and f_m is ith column of $j_w=\frac{\partial f}{\partial w}=(\frac{2}{h}f_{\dot{w}}+f_w)$ (assume that m is the ith element of w). Equation (6) is the calculating equation used in algorithm to calculate SSF, in which both j_y and f_m all depend on converged timing waveforms (in fact, j_y is also the Jacobian for timing calculation at the same time point). So, calculation for (6) has to wait for timing calculation of the same time point. We note that if m is the circuit variable of a node in one preceding subcircuit of a, say p, *s = SSF(m, y)* can be used to indicate whether a has been influenced by p via m.



(a)



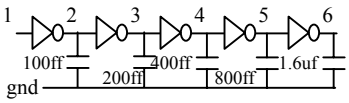
(b)



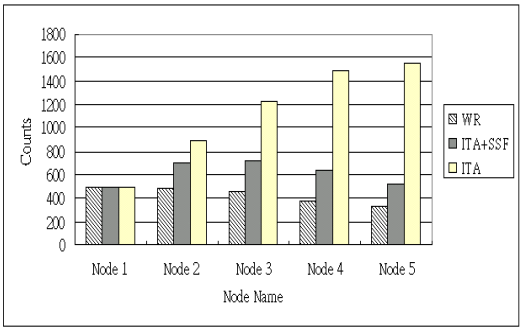
(c)

Fig. 1 (a) Schematic. (b) Timing waveforms. (c) SSF waveforms with respect to V2.

Fig. 1 demonstrates a circuit and SSF waveforms. Fig. 1(a) is the circuit schematic, Fig. 1(b) is the timing waveforms, and Fig. 1(c)



(a)



(b)

Fig. 2 (a) Schematic. (b) Counts of calculation of subcircuit.

contains waveforms of all SSF with respect to V2. We can find SSF of V3 and V4 with respect to V2 are 1 when capacitors have been fully charged. The SSF of V5 with respect to V2 depends on operation states of transistors, in which SSF exists only when MOSFETs are in active region. Due to the NAND gate's turning off function, SSF of V6 with respect to V2 is zero.

III. Exploiting SSF in ITA

In this section, we will show the method to utilize SSF to enhance the performance of ITA. Section 1 has mentioned that signal flow is used in partitioning and simulation processes of Relaxation-based algorithms. We explain how strength of signal flow affects the simulation process of ITA at first. ITA (GVT-ITA [3] here) is a robust, efficient and flexible algorithm for circuit simulation. It uses a selective-trace scheme [3] to dynamically trace subcircuits in the nonlinear equation solving process. This software scheme schedules succeeding subcircuits (which is defined by signal flow graph) of a subcircuit no matter how big the degree of influence is. So, ITA suffers from the lack of ability of using multirate behaviors and latency (ITA can utilize latency, but not well) [3] (Fig. 2 demonstrates this phenomena, in which the subcircuit calculation count of ITA increases as the position of subcircuit approaches the rear end.). ITA always calculates too many time points for subcircuits near primary outputs. Since this is the major drawback of ITA (called *over-scheduling* problem in this paper), we try to use SSF to solve it. Our idea is to utilize SSF in selective-trace scheme, and to make it more "intelligent". Let's investigate ITA algorithm at first. Following codes represent ITA algorithm.

Algorithm 1 (ITA Algorithm for Circuit Simulation):

```
/* Simulation duration is T_begin ~T_end */
// E() is an priority queue, whose elements are queues of subcircuits
Put subcircuits connected to primary input into E(T_begin);
while(E is not empty) {
    t_n+1 = the smallest event-time in E;
    for(k = 1; E(t_n+1) is not empty; k++) {
        // k is the relaxation index
        Clear TMP;     // TMP is a queue
        for(each subcircuit a in E(t_n+1)) {
            // E(t_n+1) is a queue
            Solve a at t_n+1 for transient responses;
            if(a has been converged) {      // converged
                Estimate next solving time t_next and
                    add a into E(t_next);
            }
            else {  // not converged
                Add a into TMP;
LB:             Add f_next_subcircuit(a) into E(t_n+1);
            }
        }
        E(t_n+1) = TMP;
    }
}
```

We can find that ITA is composed of three major loops, which are designated to treat (from outer to inner) time points, nonlinear relaxation iterations and subcircuits respectively. In the line labeled with LB, the function $f_{next\_subcircuit}(a)$ is the *activating function* of ITA. We find that $f_{next\_subcircuit}(a)$ maintains the selective-tracing [3] ability of ITA, and can be represented as follows:

$$f_{next\_subcircuit}(a) = \{\}, \text{if } a \text{ converges} \quad (7a)$$

$$f_{next\_subcircuit}(a) = f_{succeeding\_subcircuit}(a), \text{ otherwise} \quad (7b)$$

where $f_{succeeding\_subcircuit}(a)$ is the set of all fan-out

subcircuits of a (the fan-out relationships are determined in partitioning process that uses static signal flow). To solve the over-scheduling problem, this activating function is modified. Our idea is to supply Equation (7) additional information to help it to "bypass" unnecessary scheduling. Of course, the guidance information, SSF, is used to do such work. Consider that there exist p subcircuits s_i, $1 \le i \le p$, which are all fan-out subcircuits of subcircuit a. Assume that node n_j (with circuit variable $v(j)$) of a is the node affecting subcircuit s_j, N_j is the set of all nodes in s_j, and v_{ita_tol} is the convergence voltage tolerance for ITA's nonlinear relaxation to converge. We have the new activating function:

$$g_{next\_subcircuit}(a) = \{\}, \text{if } a \text{ converges} \quad (8a)$$

$$g_{next\_subcircuit}(a) = \{s_j \mid v_{chg}(j) > v_{\min}(j), 1 \le j \le p\}, \text{otherwise} \quad (8b)$$

Other equations follow:

$$v_{chg}(j) = \mid v(j)_{t_n} - v(j)^{k}_{t_{n+1}} \mid \quad (9)$$

where k is the index for ITA's nonlinear relaxation.

$$mssf(j) = \max\{ABS(SSF(n_j, \hat{n}_k)) \mid \hat{n}_k \in N_j\} \quad (10)$$

$$v_{\min}(j) = \frac{v_{ita\_tol}}{mssf(j)} \quad (11)$$

Here, $v_{chg}(j)$ is the variation of circuit variable $v(j)$ between last converged value at t_n and current value (at t_{n+1}, kth nonlinear relaxation iteration). Equation (10) shows that $mssf(j)$ is the maximum SSF of s_j with respect to n_j of a. In Equation (11), v_{ita_tol} is the minimum voltage change that ITA's nonlinear relaxation cares. Dividing v_{ita_tol} by $mssf(j)$ gives a threshold value $v_{min}(j)$ for $v(j)$. If the variation on $v(j)$ exceeds $v_{min}(j)$, a significant disturbance (larger then v_{ita_tol}) in s_j might be happen, then s_j can be scheduled. So, Equation (8) is derived. By the bypass ability of $g_{next\_subcircuit}( )$, many unnecessary calculations for succeeding subcircuits of a can be saved, which will be illustrated by examples in next section. Now we have ITA algorithm that adopts SSF:

Algorithm 2 (ITA Algorithm Adopting SSF):

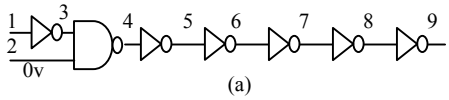
```
/* Simulation duration is T_begin ~T_end */
// E() is an priority queue, whose elements are queues of subcircuits
Put subcircuits connected to primary input into E(T_begin);
while(E is not empty) {
    t_{n+1} = the smallest event-time in E;
    for(k = 1; E(t_{n+1}) is not empty; k++) {
        // k is the relaxation index
        Clear TMP;    // TMP is a queue
        for(each subcircuit a in E(t_{n+1})) {
            Solve a at t_{n+1} for transient responses;
            if(a has been converged) {      // converged
                Estimate next solving time t_next and
                    add a into E(t_next);
                // if a has been converged, calculate its
                //   SSF with respect to all input variables
MK:             for(each input variables m of a) {
                    if(u is not in any subcircuit) continue;
                    Use (6) to calculate SSF of all nodes
                        with respect to m;
                    Store s_{n+1};
                }
            }
            else {  // not converged
                Add a into TMP;
LB:             Add g_next_subcircuit(a) into E(t_{n+1});
            }
        }
        E(t_{n+1}) = TMP;
    }
}
```

At the position of label MK is the loop to calculate SSF of all nodes in a with respect to each input node m (which comes from other subcircuits). This loop calculates all SSF of subcircuit a with respect to all input circuit variables inside other subcircuits, and store these SSF for the purpose of reference by $g_{next\_subcircuit}()$. The stored SSF information of a subcircuit x is not used unless x's immediate preceding subcircuit, says y, has called $g_{next\_subcircuit}(y)$, which is not definitely happen. However, due to the continuous-computing property of SSF (SSF calculation has to use "old" data, those data of previous time point), every time points should be calculated and stored. Finally, we note that in Algorithm 2, the activating function has been replaced by $g_{next\_subcircuit}(a)$.

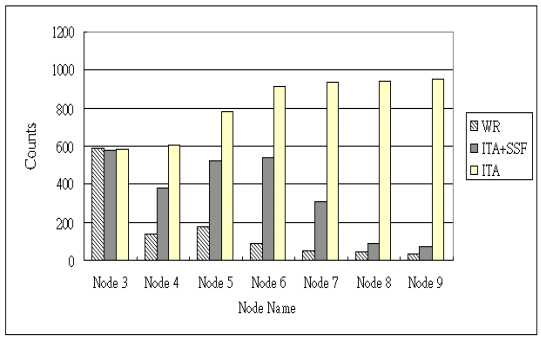
IV. Experimental Results

The proposed methods have been implemented in the circuit simulation program MOSTIME [6, 7], which contains several circuit simulation algorithms, including Waveform Relaxation (WR), ITA, Selective-tracing Waveform Relaxation (STWR) [7], and the classical direct method. So, we can compare simulation results of various algorithms.

At first, let's check the ability of the new activation function $g_{next\_subcircuit()}$ that exploiting SSF. The numbers of calculated time points of result waveforms are counted. Fig. 2 is an inverter-chain, where each node has a ground capacitor of different values. The values of these capacitors are assigned to create a multirate behavior for this circuit, in which the subcircuit closer to rear part has a lower transition rate. This arrangement causes the last inverter to be over-scheduled, since that each of its "faster" preceding subcircuits schedules it. Fig. 2(b) shows the counts of time points of timing waveforms of all nodes. Results of WR are used to represent essential time points



(a)



(b)

Fig. 3 (a) Schematic. (b) Counts of calculation of subcircuit.

TABLE I

SPECIFICATIONS OF SIMULATED CIRCUITS

Name	Node #	MOSFET #	Subcircuit #	Simulation Duration
InvChain	10	20	10	200ns
ALU	50	100	28	100ns
SynCounter	22	44	11	400ns
RippleCounter	19	38	9	400ns

TABLE II

USED CPU TIME AND COUNTS OF CALCULATED SUBCIRCUITS

Circuit	Used CPU Time		# Solved Subcircuits	
	ITA	ITA+SSF	ITA	ITA+SSF
6-stage InvChain	52.7	8.5	978,924	186,908
1-stage ALU	3.9	3.75	156,587	87,809
2-stage SynCounter	8.5	9.1	317,497	173,208
2-stage RippleCounter	5.2	7.2	181,571	148,363
4-stage ALU	18.2	10.9	379,058	138,676

CPU time is in Pentium III-550 seconds.

of each subcircuit, because that WR can exploit multirate behaviors of subcircuits and simulates each subcircuit at its own transition rate. It can be seen that ITA has caused more counts, which turns more serious at the node closer to the rear end.

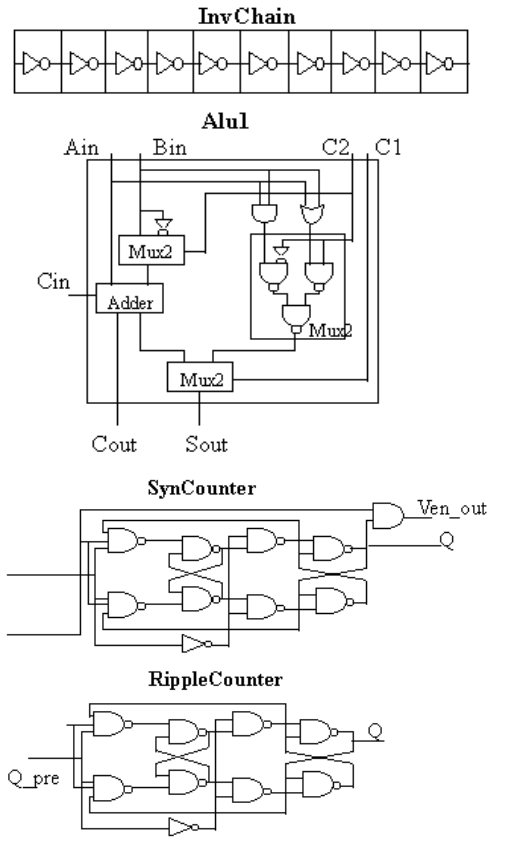
With the help of SSF, ITA reduces counts of time points dramatically. This example proves that SSF helps ITA solving multirate problems. Another example is shown in Fig. 3, in which the inverter chain is in latency due to the constant output of the NAND gate. Fig. 3(b) shows the counts of time points. We can find that SSF reduces counts again. Latency behaviors are not exploited well by ITA here. This is due to that ITA's activating function still activates succeeding latent subcircuits. By using SSF, this problem of ITA has been greatly alleviated.

Next, we check the performance of new methods in treating some larger MOSFET circuits. Table 1 lists the specification of all tested circuits (of one stage), and Table 2 lists

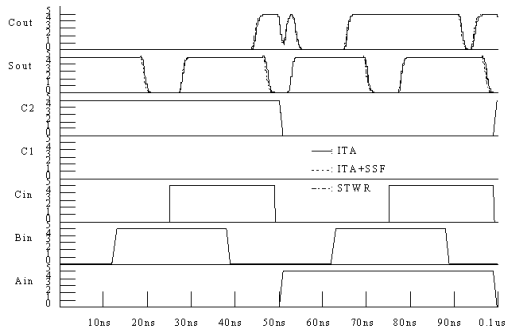
the used CPU time and counts of subcircuit calculation to compare the performance of original ITA and ITA adopting SSF (in which tested circuits are chained to form bigger circuit size). Schematics of the four tested circuits can be found in Fig. 4, where timing waveforms of ALU and SynCounter (both are one-stage circuit) are also presented. We can find that all waveforms match others well. Investigating Table 2, we find that SSF does certainly reduce the counts of subcircuit calculation. The reduction on counts of subcircuit calculation is especially clear in the "deep" (where depth of the signal flow graph [6] is big) circuit InvChain. This can be explained easily, for a deep circuit causes a more serious over-scheduling problem.

The middle three examples of Table 2 (example 2-4) shows that less counts of subcircuit calculation doesn't cause less used CPU time. We know that there exist overhead for computing SSF, the reduction on used CPU time is not so clear as that of counts of subcircuit calculation. However, in dealing with bigger circuit, exploiting SSF can always earn benefit. We can compare the last example with its smaller version, 1-stage ALU, to get this observation.

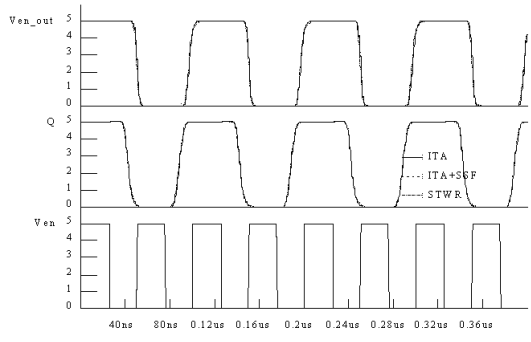
Now we investigate the time complexity of the new method. Circuits in Table 1 are cascaded to form test circuits of different size, which are simulated by ITA, ITA with SSF and STWR (which can exploit multirate of simulated circuits) respectively. The result used CPU time is accumulated in Fig. 5. We observe that the reduction on used CPU time turns more significant as the circuit size turns bigger, which justify that to exploit SSF in bigger circuit is more efficient. These test circuits' sizes are



(a)



(b)



(c)

Fig. 4 (a) Schematics of circuits. (b) Timing waveforms of the 1st bit of ALU. (c) Timing waveforms of the 1st bit of SynCounter.

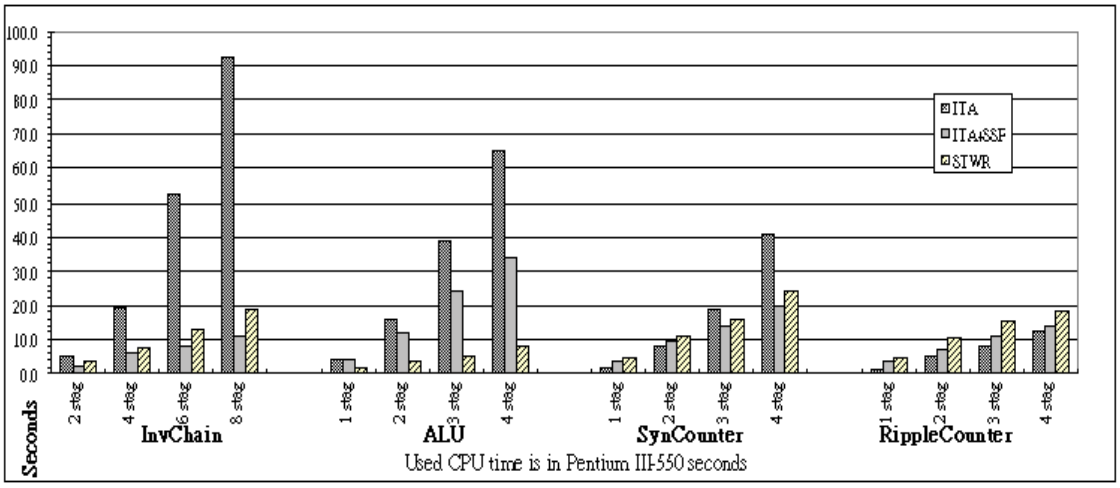


Fig. 5. Comparisons for used CPU time.

linearly increased, so the resulted CPU time can be used to see time complexities of algorithms. Now we observe the curve composed of used CPU time to check the time complexities of algorithms. It can be found that time complexities of both ITA adopting SSF and STWR are nearly linear in four example circuits, while that of ITA is not so good. Such a better time complexity is very valuable for large-scale circuit simulation algorithm. We find again that new method works better in the deep circuit InvChain. In the "shallow" circuit ALU it works well only when the scale of simulated circuit turns bigger. The later two example circuits are tightly-coupled circuits, in which the new method's effect is not so clear, for the new activation function can't encounter much unnecessary scheduling and then bypass them. In a synchronous counter, all subcircuits transit at the same time, so it becomes a "deeper" circuit than the ripple counter (in which every flip flop transits at different time). Since synchronous counter is deeper than ripple counter, SSF works well in dealing with synchronous counter. If too few subcircuit calculations have been saved, ITA adopting SSF might use more CPU time than ITA, due to the overhead for SSF calculation. Note also that ITA-based algorithms solve these feedback circuits better than STWR.

In Fig. 5, the time complexity of the proposed method is shown to be better than that of ITA. However, the performance of the new method is not good in all cases. We can consider two methods to enhance the performance. First one is to minimize the overhead of SSF calculation, such as to save SSF calculations on subcircuits near primary output. Second one is to consider other method to utilize SSF, such that simulated circuits other than the type of deep circuits can get the benefit of SSF. For example, we can merge two adjacent subcircuits as they are found to be strongly coupled.

V. Conclusion

The "modified" ITA proposed in this paper retains all advantages of the original ITA algorithm, while it further has more advantages, including the lower time complexity (more capable in simulating large circuits), better efficiency in dealing with deep circuits, and more capability to utilize multirate and latency behaviors of simulated circuits. The only drawback is the overhead for calculating SSF.

However, the bigger the simulated circuit is, this drawback becomes less significant. In other word, the proposed method has time complexity which is closer to linear than that of the original ITA. This paper demonstrates the method to use guidance information to guide the circuit simulation process. We can call this new idea “simulation with intelligence”. Experimental examples have shown the success of this new idea.

As the SSF is concerned, there exist more methods to enhance circuit simulation performance. For example, we can merge strongly coupled subcircuits (which will exhaust a lot of CPU time for Relaxation-based circuit simulation algorithms) which can be detected by inspecting SSF between them. More researches should be done to explore the usage of SSF in Relaxation-based circuit simulation.

Reference

[1] A. R. Newton and A. L. Sangiovanni-Vincentelli, "Relaxation-based electrical simulation, " IEEE Trans, Computer-aided Design, Vol. CAD-3, pp. 308-311, Oct. 1984.

[2] E. Lelarasmee, A. E. Ruehli, and A. L. Sangiovanni-Vincentelli, "The waveform relaxation method for time-domain analysis of large scale integrated circuits," IEEE Trans, Computer-aided Design, vol. CAD-1, pp. 131-145, Aug. 1982.

[3] R. A. Saleh and A. R. Newton, "The exploitation of latency and multirate behavior using nonlinear relaxation for circuit simulation," IEEE Trans. Computer-Aided Design, vol. 8, pp. 1286-1298, Dec. 1989.

[4] T. V. Nguyen, A. Devgan, O. J. Nastov, and D. W. Winston, "Transient sensitivity computation in controlled explicit piecewise linear simulation," IEEE Trans., Computer-aided Design, Vol. 19, NO. 1, pp. 98-110, Jan. 2000.

[5] P. Penfield and J. Rubinstein, "Signal delay in rc tree networks," Proceedings of the 2nd Caltech VLSI Conference, pp. 269-283, March 1981.

[6] C. J. Chen, W.S. Feng, "Relaxation-based transient sensitivity computations for MOSFET circuits," IEEE Trans. on CAD., Vol. 14, No. 2, pp. 173-185, Feb. 1995.

[7] C. J. Chen, and W.S. Feng, "Transient sensitivity computations of MOSFET circuits using Iterated Timing Analysis and Selective-tracing Waveform Relaxation," Proceeding of 31st Design Automation Conference, pp. 581-585, San Diego CA, June 1994.