附件一

行政院國家科學委員會補助專題研究計畫 ■ 成 果 報 告 □期中進度報告

（計畫名稱）

應用信號流強度於大型電路之線路模擬及暫態敏感度計算

計畫類別：■ 個別型計畫　　□ 整合型計畫
計畫編號：NSC 91－2215－E －034 －001 －
執行期間： 91 年 8 月 1 日至 92 年 10 月 31 日

計畫主持人：陳俊榮
共同主持人：
計畫參與人員：

成果報告類型(依經費核定清單規定繳交)：■精簡報告 □完整報告

本成果報告包括以下應繳交之附件：
■發表之論文兩份
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
□出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢
□涉及專利或其他智慧財產權，□一年□二年後可公開查詢

執行單位：中國文化大學資訊科學系

中 華 民 國 93 年 1 月 6 日

中英文摘要及關鍵辭

一、 計畫中文摘要

關鍵詞：信號流強度，暫態敏感度，線路模擬，基於鬆弛法

在前一年度的計劃中我們已經求得信號流強度的計算方法，而且也順利將之應用於ITA(Iterated Timing Analysis)中，顯著改進了大型電路的模擬效能，請參考已發表論文[1][2][3]。本計劃延續利用信號流強度製作智慧型演算法的基調，可以歸納出兩個重點，其一是應用信號流強度計算暫態敏感度，其二是將信號流強度應用在WR(Waveform Relaxation)演算法中。

信號流強度在線路模擬中是一種額外的資料，線路設計者並不需要它，其存在是為了幫助演算法加速模擬電路，計算信號流強度也要花費若干計算時間，所以應該盡量利用之。我們考慮線路暫態敏感度的計算，經過理論推導之後，發現可以利用信號流強度節省其計算時間。我們計劃在 ITA 演算法中製作這個方法，為了便於利用信號流強度，我們推導了可以同時計算電路時序解和暫態敏感度解的”同步版”ITA 演算法(Simultaneous ITA)，並加以實作，測試結果相當不錯。

我們之前使用的線路模擬演算法是ITA，由於其不能利用到線路的多頻特性(電路各部分的運作速度不同)，對計算效能的改善不能達到更高的境界，於是考慮可以應用多頻特性的 WR 演算法，但是 WR 有嚴重缺點，即模擬某些線路時會有收斂問題，我們發現可以利用信號流強度動態地偵測出會造成模擬問題的部分電路，然後使用如同 ITA 般較穩固的處理方式處理它們，將收斂問題徹底解決。

以上兩個目的都已經達成，而且發表了國際性的研討會論文[4][5][6]，本計畫的執行成果可說是非常成功的。

二、 計畫英文摘要

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

In the NSC project of last year, we have devised methods to calculate Strength of Signal Flow (SSF), and utilize it in ITA (Iterated Timing Analysis) algorithm to improve the performance of large-scale circuit simulation, which has achieved success and results in papers [1][2][3]. In this project, there are two points. First one is to utilize SSF to calculate transient sensitivities of the simulated circuits, and second one is to apply SSF in WR (Waveform Relaxation) algorithm.

SSF is additional information, which is not required by circuit designers. We calculate SSF in the purpose of helping circuit simulation. Since SSF exhausts CPU times, we have to utilize it as much as possible. After some mathematic study, we have found that SSF can assist the calculating of transient sensitivities. To use SSF in sensitivity calculation, we derive the new version of ITA algorithm, Simultanelusly-ITA, which calculates timing information as well as transient sensitivities simultaneously. The implementation have been made and tested, whose result is good.

ITA algorithm can’t utilize multi-rate properties of simulated circuits, which limits the degree of performance improvement. Therefore, we consider of using WR algorithm, which can utilize simulated circuits’ multi-rate behaviors, in this project. However, WR faces convergence problem in treating some kind of circuits. We found that we can use SSF to identify the portion of the simulated circuit that causes problem to WR, and then use more robust method (such as ITA’s method) to treat them to solve WR’s convergence problem.

Above two goals have all been achieved, and the results have been published in international conference papers [4][5][6]. The executing results of this project can be said to be very successful.

報告內容

一、前言

在前一年度的計劃中我們發展出信號流強度的計算方法，而且也順利將之應用於ITA(Iterated Timing Analysis)中，顯著改進了大型電路的模擬效能，請參考已發表論文[1][2][3]。本計劃延續利用信號流強度製作智慧型演算法的基調，針對兩個重點做研究，其一是應用信號流強度計算暫態敏感度，其二是將信號流強度應用在WR(Waveform Relaxation)演算法中。現在可以說是完全達成了研究目標。

二、研究目的

信號流強度在線路模擬中是一種額外的資料，線路設計者並不需要它，其存在是為了幫助演算法加速模擬電路，計算信號流強度也要花費若干計算時間，所以應該盡量利用之，於是我們考慮利用它於線路的暫態敏感度計算上。我們考慮線路暫態敏感度的計算，經過理論推導之後，發現可以利用信號流強度節省其計算時間。我們計劃在ITA演算法中製作這個方法，為了便於利用信號流強度，我們推導了可以同時計算電路時序解和暫態敏感度解的"同步版"ITA 演算法(Simultaneous ITA)，然後將信號流強度計算敏感度的公式製作於此版本中，加以測試，結果相當不錯[4]。

我們之前使用的線路模擬演算法是ITA，由於其不能利用到線路的多頻特性(電路各部分的運作速度不同)，對計算效能的改善不能達到更高的境界，於是考慮可以應用多頻特性的WR演算法，但是WR有嚴重缺點，即模擬某些線路時有收斂問題，我們發現可以利用信號流強度偵測出會造成模擬問題的部分電路，然後使用如同ITA般較穩固的處理方式處理它們，將收斂問題徹底解決。於是我們推導出ITA-WR[5][6]等演算法，加以實作，測試之，發現結果相當不錯。

三、文獻探討

本計畫針對兩個研究主題加以發揮，第一個是有關暫態敏感度的計算，第二個是有關於新的基於鬆弛演算法的研究。在第一個主題方面，有許多的先前的研究，可以用Adjoint 法計算，也可以用 Direct 法計算，也有討論在 Piece-wise Linear 模式下的計算法[13-17]；信號流強度的計算和暫態敏感度的計算類似但是未有討論共同計算兩者的，本計畫是這方面的最初研究。在第二個主題方面，使用基於鬆弛法求解線路模擬問題已經出現許多年，傳統的做法有 WR 和 ITA 的做法[10-12]，但是未有考慮將這兩種演算法結合求取其優點之綜合的，本計畫提出的 ITA-WR 演算法是這方面初始的研究。

四、研究方法

本計劃是屬於超大型積體電路計算機輔助設計的範疇，是使用軟體去解決硬體的問題，此類的計劃一般的研究方法是先觀察、探求問題的癥結並歸結出需求，然後在學理上尋求其解法，解決方法找出後，理論評估其可行性和效果，若一切沒問題，則開始設計其實際製作的細節，亦即設計其演算法，然後將之實際製作，程式完成後加以測試、驗正、修改演算法或程式，完成所需目標。將本計劃的研究方法條列如下：

1. 探求問題的癥結並歸結出需求：由於我們已有 SSF 計算功能，其花費了額外的計算時間，雖然可以因此加速模擬，但是是否可以多加利用 SSF 的價值? 所以要嘗試將 SSF 用在敏感度的計算上。另外，ITA 不能使用電路的多頻性，所以也需要研究如何利用 SSF 於其他的演算法(WR)，嘗試改善其功能。

2. 學理上尋求其解法：考慮利用信號流強度計算敏感度的方法，和 WR

演算法的改良方向(針對 WR 的缺點，以找出若干做法，如 global-time step, windowing)。

3. 理論評估其可行性和效果：製作初期的驗證用程式，實際模擬電路，發現所推導敏感度公式和新版 ITA 演算法的正確性；實驗數個 WR 較難處理的電路，從而確定 WR 改良方向是對的.。

4. 設計演算法：思考如何將理論上正確的做法實際以程式實做之，如何將之放入 MOSTIME 中。

5. 實際製作：Coding，寫程式。

6. 程式測試、驗正：配合 SPICE 和 MOSTIME 中的基於鬆弛演算法程式驗正所得結果的正確信，以及在效率上的增進。

本計劃將利用基於鬆弛法線路模擬器 MOSTIME[1-9]，在其上建構所有的程式；因為 MOSTIME 已有敏感度和信號流強度計算功能，也有 WR 演算法，本計劃所需之實做可以在此得到完全的配合,另外 MOSTIME 中也有其他基於鬆弛法類型的演算法如 ITA 和 STWR 等，可以將所計算出的結果和這些演算法做比較。

五、結果與討論

本計畫的結果可以論文[4]和[5]代表，請看到附件中的此兩篇論文。以下列出此兩篇論文的重點部份。

(一). 敏感度計算部分:

以下列出此部分的重要成果，第一個部分是使用 SSF 計算暫態敏感度的方法，第二個部分是同步 ITA 演算法，第三個部分是各個電路的模擬結果，包括時間和波形圖，有若干定義未描述清楚，請看到附件的論文本身。

1. SSF-based Sensitivity Computation

The calculating equation for strength of signal flow is as follows:

$$J_y s_{n+1} = -J_w p + (\frac{2}{h} f_{\dot{y}} s_n + f_{\dot{y}} \dot{s}_n + \frac{2}{h} f_{\dot{w}} p_n + f_{\dot{w}} \dot{p}_n) - f_d$$

$$= -J_w p + O_n - f_d \quad (1)$$

Where O_n is the variable collecting data at old time point t_n. We note that (1) is the *calculating equation* used in ITA algorithm for sensitivity computation. We can move J_y to right-hand side to get:

$$s_{n+1} = -J_y^{-1} J_w p + J_y^{-1}(O_n - f_d) \quad (2)$$

This equation can be processed by substituting equation (13) (please refer to the attached paper) to get:

$$s_{n+1} = -\alpha p + J_y^{-1}(O_n - f_d) \quad (3)$$

Where α is the matrix composed of all SSF of subcircuit *a*. So, in the case that SSF matrix α exist, we can use (3) rather than (1) to calculate transient sensitivities in ITA (i.e. to be the calculating equation). The obvious difference between (1) and (3) is that (1) uses J_w, which is an "expensive" quantity (especially when the models circuit elements are complex). We note that J_w is also used in solving (13) for SSF. If we use (1), we should recalculate J_w or store it when it has been derived in solving (13), both methods require extra computation overheads (CPU time or memory space). Moreover, calculating equations are solved several times for each time point, which makes the method of recalculating J_w more inefficient. Since SSF is available after solving (13), to use (3) as the calculating equation can save these overheads. We can prove the cost for solving (3) is just the same as that for (1). The right term, *A*, of right-hand side of (3) is obtained by solving the following linear system:

$$J_y A = (O_n - f_d) \quad (4)$$

So, operations of the process solving (3) include one linear system solving, one matrix multiplication (α times p), and one matrix addition, which are completely the same as that of (1). Therefore, we can conclude that (3) is more efficient than (1).

2. Simultaneously-Computing ITA

The algebraic SSFS is the same as algebraic SSF in Timing-ITA, so putting Sensitivity-ITA and Timing-ITA together can save the overhead for recalculating SSF (or the extra space to store SSF), and other quantities such as J_y. Algorithm 1 is the pseudo code for this combination:

Algorithm 1 (Simu-ITA)

/* Simulation duration is T_{begin} ~T_{end} ; E() is a priority queue, whose elements are ordinary queues */

{ Put subcircuits connected to primary input into $E(T_{begin})$, and set their states be *in_solving_timing*;

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
            if(state(a) is in_solving_timing) {
                Solve a at t_{n+1} for timing responses;
                if(a has converged on timing)
                    state(a) = in_solving_sensitivity;
            }
            else if(state(a) is in_solving_sensitivity) {
                Solve a at t_{n+1} for transient sensitivities;
                if(a has converged on sensitivity)
                    state(a) = all_converged;
            }
            if(state(a) is all_converged) {
                Estimate next solving time t_next ;
                Add a into E(t_next);
                state(a) = in_solving_timing;
            }
            else {   // not converged
                Add a into TMP;
                // Selective-tracing scheme
LB1:        if(state(a) is in_solving_timing) {
                    Add fan_out_ssf(a)    into E(t_{n+1}), and
                        set their state be in_solving_timing;
                }
    LB2:        else {
                    Add fan_out_ssfs(a)    into E(t_{n+1}), and
                        set their state be in_solving_sensitivity;
                }
            }
        }
        E(t_{n+1}) = TMP;
    }
}
```

3. Experimental results

Following tables, Table 1 and Table 2, show the performance of SSF-based sensitivity computation and that of Simu-ITA respectively. Figure 1 and Figure 2 shows schematics of tested circuits and waveform comparisons respectively. The SSF-based sensitivity computation is proven to be faster than the classical direct method. Simu-ITA is also proven to be better than Separated-ITA. We can check the waveform comparisons to see the correctness of the implemented algorithms.

Table 1: Performance of SSF-based sensitivity computation

Circuits	Calculation # of J_w		Used CPU Time	
	By (18)	By (20)	By (18)	By (20)
10-stage InvChain	69,123	23,041	3.20	2.50
50-stage InvChain	783,786	261,262	37.80	34.60
4-stage Shift Register	73,437	27,710	5.50	4.80
4-stage SynCounter	417,449	130,994	34.60	31.90

Used CPU time is in Pentium III-550 seconds.

Table 2: Performance of Simu-ITA

Circuits	Separated-ITA	Simu-ITA
10-stage InvChain	3.4	2.5
50-stage InvChain	51.5	34.6
4-stage Shift Register	6.1	4.8
4-stage SynCounter	35.4	31.9

Used CPU time is in Pentium III-550 seconds.

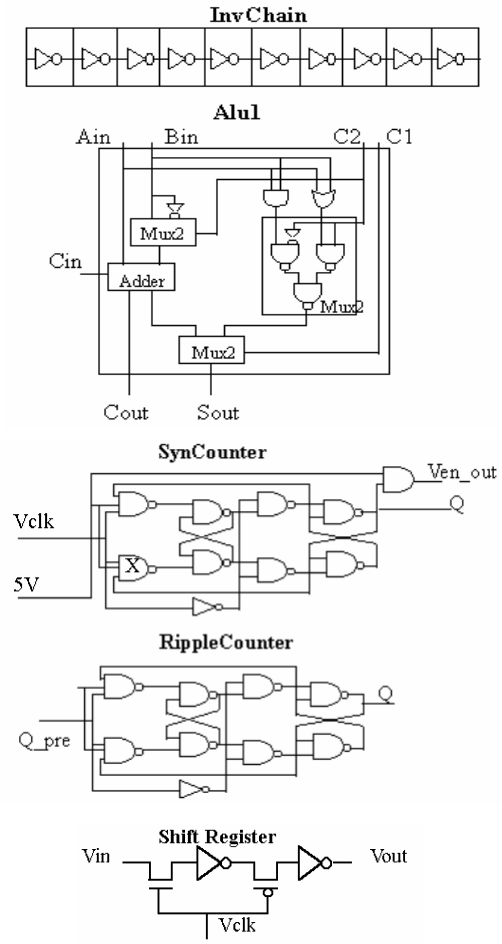
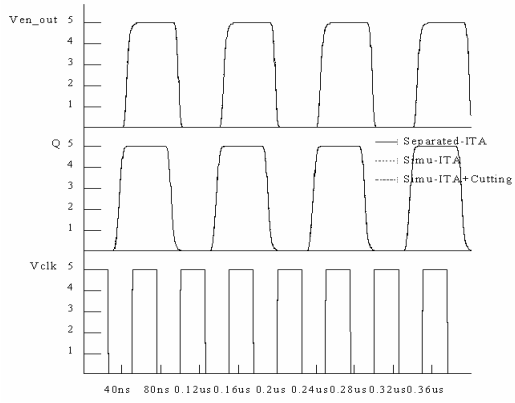
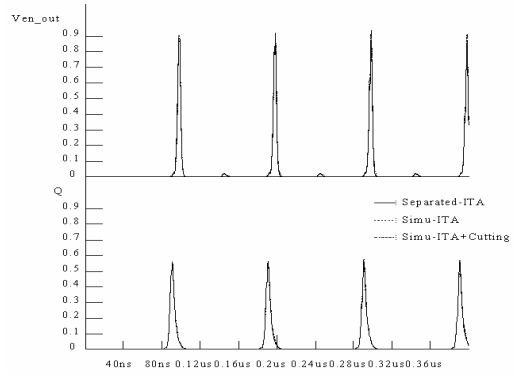


Figure1: Schematics of tested circuits.



(a)



(b)

Figure 2: The results of SynCounter. (a) Timing waveforms obtained by using Separated-ITA, Simu-ITA, and Simu-ITA with Cutting-scheme. (b) Sensitivity waveforms of three algorithms, where the design parameter is the width of a MOSFET inside X of SynCounter in Figure 1.

(二). ITA-WR 演算法部分:

此演算法的做法是將電路分成了兩個部分，即回授電路部份和無回授電路部分，然後兩個部分使用適用的演算法模擬之，所以演算法本身包括了兩個主要的部分，第一個是利用信號流強度分辨電路的部分，第二個是模擬演算法本身。

1. Identifying Feedback Portions of Simulated Circuit

The first step of ITA-WR algorithm is to identify feedback portions of the simulated circuits by using SSF (strength of signal flow). Here are the algorithms to do this job.

Algorithm 2 (Identifying Feedback Loop):

Input: Directed graph *DG* representing signal flow graph of the simulated circuit.
Output: Blocks, one of which represents a SCC (Strongly Connected Component).

```
void IdentifyFeedback(DG)
{
    Use depth-first-search to label all vertices with their visiting order;
    Clear all vertices' mark;
    for(all vertices sub in DG) {
        for(all sub's incident edges ed, <pre, sub>) {
            if(ed is a back edge) {
                /* according to ed, finding all corresponding loops */
                Put sub into an empty block;
                DfsFindingLoop(sub, pre);
            }
        }
    }
}

bool DfsFindingLoop(sub, pre)
{
    mark[sub] = 1;
    if(sub == pre) found = 1;
    else {
        found = 0;
        for(all fan-out edges ed of sub, ed = <sub, nxt>) {
            if(! mark[nxt]) tmp = DfsFindingLoop(nxt, pre);
            else tmp = 0;
            if(tmp) {
                Add nxt into the same block of sub;
                found = 1;
            }
        }
    }
    return(found);
}
```

2. The ITA-WR Algorithm

Here, the ITA part doesn't use selective-tracing, so it becomes Nonlinear Relaxation (NR) algorithm. And the used WR algorithm is the more advanced version called STWR[9], so the algorithm is called NR-STWR. In fact, it is one version of ITA-WR algorithms. There are some software scheme inside NR-STWR to make it as good as ITA-WR algorithm.

Algorithm 3 (NR-STWR Circuit Simulation Algorithm):

Input: The simulated circuit been partitioned into subcircuits, input waveforms, and simulation time duration [t_0, t_{begin}].
Output: Time waveforms of all circuit nodes.

```
NR_STWR()
{
    Reset all subcircuits' t_c to T_begin;
    for(k = 1; T_convergence != T_end ; k++) {     // k is the relaxation index
        for(all subcircuit x whose t_c != T_end ) {
            x.t_{n+1} = x.t_c;
            Put x into PQ;
        }  // PQ is a priority queue, whose elements are subcircuits
        while(PQ is not empty) {
            Delete the subcircuit a having minimum t_{n+1} from PQ;
            Put a into q;
            in_ITC = 0;
L1:         while(q is not empty) { // iteration for NR-queue
                Delete a from q; // q is a normal first-in-first-out queue
L2:             if(! in_ITC && a is in a block) {
                    Put all subcircuits of the same block into q;
                    t_min = MIN{b.t_{n+1} | b ∈ q};
                    for(b ∈ q) b.t_{n+1} = t_min;
                    in_ITC = 1;
                }
                Solve a at a.t_{n+1} for transient responses;
L6:             // spike-checking scheme is added here
                if(fail for solving or answer is unacceptable) {
                    // Need roll back
                    Shrink the time step of a, reset a.t_{n+1};
L3:                 if(in_ITC) Discard all contents of q;
                    goto L5;
                }
                if(in_ITC && a's solution is not converged) {
L4:                 Add a into q;
                    Save time waveforms into tables;
                    continue; // process next subcircuit in q
                }
                if(a's last time point is converged) a.t_c = t_{n+1};
                Store time waveforms into tables;
                Estimate a's next time point and store it into a.t_{n+1};
                if(exact(a)) add a into PQ;
                for(all fan-out subcircuits w of a) {
                    if(w is not in PQ and exact(w)) add w into PQ;
                }
                if(k > 1 && a has been not converged for MUCN time points)
                {   // MUCN controls window size of each relaxation
                    T_convergence = the smallest t_c of all subcircuits;
                    goto L5;    // end a relaxation
                }
            } // loop end of while(q is not empty)
        } // loop end of while(PQ is not empty)
L5:
    }
}
```

3. The Running Results

Here we check the performance and waveform comparisons for algorithms.

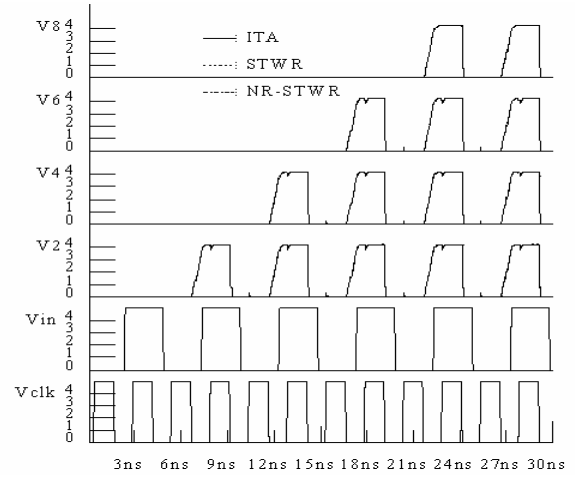
Table 3
CIRCUIT SPECIFICATIONS

Number	Circuit	Node#	MOSFET#	Subckt#	Block#
1	Inv10	10	20	10	0
2	Inv100	100	200	100	0
3	Alu1	50	100	28	0
4	Alu4	200	400	112	0
5	Pm	400	800	112	0
6	Sreg1	4	6	2	1
7	Sreg8	32	48	16	1
8	Ceamp2	4	2*	2	1
9	Ceamp10	20	10*	10	1
10	Ring3	3	6	3	1
11	Ring11	11	22	11	1
12	Sct1	22	44	11	1
13	Sct4	88	176	44	4

*: Number of bipolar transistors.

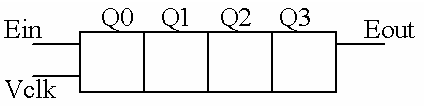
At first, we check the waveform comparisons explained by Figure 3 and Figure 4. The discussed circuits are all feedback circuits, but the waveform match each other well, which justifies the correctness of the implementation. Then we check the running results listed in Table 4 (where Table 3 shows the circuit specifications). We can find that WR-ITA (NR-STWR) performs well in two kind of circuits (feedback circuits and one-way circuits).

(a)

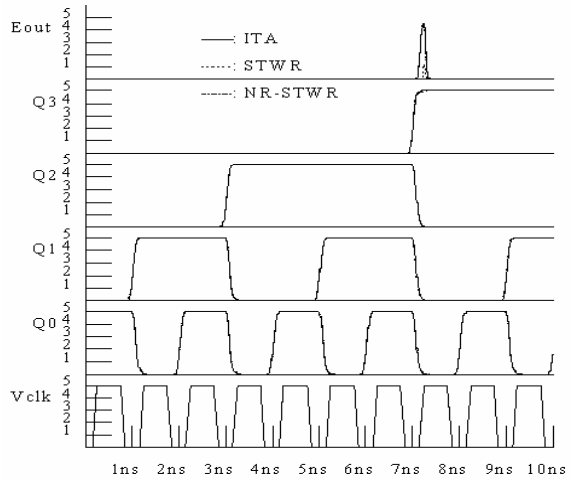


(b)

Fig. 3 (a) Schematic of Sreg8, in which every tiles are all Sreg1. (b) Its time waveforms calculated by algorithms.

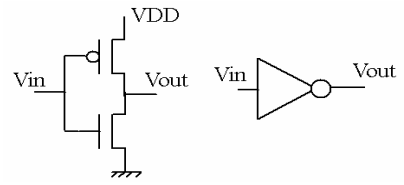


(a)

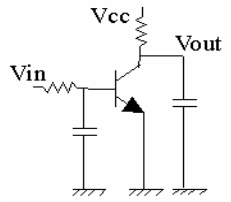


(b)

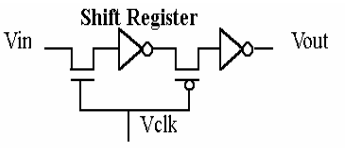
Fig. 4 (a) Schematic of Sct4, in which every tiles are all Sct1. (b) Its time waveforms calculated by algorithms.



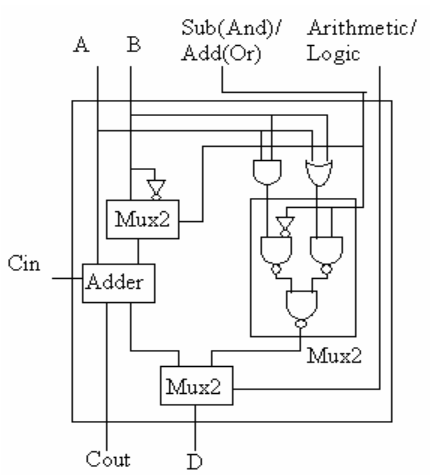
(a)



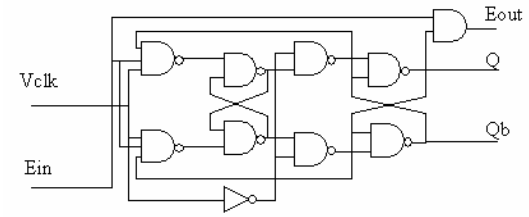
(b)



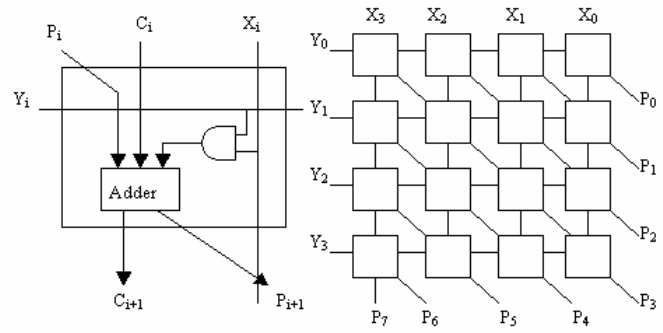
(c)



(d)



(e)



(f)

Fig. 5 Schematics of tested circuits. (a) Inv1, CMOS inverter. (b) Ceamp1, common-emitter amplifier. (c) Sreg1, shift register. (d) Alu1. (e) Sct1, synchronized counter. (f) Pm, parallel multiplier.

TABLE 4
RUNNING RESULTS OF ALGORITHMS

Circuit	Subcircuit Calculated #			CPU Time			Relaxation Count	
	ITA	STWR	NR-STWR	ITA	STWR	NR-STWR	STWR	NR-STWR
Inv10	19,637	4,384	4,684	0.4	0.4	0.4	2	2
Inv100	2,471K	0.245K	0.188K	19	4.3	3.2	17	2
Alu1	165,781	63,480	43,196	3.3	2.1	2.2	21	2
Alu4	745,747	244,236	144,919	12	9.9	6.8	10	2
Pm	221,147	40,534	41,545	9.9	3.3	3.8	6	2
Sreg1	11,112	14,542	15,696	0.3	1.3	0.4	172	2
Sreg8	79,165	414,490	79,262	1.2	22	3.1	69	2
Qinv2	2,798	1,130	2,430	0.4	0.3	0.3	2	2
Qinv10	45,307	52,067	41,382	1.2	3.1	1.1	35	2
Ring3	1,227	1,847	1,597	0.3	0.4	0.3	24	2
Ring11	66,917	44,535	83,543	1.1	1.2	1.5	117	2
Sct1	50,296	36,639	31,522	1.2	2.3	1.1	57	2
Sct4	318,558	168,139	107,506	5.3	7.4	2.9	55	3

CPU time is in Pentium IV-1.4G seconds.

討論

本計畫的初始規劃可以說大部分都達成了，其結果也都相當不錯，這是一個成功執行的計畫。現在看到若干可以討論的部分，首先，在 WR 演算法方面，有關於使用信號流強度調整動態 WR 的 Windows 的規劃，實際製作程式後，我們發現，大部分的線路都能在兩次的 WR iterations 裡結束掉，這是因為所有回授都被去除掉了，所以，動態調整視窗大小的功能就不需要了，因此這部分就加以省略。另外在未來進步空間的方面，計畫中提出的 ITA-WR 演算法使用靜態的信號流強度辨別回授電路，使得效果不是最好的，因為電路動態進行時，信號流強度也會隨時改變，其回授型態也是隨時更改的，所以考慮如何應用動態的信號流分析是未來的重要課題。本計畫也推導出了快速的敏感度計算方法，未來也應該運用此資訊，比如說最佳化，電晶體尺寸調整等。

六、參考文獻

[1] Chun-Jung Chen, “Calculation for strength of signal flow and its application in circuit simulation algorithm,” National Computer Symposium (全國計算機會議), Taipei, Taiwan, December 2001. NSC-90-2215-E-034-001.

[2] Chun-Jung Chen, and Wen-Pin Tai, “Signal flow analysis and its utilization in Relaxation-based circuit simulation,” The International Conference on Fundamentals of Electronics, Communications and Computer Sciences, Tokyo Japan, March 2002. NSC-90-2215-E-034-001.

[3] Chun-Jung Chen, Wen-Pin Tai, and Jenn-Dong Sun, “Strength of signal flow in circuit simulation and its application,” Symposium on Design, Test, Integration, and Packaging of MEMS/MOEMS, Canne France, May 2002, NSC-90-2215-E-034-001.

[4] Chun-Jung Chen, Wen-Pin Tai, and Jenn-Dong Sun “Signal flow analysis and its utilization in Relaxation-based transient sensitivity simulations,” International Symposium on Intelligent Signal Processing and Communication Systems, GaoShiung, Taiwan, November, 2002. NSC-91-2215-E-034-001.

[5] Chun-Jung Chen, Weishing Liu, and I-Shu Chung, “Large-scale circuit simulation by using selective-tracing Waveform Relaxation and Nonlinear Relaxation,” The 1st International Workshop on Compact Modeling, Yokohama, Japan, January 2004, NSC-91-2215-E-034-001.

[6] Chun-Jung Chen, Wen-Pin Tai, Hwe-Hong Shu, Jenn-Dong Sun, and Weishing Liu, “Large-scale Circuit Simulation by Using Composition of Selective-tracing Waveform Relaxation and Iterated Timing Analysis,” The 46th IEEE Midwest Symposium on Circuit and System, Cairo, Egypt, December, 2003, NSC-91-2215-E-034-001.

[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.

[8] C. J. Chen, W.S. Feng, “Relaxation-based transient sensitivity computations for MOSFET circuits,” IEEE Trans. on Computer-aided Design, Vol. 14, No. 2, pp. 173-185, Feb. 1995.

[9] C. J. Chen, “Relaxation-based circuit simulation for large-scale circuits with lossy transmission lines,” Proceeding of 2nd Modeling and Simulation for Microsystems Conference, pp. 258-261, San Juan Puerto Rico, April 1999.

[10] 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.

[11] 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, December 1989.

[12] 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.

[13] 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.

[14] Andrew R. Conn, Paula K. Coulman, Rund A. Haring, Gregory L. Morrill, Chandu Visweswariah, and Chai Wah Wu, "JiffyTune : Circuit

Optimization using time-domain sensitivities", IEEE Tran. on Computer-aided Design, Vol. 17, No. 12, pp. 1292-1309, Dec. 1998.

[15] R. Kay, and L. T. Pileggi, "EWA：Efficient wiring-sizing algorithm for signal nets and clock nets," IEEE Trans., Computer-aided Design, VOL. 17, NO. 1, pp. 40-49, January, 1998.

[16] S. Contini, S. Scheer, and M. Wilikens, "Sensitivity analysis for system design improvement," Proceeding of the International Conference on Dependable Systems and Networks, DSN 2000, 2000.

[17] S. W. Director and R. A. Rohrer, “The generalized adjoint network and network sensitivities,” *IEEE Trans. Circuit Theory*, vol. CT-16, pp.318-323, Aug. 1969.