行政院國家科學委員會專題研究計畫 成果報告

應用電路型態辨認於選擇走訪波形鬆弛演算法，漸增電路模擬，和對於多設計參數之暫態敏感度的同步計算

研究成果報告(精簡版)

計 畫 類 別	：個別型
計 畫 編 號	：NSC 94-2215-E-034-001-
執 行 期 間	：94 年 08 月 01 日至 95 年 10 月 31 日
執 行 單 位	：中國文化大學資訊科學系

計畫主持人：陳俊榮

計畫參與人員：工讀生：梁家豪、李昇鴻、陳淑斐、林宛頻、陳福維

報 告 附 件 ：出席國際會議研究心得報告及發表論文

處 理 方 式 ：本計畫涉及專利或其他智慧財產權，1 年後可公開查詢

中 華 民 國 96 年 01 月 26 日

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

應用電路型態辨認於選擇走訪波形鬆弛演算法，漸增電路模擬，和對於多設計參數之暫態敏感度的同步計算

計畫類別：■ 個別型計畫　　□ 整合型計畫
計畫編號：NSC 94－2215－E －034－001－
執行期間：　94 年 8　月 1　日至 95　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：
□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：中國文化大學資科系

中　華　民　國　96　年　1　月　28　日

應用電路型態辨認於選擇走訪波形鬆弛演算法，漸增電路模擬，和對於多設計參數之暫態敏感度的同步計算

Applying circuit type identification on STWR, Incremental circuit simulation, and simultaneous computations for transient sensitivities with respect to multiple design parameters

計畫編號：NSC94-2215-E-034-001

執行期間：94 年 8 月 1 日 至 95 年 10 月 31 日

主持人：陳俊榮 中國文化大學資訊科學系副教授

<u>中英文摘要</u>

中文摘要

關鍵字：線路模擬，漸增模擬，信號流，暫態敏感度

本計畫進行電腦自動化設計領域線路模擬方面的研究，包含三個主題，分別為使用靜態電路型態辨認於選擇走訪波形鬆弛演算法(STWR Algorithm)，時序及敏感度的漸增模擬和對於多設計參數之敏感度的同步計算方法。

我們之前的國科會計畫研究成果指出了應用靜態電路型態辨認於基於鬆弛線路模擬可以大幅加強模擬效能，其中線路分成了有迴授部分和無迴授部份兩種型態。在本計畫的第一個主題中，我們提出將線路型態進一步分成三種的做法，那就是無迴授部分，相鄰交聯部份和迴圈交聯部份，然後搭配 STWR 進行模擬，由於 STWR 的高效率和較詳細的線路型態分辨，模擬效能進一步提高了。

漸增線路模擬是很實用的模擬技巧，使用前一次模擬所得的波形，加速後續的模擬，使用於電路設計過程中可以大幅減少線路驗證的時間。過去已有多種技術，但在最好的 Incremental-in-Time 技術中仍有不少多餘的子線路計算發生；我們提出可以精確追蹤子線路變化度，只模擬真正有波形變化的子線路的做法，此做法配合 STWR，成為 Incremental-in-Change 法，程式測試結果驗證了其優異的效能。另外，於自動化電路設計方面敏感度漸增模擬也是很實用且被需求的技術，但未有人發表此方面的研究，我們在此主題中一併提出其做法。

電路的敏感度資訊矩陣是由電路變數對所有設計參數的微分得到的，其中橫列是某電路變數對所有設計參數的敏感度，而縱行是所有電路變數對某設計參數的敏感度，使用時經常是以橫列為單位被要求的，亦即需要的是某電路變數對所有設計參數的敏感度；傳統的敏感度計算法有 Direct 法和 Adjoint 法，Direct 法較好計算，但不幸的是其使用 column-major 的順序計算敏感度，使計算效率打了折扣，而 Adjoint 法是利用 row-major 方式計算，但是其計算方式較複雜。我們提出了使用基於鬆弛法計算敏感度的新做法，其中子線路計算採 Direct 法且使用 row-major 計算順序，新方法在設計參數個數較多時確實可大幅的效率改進。

英文摘要

Keyword: Circuit simulation, incremental simulation, signal flow, transient sensitivity

There are three subjects in this report, which are using static circuit type identifications in

STWR algorithm, timing and sensitivity incremental simulation, and simultaneous computation for sensitivities with respect to multiple circuit design parameters.

To use circuit type identifications in relaxation-based circuit simulation can dramatically enhance the simulation efficiency. In our previous work, circuits are classified into two types, which are feedback type and non-feedback type. In the first subject of this proposal, we introduce the method to divide the simulated circuit into three portions, including non-feedback portion, adjacent coupling portion and directed loop portion. We apply these circuit identifications in STWR. Due to the high efficiency of STWR and the detailed circuit type identifications, the simulation efficiency has been further enhanced.

Incremental simulation is a practical technique in circuit design process, which can dramatically reduce circuit simulation time. There are many techniques proposed, but the best of which, the Incremental-in-Time technique, still shows excessive redundant subcircuit calculations. We propose a new method based on STWR, which can precisely trace the waveform variations and simulate those subcircuits that really change. We call this method the Incremental-in-Change strategy. The experimental results support this characteristic we claimed. By the way, the sensitivity incremental simulation is also needed in design automation process, however there is no related research found. We also constructed sensitivity incremental simulation in this subject, too.

Differentiating circuit variables with respect to all design parameters gives the sensitivity matrix. In this matrix, one row contains sensitivities of a circuit variable with respect to all design parameters, and one column contains sensitivities of all circuit variables with respect to one design parameter. In practical usage, one row of this matrix is usually requested at a time. Classical methods to simulate transient sensitivity are Direct method and Adjoint method. Direct method is easier to calculate, but, unfortunately, it calculates sensitivity in column-major order, which seriously worsen the computation efficiency. We propose a new method for relaxation-based sensitivity computation, in which a sensitivity subcircuit is calculated by Direct method and in row-major order. Theoretical discussion explains that the new method has much better performance in dealing with bigger number of design parameters.

<u>報告內容</u>

一、研究目的與文獻探討

本計畫有三個主題，第一個主題研究將選擇走訪波形鬆弛線路模擬演算法 STWR (Selective-tracing Waveform Relaxation) [1]和靜態線路型態辨認結合，第二個主題研究漸增模擬(incremental simulation)，第三個主題則研究單線路變數對於多設計參數的敏感度(sensitivity of a circuit variable with respect to many design parameters) 之同步計算方法。

在第一個主題裡，我們的目的是獲取更有效率且穩固的大型電路模擬演算法，這是自之前的研究成果依序發展而得的。在 92 年的國科會計畫中，我們的研究成果之一是結合靜態線路型態辨認於基於鬆弛線路模擬演算法[1][2][3]，其做法基本上是先辨認被模擬電路裡所有子線路的型態，將之分成具迴授效果的 feedback portion 和不具迴授效果的 one-way portion，然後使用不同的模擬方法處理之，feedback portion 使用較穩固(robust，亦即無收斂問題)但效率較差的 ITA(Iterated Timing Analysis)[4](或稱作 Nonlinear Relaxation 處理)，而 one-way portion 則使用效率較佳(但對迴授電路有收斂問題)的 WR(Waveform Relaxation)[4]

或是 STWR 處理，實做上是以 WR 作為主程式，feedback portion 則視做 WR 之下的 macro-subcircuit(也是子線路)然後以 ITA 模擬，我們發展了數種版本，但可以 ITA-WR[1](以 WR 為主程式，以 ITA 處理 macro-subcircuit)做為代表。我們認為以上的做法可以透過更精密的做法以獲得改進，那就是將線路分辨成更多種型態，如將 feedback portion 更進一步分辨成為 adjacent coupling portion(子線路間彼此有強烈交聯)和 directed loop portion(單純有向迴授圈的子線路)，因為這兩類電路型態在交聯(coupling)強度上其實是有差別的。但是 ITA-WR 的架構限制了這種更進一步的處理，於是我們轉向去利用較有彈性的 STWR 演算法。

STWR 使用精確的子線路排序法(subcircuit scheduling，子線路排序後以進行計算)，精確到一個子線路一個時間點的規格，任何一個子線路的的排序都要符合 exact-condition[5] 的規定，如此精細的子線路排序方式使得 STWR 演算法很有計算效率，它擁有較低的子線路計算總次數，另外它也有動態視窗(dynamic windowing) 的功能，它是 WR 的進階改良版，具有較好的計算效率，而且也有較好的收斂性。但是我們發現 STWR 還是有若干缺點，主要是處理強烈交聯電路(比如說 shift registers)和大的迴圈電路(比如是 ring oscillator)時其效率較差，因為雖然可以收斂，但會用掉多次的迴圈數。STWR 的缺點恰巧可用前述的電路型態辨認的做法彌補，所以此主題的目的就是利用 STWR 搭配電路型態辨認以獲取更好的演算法。

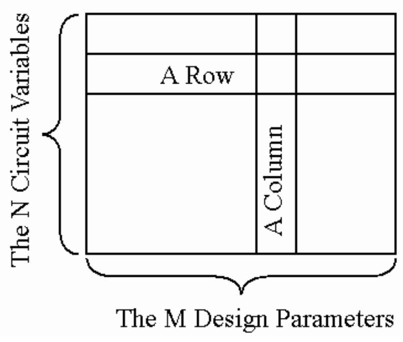
第二個主題討論漸增模擬(incremental simulation)[6][7][8]，這是用來模擬輕微變更設計(或變更輸入信號)的電路，而之前電路模擬的暫態波形被保存著可以加以利用，模擬的原則是盡量利用此保存的波形，比如說模擬時檢查電路模擬過程，只模擬有波形變化(和上次模擬結果不同)的部分，省去未變化部分的模擬而直接以上次的波形代替；若設計或輸入信號變更得很少，則漸增模擬可以省去許多模擬時間，大幅加快線路模擬的驗證速度。在 IC 設計過程中常會用到漸增模擬，設計過程中的人為調整或是電路最佳化程式的自動調整都會要求多次的線路模擬，而每次模擬所處理的電路設計通常只有稍微的變化，所以漸增模擬是很實用的一種技術。

一開始，漸增模擬是應用在數位電路的模擬上[6]，之後才應用到線路模擬上[7][8]，在線路模擬裡採用漸增模擬較為困難，這是因為子線路間常有雙向交聯關係，使得計算較為複雜；針對線路模擬的漸增模擬先後有兩種技術被提了出來，首先是使用 Incremental-in-Space 技術配合 ITA 演算法的做法[7]，後來則有使用 Incremental-in-Time 技術配合 WR 演算法的做法[8]，這些方法都是搭配基於鬆弛演算法，其原因是它們的彈性。但此兩種做法的缺點是效率性不佳，其中 Incremental-in-Space 的效率明顯極差，一旦某個子線路開始計算，就會計算到終點時間，而 Incremental-in-Time 較好，但是在某個 window 中，某子線路的計算也是從頭到尾的，也就是其子線路計算量的解析度成為 window 的大小。由於子線路計算量解析度的大小影響計算效率，所以我們希望使用最小的解析度，那就是時間點解析度，以獲取最佳的計算效率，我們就使用 STWR 演算法，因為它的子線路排序是精確到時間點的規格。

第三個主題討論的是多設計參數下敏感度計算的效率問題。傳統上敏感度的計算方法分兩種，是 Direct 法[9]和 Adjoint 法[10]，其中 Direct 法是比較自然好計算的方法，原電路經過微分處理後得到的敏感度電路，可以和原電路一起處理，我們可以使用原電路使用過的數值資料(Jacobin 等)和 time step 資訊，經濟快速地算出暫態敏感度；另一種 Adjoint 法則較不好計算，必須求取 Adjoint circuit 上的電路變數之值，再和原電路的電路變數一起計算以

求出敏感度，而 Adjoint circuit 上波形的計算需由時間軸的最終點開始反向往初始點方向求取，此意味著需要兩個模擬過程(pass)，第一個是求取原電路的波形，第二個是反向求取 Adjoint circuit 的波形和將所得配合原電路波形組合出敏感度，在第一個 pass 中必須存起第二個 pass 中會參考到的數值資料(如 Jacobin)，或者在第二個 pass 中要重新計算第一個 pass 中的數值資料，不管如何，都會額外要求計算時間或記憶體空間。但是 Direct 法的效率被其以設計參數為準的計算順序打了折扣，因為一般敏感度的取用是以變數為準的(比如一次求取對於某變數的所有敏感度)，我們看到圖一以了解這兩種計算法的計算順序，所表示的是敏感度矩陣，其中橫列代表某電路變數對於所有設計參數的敏感度，而直行則代表所有電路變數相對於某設計參數的敏感度，在實際應用敏感度的場合常常是問到某個電路變數對於所有設計參數的敏感度(也就是橫列)，比如說最佳化程式要最佳化某個特定的電路變數，那就要問到所有可調整的設計參數對其的敏感度；Direct 法計算敏感度的順序是採圖一中的直行方式(可以稱之為 column-major)，而 Adjoint 法則是採用橫列的方式(可以稱之為 row-major)，所以採用 Direct 法計算敏感度時，為了收集某個橫列的數值，就必須算出所有直行的數值，然後再收集之，這會造成很大量的計算浪費。所以，敏感度計算演算法的計算順序對效率來講是很重要的。

Direct 法的效率被其計算順序打了折扣，而 Adjoint 法的計算又太麻煩，那麼是否可以將 Direct 法加以改善使其以 row-major 的順序計算呢？而這就是本計畫第三個主題的目標。



圖一：所有電路變數對所有設計參數的敏感度矩陣。

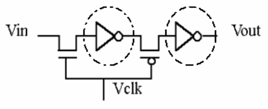
二、研究方法及成果

針對第一個主題，我們改進 STWR，搭配電路型態辨認，使用的方法如下：

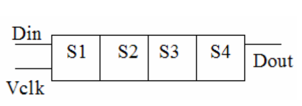
(1) 辨認模擬中電路，依子線路交聯程度將其分成 one-way portion(具有最低的交聯程度)，directed loop portion(具有中等的交聯程度)，和 adjacent coupling portion(具有最高的交聯程度)。

(2) 使用 STWR 作為主程式，額外使用 *synchronizing scheme* 處理 adjacent coupling portion，也額外使用 *global exact-condition scheme* 處理 directed loop portion，做法上就是將有迴授效果的兩種線路型態視為 macro-subcircuit。

我們就是利用了此兩個軟體機制解決電路中子線路交聯的問題。詳盡的演算法和效能比較請看到已於去年發表於香港的論文[11]，此論文為本報告的附錄。圖二是測試電路，表格一則紀錄了改良後的 STWR(STWR+)的效能，可見應用電路辨認是有較好結果的。表格二列出了所有測試電路的規格，表格三則列示這些電路的測試結果，我們可以發現 STWR+



(a)



(b)

圖二：(a) One-bit shift register 之線路圖; 虛線圈代表子線路。 (b) Two-bit shift register；共有四個子線路。

Table 1: Number of Subcircuit Calculations in Simulating Fig. 2.

Algorithm	*S1*	*S2*	*S3*	*S4*	*Total*
STWR	5,555	8,721	5,536	2,058	21,870
STWR+	3,832	4,083	4,261	3,333	15,509

Table 2: Specifications of Tested Circuits

Circuit	*Node#*	*MOSFET#*	*Subckt#*
1: 8-bit ALU	400	800	224
2: 8-bit Shift Register	32	48	16
3: 4-bit Binary Counter	88	176	44
4: CPU	465	946	224

Table 3: Running Results of Algorithms

Circuit	*Subcircuit Calculation #*			*CPU Time**		
	ITA	*WR*	*STWR+*	*ITA*	*WR*	*STWR+*
1	1,107K	222K	196K	17	11	7
2	70K	N. A.+	88K	2	N. A.	5
3	600K	261K	195K	9	16	9
4	3,401K	98.6K	94.1K	46	4.2	3.9

*: CPU time is in Pentium IV 1.4G second.   +: Due to divergence.

確實有較好的執行成效。

本計畫的第二個主題就是研究以 STWR 演算法進行漸增線路模擬的方法，我們將使用 STWR 作為模擬演算法，配合較精準版本的 Incremental-in-Time 技巧(較 Incremental-in-Time 要精密，原來是 incremental in time “window”，這個做法是 incremental in “time point”)，也就是稱為 *Incremental-in-Change* 的技術來獲致更好的漸增模擬效能，做法列示如下：

(1) STWR 演算法可以動態地自動計算 window 大小，有效解決一般 WR 演算法這方面的問題。

(2) 利用 STWR 以 time point 規格精密度 schedule 子線路的特點，精確追蹤子線路的變化度，子線路一旦停止波形變化(亦即又和儲存的波形吻合)便馬上停止模擬，以節省計算能量，如此便解決了前述 Incremental-in-Time 法中 global window 所引發的計算效率問題；我們稱呼這種擁有精密的子線路變化度追蹤效能的漸增模擬方法為 Incremental-in-Change，表示漸增模擬的啟動完全是根據子線路波形的變化度。

詳盡的演算法和效能比較還是一樣的參考到[11]。表格五列出了漸增模擬的效能，新方法叫做 IIC，測試電路中括號標示了設計有所變更的子線路，我們將變更大致調成 10%的範圍左右，可以見到模擬時間有大幅的減少，但是和子線路計算數減少的比例比起來卻沒那麼好，可見 IIC 的 overhead 是大的，這部分是需要繼續改善的地方。

Table 5: Running Results of Incremental Simulations

Circuit	*Subcircuit Cal. #*		*CPU Time**	
	STWR	*IIC*	*STWR*	*IIC*
8-bit ALU(Vcin)	135K	10.2K	5.3	1.8
Inv10(S1)+	53.3K	53.0K	1.3	1.4
Inv10(S6)	53.4K	41.7K	1.4	1.5
Inv10(S10)	53.3K	7.53K	1.4	0.9
4-bit Shift Register(S8)	85.7K	42.2K	4.2	2.6

+: S1 is the 1st subcircuit, counted from the end connected to primary inputs. *: CPU time is in Pentium IV 1.4G second.

第三個主題提到的是使用 Direct 法以 Row-major 方式求暫態敏感度的做法，我們所提出方法的重點列示如下：

(1) 推導出 Direct 使用的 row-major 敏感度計算公式。

(2) 基本做法是去使用基於鬆弛演算法計算敏感度。計算敏感度子線路時，根據子線路中節點數目，設計參數數目，和外接節點數等決定要用 row-major 或是標準的 Direct 法計算。

(3) 可以利用 row-major 法所產生的信號流強度以加速時序模擬的速度[12]，這麼做時敏感

度模擬必須和時序模擬同步進行。

(4) 使用 pruning scheme[13]針對需用敏感度的輸出節點修剪電路(刪去對輸出敏感度的計算無關的子線路) ，如此做對 row-major 計算法較為有利，可以增加其被切換到的頻率，整體而言可以大幅地加速計算。

計算的結果顯示，只有在子線路的節點數目較大時(約大於 20 個節點)，才會轉成 row-major 的計算方式，而獲取較佳的計算效能，根據子線路節點數目的不同，獲取的效能增進度從 10%到 20%不等。我們認為此部分的效率增進較為有限。

三、結論與討論

本計畫有三個主題，第一個主題完成後得到了應用詳細的子線路型態辨認於基於鬆弛演算法中的成果，帶引基於鬆弛線路模擬進入更佳的計算效能的境界。第二個主題完成後，得到不應用額外資訊下漸增模擬的最佳模擬成果，因為我們所達到的是 incremental in change，嚴格地只模擬有變化的子線路。第三個主題完成後，我們得到一個更進階的基於鬆弛大型電路敏感度計算方法，這是首度有人將基於鬆弛，Direct 法和 row-major 敏感度計算順序結合在一起的做法，對敏感度計算帶入了一個新的策略。

綜論之，去年的計劃研究順利，前兩個主題也已經發表在 IEEE 的國際研討會中，我們認為去年的研究計劃是成功的。不過研究成果還有改善的空間，所以今後我們將在此領域繼續研究。

參考文獻

1. Chun-Jung Chen, Shu-Chung Yi, Hui-Huang Hsu, and Weishing Liu, “Large-scale circuit simulation by using selective-tracing Waveform Relaxation and Nonlinear Relaxation,” The 1st International Workshop on Compact Modeling, Yokohama, Japan, pp. 50-55, January 2004, NSC-91-2215-E-034-001.
2. Chun-Jung Chen, Tai-Ning Yang, Shu-Chung Yi, Hui-Huang Hsu, and Weishing Liu, “Large-scale Circuit Simulation by Using Composition of Waveform Relaxation and Iterated Timing Analysis Algorithms,” Symposium on Design, Test, Integration, and Packaging of MEMS/MOEMS, Montreux, Switzerland, pp. 167-172, May 2004, NSC-91-2215-E-034-001.
3. Chun-Jung Chen, Wen-Pin Tai, Hui-Huang Hsu, 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.
4. 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.
5. Chun-Jung 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.
6. K. Choi, S. Y. Hwang, and T. Blank, “Incremental-in-time algorithm for digital simulation,” Proceedings of the 25th ACM/IEEE DAC, pp. 501-505, June 1988.
7. Y. C. Ju, F. L. Yang, and R. A. Saleh, “Mixed-mode incremental simulation and concurrent fault simulation,” Proceeding of the ICCAD, pp. 158-161, Nov. 1990.

8. Y. C. Ju, and R. A. Saleh, "Incremental circuit simulation using waveform relaxation," Proceeding of the ACM/IEEE DAC, pp. 8-11, 1992.
9. D. A. Hocevar, P. Yang, T. N. Trick, and B. D. Epler, "Transient sensitivity computation for MOSFET circuits," IEEE trans. on CAD, vol. CAD-4, pp. 609-620, Oct. 1985.
10. 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.
11. Chun-Jung Chen, Jung-Lang Yu, and Tai-Ning yang, "Selective-tracing waveform relaxation algorithm for Incremental circuit simulation," International Symposium on Intelligent Signal Processing and Communication Systems, Hong Kong, pp. 205-208, December 13-16, 2005.
12. 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.
13. Chun-Jung 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.

計畫成果自評

此次國科會計畫的執行相當徹底，在第一和第二個主題方面都有完整的良好的結果，第二個主題有關漸增模擬的部分更發表了國際研討會的論文，於 2005 年底於香港中文大學發表；但是第三個主題的結果較不動人，這個主題研究使用 row-major 順序去計算子線路裡面的暫態敏感度，希望減少計算的時間，因為子線路的大小已經受限，常常較好的做法就是直接使用 Direct Approach 去求解，用 row-major 法較有利的情況只發生在較大子線路的情形下，此類子線路較為稀少，所以相對下使得這個方法的功效打了折扣，不過這個方法是可行的，可以繼續發展，我們已經計畫在新的國科會計畫中繼續研究之。總之，很欣慰的，這次國科會執行順利，獲得了良好的結果。以下就各項目逐一說明。

研究內容與原計畫相符程度：可以說是完全相符。

達成預期目標情況：第一和第二個主題可說是完全達到預期目標，第三個主題也照預期製作完成，惟效果不如預期顯眼。

研究成果之學術或應用價值：在線路模擬和敏感度計算的應用價值是很高的。在 EDA(Electrical Design Automation)的學術價值也是高的。

是否適合在學術期刊發表或申請專利：適合發表論文，已經有國際研討會論文發表(見附錄)；因為是軟體，較不適合申請專利。

主要發現或其他有關價值：研究漸增模擬之後發現利用設計者的使用習慣可以大幅加速線路模擬，既然如此，似乎可以就這個方向進一步發揮，比如需求模擬(Simulation on Demand)的研究，只模擬設計者需要的部分電路，這將是我們未來的研究方向。

附錄

Chun-Jung Chen, Jung-Lang Yu, and Tai-Ning yang, "Selective-tracing waveform relaxation algorithm for Incremental circuit simulation," International Symposium on Intelligent Signal Processing and Communication Systems, Hong Kong, pp. 205-208, December 13-16, 2005, NSC-94-2215-E-034-001. **IEEE supported**.

SELECTIVE-TRACING WAVEFORM RELAXATION ALGORITHM FOR INCREMENTAL CIRCUIT SIMULATION[+]

Chun-Jung Chen, Jung-Lang Yu[*], and Tai-Ning Yang

Department of Computer Science, Chinese Culture University, Taiwan

*Department of Electronic Engineering, Fu Jen Catholic University

ABSTRACT

This paper discusses the large-scale circuit simulation and incremental circuit simulation, both which are based on Selective-tracing Waveform Relaxation (STWR) algorithm. Utilizing the concept of exactness in simulating subcircuits, STWR usually spends fewer subcircuit calculations than traditional algorithms, such as WR and ITA. But STWR suffers from robustness problems. Techniques are proposed to reinforce characteristics of STWR. This paper also describes the incremental circuit simulation that is based on STWR. This new method manages the influence of design changes more exactly than previous works, and hence exhibits better efficiency. We have implemented all proposed methods and performed experiments. Simulation results justify better characteristics of proposed methods.

1. INTRODUCTION

Electrical-level circuit simulation is quite important since it provides highly precise transient response waveforms of the designed circuits. Using this information, designers can understand the most detailed behaviors of their circuits and then complete their work smooth. But the time complexity of traditional circuit simulation algorithm, the *direct approach* used in circuit simulator such as SPICE, is high, so circuit simulation is quite time expensive in dealing with large-scale circuits. Relaxation-based algorithms [1, 2], which can bring the same results as that of direct approach, exhibit lower time complexities and therefore show better performance. Waveform Relaxation (WR) [1] and Iterated Timing Analysis (ITA) [2] are two traditional relaxation-based algorithms. There are numerous literatures discussing them and several commercial tools based on them. ITA is normally viewed as more robust than WR (for ITA has fewer convergence problems in simulating coupled subcircuits) and WR is normally viewed as more efficient than ITA (for WR can exploit circuits' multi-rate behaviors). Few years ago, a more advanced algorithm called Selectively-tracing Waveform Relaxation (STWR) [3] had been proposed. STWR utilizes the signal flow information more extensively and also capable of using multi-rate behaviors (it uses the *local-time-step scheme*) to get more computation efficiency. Consequently, STWR shows better performance than WR and ITA and, on the other hand, induces new problems about subcircuit scheduling and convergence. Since STWR is a quite efficient algorithm, we think we can improve its drawbacks to derive a powerful algorithm for large-scale circuit simulation. And this is the first purpose of this paper, i.e. to derive a new algorithm, *STWR+*, which is robust as well as efficient than traditional algorithms.

Our second purpose is to propose the incremental simulation using STWR. Incremental simulation is a useful technique to save simulation time for circuits that have been slightly modified. The circuit's original waveforms are stored. Incremental simulation algorithm then utilizes them to undertake simulation for the

+This paper was supported by Taiwan's National Science Council under contract NSC93-2215-E-034-001

modified circuit. The traditional electrical-level incremental simulations are those based on ITA and based on WR [4]. We utilize the detailed-scheduling characteristic of STWR to manage influence of design changes more exactly, which also be reflected on the saving of computation time.

The outline of this paper is as follows. Section 2 explains STWR and the reinforcing techniques, Section 3 describe the STWR-based incremental simulation, and Section 4 demonstrates experimental results of proposed methods. Finally, Section 5 gives a conclusion remark.

2. STWR ALGORITHM AND REINFORCING TECHNIQUES

The simulated circuit can be described as following time-varying differential equation:

$$F(Y(t),\dot{Y}(t),t)=0 \qquad (1)$$

Where Y is the vector of circuit variables, t is the time, F is a continuous function and "." means differentiation with respect to time. The simulated circuit must be partitioned into subcircuits, and the ith subcircuit is:

$$F_i(Y_i(t),\dot{Y}_i(t),D_i(t),\dot{D}_i(t),t)=0 \qquad (2)$$

For convenience, the subcircuit, say a, is expressed in the abbreviate form:

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

Where y (Y_i, a sub-vector of Y) is the vector of circuit variables in a, w (D_i, the *decoupling* vector) is the vector of circuit variables not in a, and f is a continuous function. In this paper, a *subcircuit calculation* means the computation efforts to solve (3) for $y(t_{n+1})$, t_{n+1} is a time point, which include applying integral formula to (3), and solving the derived nonlinear algebraic equations by Newton's method. Algorithm 1 lists pseudo codes of STWR.

Algorithm 1 (STWR Circuit Simulation Algorithm):

```
Input: Simulated circuit partitioned into subcircuits, primary
          inputs, and simulation time duration [$T_{begin}$, $T_{end}$].
Output: Time waveforms of all circuit nodes.
// $t_n$: previous time point; $t_{n+1}$: current solving time
// $t_c$: converged newest time point; $t_{ex}$: exactness checking time
STWR() {
    Reset all subcircuits' $t_c$ to $T_{begin}$;
    // PQ is a minimum priority queue, and q is a normal queue.
1:  Clear PQ and q;
2:  for(k = 1; $T_{convergence}$ != T ; k++) {
    // k is the relaxation index, and following is a WR relaxation
3:      for(all subcircuit x whose $t_c$ != $T_{end}$ ) {
4:          $x.t_{n+1} = x.t_c$;
5:          Put x into PQ;
        }
6:      while(PQ is not empty) {
7:          Delete the subcircuit a having minimum $t_{n+1}$ from PQ;
8:          Put a into q;
8a:         // synchronizing(a, q);
9:          while(q is not empty) {
10:             Delete one subcircuit a from q;
11:             Solve a, (3), at $a.t_{n+1}$ for answer $a.y(t_{n+1})$;
12:             if(The calculation is failed or $a.y(t_{n+1})$
                    is unacceptable) {
13:                 Shrink the time step of a, and add a into PQ;
14:                 continue;
                }
                // set convergence time
15:             if($a.t_n$ is converged) $a.t_c = t_{n+1}$;
16:             else Increase a's divergent counts;
17:             Store $a.y(t_{n+1})$, the transient responses, into tables;
18:             Estimate a's next time point and store it into $a.t_{n+1}$;
                // exact(a) implements the exact-condition
19:             if(exact(a)) add a into PQ;
20:             for(all fan-out subcircuit of a, say w) {
21:                 if(w is not in PQ and exact(w)) add w into PQ;
                }
22:             if(k > 1 && a has been not converged
                    for $tol_{divergent}$ time points) {
                    // $tol_{divergent}$ is a constant for window sizing
23:                 $T_{convergence}$ = the smallest $t_c$ of all subcircuits;
24:                 break;
                }
            }
        }
    }
}
// exact(x) == true if all x's fan-in subcircuits' $t_n > x.t_{n+1}$
25:boolean exact(x) {
26:     $t_{ex}$ = min{ $w.t_n$ | w is fan-in subcircuits of x};
27:     if($x.t_{n+1} \le t_{ex}$) return(true);
28:     else return(false);
    }
```

STWR uses selective-tracing scheme of ITA [2] to schedule subcircuits for calculation (line 20, 21). The traced subcircuit will not be calculated until it meets the exactness condition (all fan-in subcircuits have been calculated at time points newer than the solving time point, coded in exact() at line 25-28). Subcircuits are calculated at their own time steps, so multi-rate behaviors can be utilized (local time steps are estimated in line 18, and used at line 11). Besides the order of subcircuit calculations, STWR is very similar to WR since it performs relaxation technique on subcircuits' nonlinear differential equations and also exploits the windowing technique [1] (at line 22, STWR decides the sizes of windows according to subcircuits' convergence situations, i.e. that any subcircuit has been not converged for $tol_{divergent}$ terminates a relaxation).

The advantages of STWR had been illustrated in [3], major of which is to exploit *exactness concept* to reduce the number of subcircuit calculations. However, STWR has its problems. The first one is the lack of efficiency in solving strongly coupled subcircuits (due to the local-time-step scheme), and the second one is the "dead-lock" problem in dealing with global feedback subcircuit loop (consider a global feedback loop with two subcircuits only and function exact(), each subcircuit will wait for another one in order to get its exactness condition). We use the subcircuit classification technique in [5] to solve these problems. According to signal flow graph of the circuit, we classify subcircuits to those in *strongly connected components* (SCC), those in feedback loops, and those in one-way portions. The *synchronizing-scheme*, which synchronizes time points of subcircuits in the same SCC, is used to solve STWR's first problem:

Algorithm 2 (Synchronizing-scheme):

```
// the synchronizing scheme for adjacent coupling region
**void** synchronizing(subcircuit a, queue q) {
1:  if(a belongs to a SCC) {
2:      for(all subcircuits of the same SCC, w) {
3:          $w.t_{n+1} = a.t_{n+1}$;
4:          Put w into q;
        }
    }
}
```

This algorithm is to be put in line 8a of Algorithm 1. Line 9 of Algorithm 1 is added (when compared to [3]) to adopt the synchronizing-scheme. The second problem can be solved by the *group-exactness scheme*, in which the exactness condition of each subcircuit in the feedback loop, say *L*, is determined by *L*'s fan-in subcircuits rather than this subcircuit's fan-in subcircuits (line 26 of Algorithm 1). These two techniques have been implemented and tested, which will be described later.

3. STWR-BASED INCREMENTAL CIRCUIT SIMULATION

A good incremental simulation algorithm should only re-simulate portions of waveforms differ from those of previous simulation as possible. But real incremental simulation algorithms usually simulate additional subcircuits or time intervals. For convenience, we denote these additional calculations the *over-traced* calculations, and *over-traced ratio* the ratio of the number of over-traced subcircuit calculations to that of total subcircuit calculations. The smaller the over-traced ratio is, the better the incremental simulation algorithm is.

Recall that there are two previous works for incremental simulation. The first one is the ITA-based Incremental-in-space approach [4]. In this approach, the portions of the circuits affected by design changes are selectively traced and are re-simulated. The re-simulation of a subcircuit starts from the time at which it is affected by design changes to the end of the simulation duration. The time factor (e.g. a subcircuit is not affected later) is not considered in this approach, so its over-traced ratio is high. The second approach is WR-based Incremental-in-time approaches [4]. This approach further considers the time factor. If an affected (by design changes) subcircuit is no longer affected, it could be discarded from the re-simulation area. In [4], inside a window subcircuits are selectively traced and re-simulated till they converge. Succeeding windows are processed sequentially in the same way. So, the influences of design changes are captured in the "resolution" of windows. The problem of this approach is that some subcircuits are only affected for a small portion of the window (over-traced calculations would be induced), and window sizes are usually decided by simulation situations of the complete simulation (in which big windows appear frequently).

STWR precisely controls the execution of each subcircuit calculation. Therefore, it can trace the influence of design changes exactly (in the "resolution of time point"). The major idea of STWR-based method is to capture whether subcircuits' behaviors differ from those of previous simulation dynamically and according to which to arrange the incremental-simulation. If a subcircuit's waveform currently differs (affected by any design change), it was in the *changing situation* (called *changing-subcircuit*). We assign each subcircuit a variable called *s-state* (subcircuit state) to dynamically record this situation. We summarize how s-states guide the incremental simulation, how changing situations appear, disappear, and propagate, and other actions by following rules.

Rule 1 (STWR-based Incremental Circuit Simulation):

1. **Simulation Rule**: The s-state of a subcircuit is either "changing" or "following." During simulation, only the changing-subcircuits is re-simulated.
2. **Subcircuits' Design Change Rule**: The subcircuit whose internal design has been modified always in changing situation.
3. **Inputs' Design Change Rule**: Primary inputs are viewed as "pseudo" subcircuits that have s-states, too, and contents of whose s-states are dynamically determined by whether the values of primary inputs differ from those of previous simulation.
4. **Vanishing Rule**: Once a changing-subcircuit has been calculated, the obtained waveform will be compared to that of previous simulation to see whether the changing situation vanishes.
5. **Propagation Rule**: A changing-subcircuit passes the changing situation to its fan-out subcircuits.

The pseudo codes, more practical for implementation, for these rules are:

Algorithm 3 (Incremental Simulation Programs):

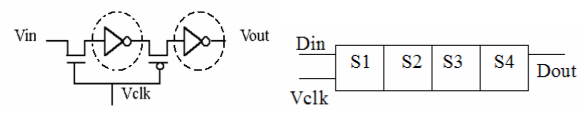
```
    /* For #1 rule. To replace line 11 of Algorithm 1 */
25:if(a.s_state == CHANGING)
26:     Solve a at $a.t_{n+1}$ for values of circuit variables;
27:else Copy the stored waveforms as a's answer;
    /* For #4 rule. To replace line 19 of Algorithm 1 */
28:if(exact(a)) {
29:     Add a into PQ;
30:     if(a.s_state == CHANGING &&
31:          a's solution is equal to the stored waveform)
32:          a.s_state = FOLLOWING;
    }
    /* For #5 rule. To replace line 21 of Algorithm 1 */
33:if(w is not in PQ and exact(w)) {
34:     Add w into PQ;
35:     if(w.s_state == FOLLOWING &&
          a.s_state == CHANGING)    w.s_state =
CHANGING;
    }
```

These codes have been implemented, and the experimental results are shown in Section 4.

4. EXPERIMENTAL RESULTS

The proposed methods have been implemented in the experimental circuit simulator MOSTIME [3, 5] that contain several relaxation-based algorithms. To emphasize simulation algorithms' effects, we use the simple analytic model for MOSFET and partition circuits manually. We use the circuit in Fig. 1, which has four strongly coupled subcircuits, to demonstrate the synchronizing-scheme. Since all subcircuits are in a SCC, they all use the same time points. Running results are collected in Table 1, which shows about 25% of calculations have been saved. We then compare STWR+ with other algorithms. Four circuits in Table 2 are tested, and their schematics are shown in Fig. 2. Table 3 summarizes the simulation results. We can see that STWR+ has used fewest subcircuit calculations and simulation time in all cases, and it is more robust than WR (see circuit #2), which justify that STWR+ is robust as well as efficient.

We now compare the incremental simulations. The new method (called *Incremental-in-change* since it intends to simulate changing subcircuits only) is compared to Incremental-in-space by simulating Fig. 2(d), Gated-chain(V2) (where "(V2)" means V2 is modified). In our experiments, ratios of design changes are made to be about 10% for all examples. Table 4 records the results. We note that both algorithms don't calculate S8 since S8 doesn't change all the time, and IIC has performed less subcircuit calculations for S5 since S5 only changes for a period only. The over-traced ratio of IIC is obviously smaller than that of IIS. Now we use IIC to simulate larger circuits. Table 5 is the running results. The waveforms of first example, 8-bit ALU, are shown in Fig. 3 (with Vcin modified for about 10% of the time interval). We can see that IIC has generated correct waveforms. This example also shows considerable speedup on simulation time.



(a)
(b)
Fig. 1: (a) The schematic of one-bit shift register; dashed circles are subcircuits. (b) Two-bit shift register; it has four subcircuits.

Table 1: Number of Subcircuit Calculations in Simulating Fig. 1.

Algorithm	*S1*	*S2*	*S3*	*S4*	*Total*
STWR	5,555	8,721	5,536	2,058	21,870
STWR+	3,832	4,083	4,261	3,333	15,509

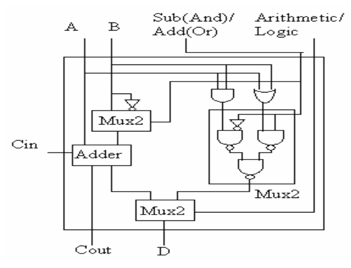
Table 2: Specifications of Tested Circuits

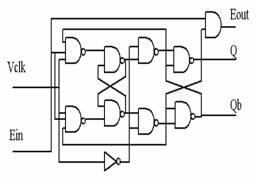
Circuit	*Node#*	*MOSFET#*	*Subckt#*
1: 8-bit ALU	400	800	224
2: 8-bit Shift Register	32	48	16
3: 4-bit Binary Counter	88	176	44
4: CPU	465	946	224

Table 3: Running Results of Algorithms

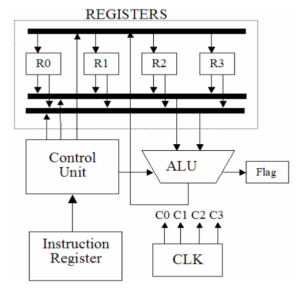
Circuit	*Subcircuit Calculation #*			*CPU Time**		
	ITA	*WR*	*STWR+*	*ITA*	*WR*	*STWR+*
1	1,107K	222K	196K	17	11	7
2	70K	N. A.+	88K	2	N. A.	5
3	600K	261K	195K	9	16	9
4	3,401K	98.6K	94.1K	46	4.2	3.9

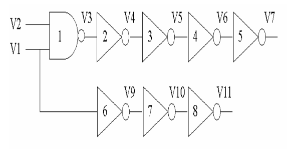
*: CPU time is in Pentium IV 1.4G second. +: Due to divergence.





(a) ALU
(b) Synchronized Counter





(c) 4-bit CPU (d) Gated-chain

Fig. 2: Schematics of tested circuits.

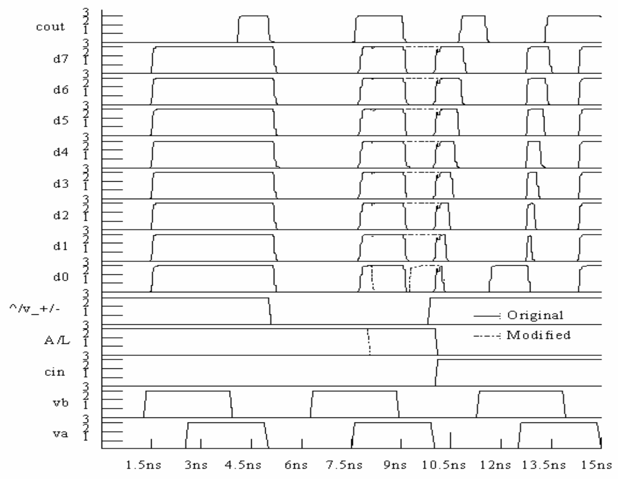


Fig. 3: Time waveforms of 8-bit ALU, in which Va and Vb are connected to all ALUs.

Table 4: Subcircuit Calculation # of Gated-chain(V2), Fig. 2 (d).

Subcircuit	***STWR***	***IIS***$^+$	***IIC****
S5	700	650	182
S8	1,132	0	0
ALL	7,270	3,626	1,044

+: Incremental-in-space, *: Incremental-in-change

Table 5: Running Results of Incremental Simulations

Circuit	***Subcircuit Cal. #***		***CPU Time****	
	STWR	*IIC*	*STWR*	*IIC*
8-bit ALU(Vcin)	135K	10.2K	5.3	1.8
Inv10(S1)$^+$	53.3K	53.0K	1.3	1.4
Inv10(S6)	53.4K	41.7K	1.4	1.5
Inv10(S10)	53.3K	7.53K	1.4	0.9
4-bit Shift Register(S8)	85.7K	42.2K	4.2	2.6

+: S1 is the 1st subcircuit, counted from the end connected to primary inputs. *: CPU time is in Pentium IV 1.4G second.

The following three experiments in Table 5 use the same 10-stage CMOS inverter chain with subcircuit S1, S6, and S10 modified respectively. According to the position of the modified subcircuit, the calculation efforts differ. The less subcircuits change in the re-simulation, the less efforts IIC does, which is the necessary characteristic for incremental simulation. The final example is a strongly coupled circuit with 16 subcircuits, which is used to show that our method can work on this kind of circuit well, too.

5. CONCLUSION

In this paper we present supplement techniques for STWR to reinforce its efficiency as well as robustness in dealing with circuits with strongly coupled subcircuits or with feedback loops. The new electrical-level incremental simulation algorithm based on STWR is also proposed, which manages the influence of design changes more exactly than previous works and exhibits better simulation performance. We think to design an index for the “change-tracing” degree of incremental simulation is necessary. Also, the method to store only portions of all waveforms might be considered in our future work.

6. REFERENCES

[1] A. R. Newton and A. L. Sangiovanni-Vincentelli, “Relaxation-based electrical simulation,” IEEE Trans on CAD, Vol. CAD-3, pp. 308-311, Oct. 1984.

[2] 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.

[3] 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.

[4] Y. C. Ju, and R. A. Saleh, “Incremental circuit simulation using waveform relaxation,” Proceeding of the ACM/IEEE DAC, pp. 8-11, 1992.

[5] C. J. Chen, T. N. Yang, S. C. Yi, H. H. Hsu, and W. Liu, “Large-scale circuit simulation by using composition of Waveform Relaxation and Iterated Timing Analysis Algorithms,” Symp. on Design, Test, Integration, and Packaging of MEMS/MOEMS, Switzerland, pp. 167-172, May 2004.

出席國際學術會議心得報告

計畫編號	NSC 94-2215-E-034-001
計畫名稱	應用電路型態辨認於選擇走訪波形鬆弛演算法，漸增電路模擬，和對於多設計參數之暫態敏感度的同步計算
出國人員姓名 服務機關及職稱	陳俊榮副教授 中國文化大學資科系
會議時間地點	Hong Kong, December 13-16, 2005
會議名稱	International Symposium on Intelligent Signal Processing and Communication Systems
發表論文題目	Selective-tracing waveform relaxation algorithm for Incremental circuit simulation

一、參加會議經過

在 2005 年末本人使用 94 年國科會的經費出席此國際性會議，此會議的主題鎖定在智慧信號處理和通信系統，包含有本人計劃的領域。會議選在中國的香港舉行，會址在香港中文大學。我的部分是 poster 發表，由於是個中型的研討會，所以與會的學者頗多，因此也被許多人問到了相關問題，也有了許多的討論，這使我我獲得了許多收穫；另外聆聽其他學者的報告也讓我受益良多。

二、與會心得

本人發表論文所使用的方法是新奇的，但是還有許多發展的空間，主要是在實作層面的，因為我的方法理論上可以獲得最佳結果(漸增模擬的比例和擾動的比例相當)，但是計算時間上卻沒有類似的結果，這就是實作的問題了，所以這個研究有必要繼續下去。另外，要進行漸增模擬有許多的另類方法，我在這趟旅程中也思考這個問題，綜合和學者討論的結果，將一些想法都寫到了 95 年的國科會計畫申請書中，而這個計畫現在正在進行中，這也是參加此次研討會的收穫。

此次與會有許多收穫，看了到別人最新的研究成果，開拓了眼界，有了未來研究方向的概念，我看了許多新研究的 EDA Tool 論文，知道 EDA 工具的進展狀況，也詢問他們設計 tool 所遇到的問題，以及未來可能的作法等，如此，可以多多掌握最新的狀況，總之，此次會議之後我對於電路驗證的研究有了些一些新的想法，可以作為我未來研究主題題目。我對於大

型電路電路階層驗證研究狀況的感覺是分成了多種不同精確度的層級，較不精確的層級就是 switch level 和 macro model level 等，較為精確的層級就是 Spice accuracy level，在此種 level 下，連接線已經普遍加入模擬了，因為連接線主宰延遲的現象是越來越明顯，所以 Spice 所使用的 Direct approach 仍大量的使用，而 partitioning 常使用在快速的線路模擬，這也是本人近年的研究範疇，不過推測商業化程式多還是使用 ITA 演算法，並未有使用信號流強度模擬，或多種演算法混用的方式，所以我想商業化程式還有許多發展的空間。另外，傳輸線的分割模型似乎沒有太大的進展，許多傳輸線交聯在一起會大大的減弱模擬的速度，但這個問題未有很好的解決方法，未來，可以往這方面進行。

最後，漸增模擬似乎可以加以擴展，這個技術本來就是利用設計者的使用習慣去節省計算時間的，既然如此，可以就這個觀念繼續發揮，比如說只模擬設計者要看的部分，也可以節省不少的計算時間，這也是此次參加會議的收穫。

所發表論文

Chun-Jung Chen, Jung-Lang Yu, and Tai-Ning yang, “Selective-tracing waveform relaxation algorithm for Incremental circuit simulation,” International Symposium on Intelligent Signal Processing and Communication Systems, Hong Kong, pp. 205-208, December 13-16, 2005, NSC-94-2215-E-034-001. **IEEE supported**.

請見研究成果報告，或看到 http://staff.pccu.edu.tw/~teacherchen/ispac05.pdf