行政院國家科學委員會專題研究計畫 成果報告

FSTA：正規靜態時序分析技術(II)
研究成果報告(精簡版)

計 畫 類 別 ：個別型
計 畫 編 號 ：NSC 95-2221-E-002-396-
執 行 期 間 ：95 年 08 月 01 日至 96 年 07 月 31 日
執 行 單 位 ：國立臺灣大學電子工程學研究所

計 畫 主 持 人：黃鐘揚

計畫參與人員：博士班研究生-兼任助理：蔡師蘅、黃紹倫、周俊男
碩士班研究生-兼任助理：李晨豪、吳濟安

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 ：本計畫可公開查詢

中 華 民 國 96 年 12 月 13 日

行政院國家科學委員會補助專題研究計畫成果報告

※※※※※※※※※※※※※※※※※※※※※※※※※※
※　　　**FSTA：正規靜態時序分析技術(II)**　　　※
※※※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：■個別型計畫　　□整合型計畫
計畫編號：NSC 95-2221-E-002-396
執行期間：　95 年 8 月 1 日至 96 年 7 月 31 日

計畫主持人：黃鐘揚　台灣大學電機系暨電子所助理教授
共同主持人：
計畫參與人員：
蔡詩薾　台灣大學電子工程學研究所
李晨豪　台灣大學電機工程學研究所
黃紹倫　台灣大學電子工程學研究所
吳濟安　台灣大學電子工程學研究所

執行單位：台灣大學電子工程學研究所

中　華　民　國　96　年　7　月　31　日

FSTA: 正規靜態時序分析技術(II)

"Formal Statistical Timing Analysis"

計畫編號：NSC 95-2221-E-002-396
執行期間：95 年 8 月 1 日至 96 年 7 月 31 日

主持人：黃鐘揚 台灣大學電機系暨電子所助理教授

一、中文摘要

本計劃今年專注於使用正規驗證方法，建立更精確的靜態時序分析工具，主要完成的項目有四：(一)倒推式靜態時序分析，(二)謬誤路徑檢驗，(三)電路多重轉換情況之最大延遲測試訊號的產生，以及(四)動態時序驗證。其中項目。在項目(一)中，我們實作一種能正確判斷關鍵路徑的靜態時序分析工具[1]，我們以之取得初始之關鍵路徑，並作為最後實驗結果之對照組。項目(二)中，我們使用正規引擎對此條路徑是否為謬誤路徑做檢驗。假定此路徑為真，則在(三)中我們同樣利用正規引擎，產生造成電路延遲極大的測試訊號，再以項目(四)的動態時序分析來檢驗我們所得到的電路延遲。如此一來我們的時序分析工具可以提供使用者做電路關鍵路徑的確認，以不錯的效率找到最大的電路延遲信號，而這些信號對使用者在電路怖局的時序模擬是相當有幫助的。

關鍵詞：時序分析，時序模擬，時序關鍵路徑，偽路徑，正規方法引擎，動態時序模擬，靜態時序模擬，多重輸入信號變換。

英文摘要

In this year we dedicated to establish a precise static timing analysis (STA) tool on the formal verification engine. Specifically, we accomplished four major subjects: (1) we have implemented a static timing analysis tool using backward signal propagation[1]. (2) We perform false path checking on critical path in the circuit by Boolean Satisfiability (SAT) engine. (3) If this path is true, we will try to push the delay bound obtained in the STA process by considering simultaneous input transitions of the circuit. (4) If we get certain input patterns from STA engine, then we can evaluate the new delay data by dynamic timing analysis (DTA). In this way we can get an accurate delay bound and solve the circuit false path problem at the same time. Moreover, because we only simulate several input patterns in the circuit, the performance of our method would be better than traditional DTA. In our experiments, we demonstrate that our method can achieve both better performance and accuracy than random simulation. As a result, we believe that the test patterns we derived by the formal engine would be useful and important to the post-layout timing simulation stage.

Keywords：Timing analysis, timing simulation, critical path, false path, formal verification engine, dynamic timing analysis, static timing analysis, multiple input transitions.

二、計畫的緣由與目的

Timing verification is a very important step in the current IC design flow [2]. It analyzes the delays of all the paths in the circuit under verification and checks whether the most critical ones meet the timing constraints. In this way, we can guarantee the circuit can function correctly.

There are two methods in performing timing analysis. The first one is called ***Dynamic Timing Analysis (DTA).*** DTA is a simulation-based approach that applies input patterns/transitions to the circuit and observes the output waveforms for the delay information. The advantage of this method is that it can accurately compute the delay for the applied pattern. However, since the number of simulation patterns is exponential to the number of inputs in a design, it is practically impossible to emulate all the patterns to obtain the critical delay information for the whole circuit.

Static Timing Analysis (STA), on the other hand, is an alternative approach [3]. It is more popular than DTA in timing verification because it does not require the user to provide the input patterns yet still can achieve reasonably accurate timing verification results with very high speed and capacity. It usually adopts the table look-up model for the cell delay calculation, where the ***input transition time*** and ***load capacitance*** are two indices for the delay table. Delay information and critical path are then determined by composing the worst case of single input pin transitions.

Although STA works well most of the time for the circuit designers, it has some limitations. First, this method usually leads into approximate results because it only considers single input pin transition. In real-life circuit operations, inputs can transit simultaneously. If we take the multi-input transition effect into consideration, the cell delay can be larger and thus the timing constraint may be violated. Another limitation on STA is its capability in detecting false paths. An example of a false path is as shown in Fig 1. If the determined critical path is a false path, we should try to find out another one whose delay bound is next to the original one. Moreover, STA does not provide test pattern to users. In the post-layout simulation stage, users usually apply device-level simulation tools such as SPICE with more accurate delay model. In other words, they will need the input patterns that can simulate the most critical delay situations.

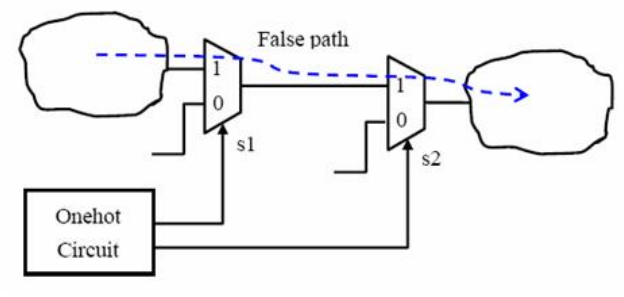


Fig.1 False Path Problem

In this project, we proposed a new algorithm in timing verification. It combines the advantages of STA and DTA as shown in Fig.2. We integrated the timing analysis and formal verification technologies to improve the efficiency and accuracy and at the same time provided a better timing analysis tool to IC designers.

Our contributions include: (1) We consider multiple transition delay model in order to approach the actual delay bound. In our experiments, we demonstrate that our method is more accurate in computing the critical path delay by as much as 10% and 50% when compared to the traditional STA and the random DTA, respectively. (2) We use Boolean Satisfiability (SAT) engine [4] to verify critical path from STA. If the critical path is false, timing analysis to this path is meaningless. Our method can avoid false alarm in timing analysis and report to user if there are false paths from STA. (3) Our method also provides input patterns for the most timing critical path delay [5]. In other words, these patterns guarantee to witness a transition to propagate along the critical path with the worst delay in timing simulation. They can also be used in diagnose the timing violations.

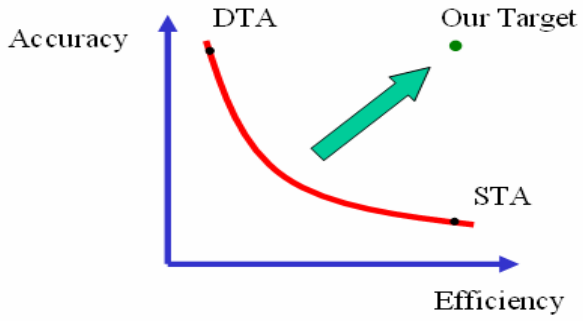


Fig.2 Efficiency and Accuracy of Timing Analysis

三、多重轉化時序資料庫之建立

3.2 Timing Analysis Considering Multiple Transitions

In this section we will describe the delay modeling, explain our experiments and observations, and show how we reduce the complexity in the simulation problem.

3.2a H-Spice Experiments to Collect Multi-Input Transition Delay Table

Before we started developing our timing engine, we had to conduct many experiments in Perl [6] and Spice [7] for multi-transition switch-level simulations. By enumerating various circuit constraints such as input transition delays and equivalent gate-output load capacitances, we got lots of data for delay time and behavior in different operating conditions [8]. Following are our summary and understanding about our external experiments.

In our Spice experiments (Fig. 3), we found that the transition time of single input pin and cell load capacitance can not dominate delay behavior. Multi-transition inputs sometimes lead to more critical delay. Actually the delay information is according to several variables such as transition time of all input pins, difference of all input pins' arrival time, and cell load capacitance. Different compositions of these variables will result in different delay behaviors.

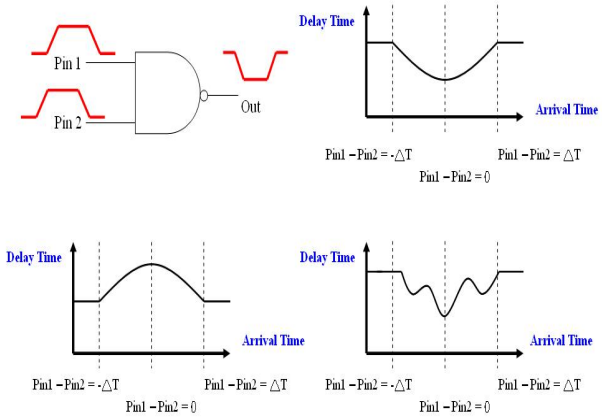


Fig. 3 Timing Experiments with Multiple Input Transitions

However, if we deal with all these variables directly, the problem grows exponentially with circuit size and the complexity of circuit components. In order to speed up our performance and reduce evaluation complexity, we combined STA and DTA in our method. We take advantage of their benefits in timing analysis and convert this problem into formal domain, solving false path problem at the same time.

3.2b Multi-Input Transition Delay Calculation Model

As in most timing analysis algorithms, our timing analysis method works on all paths located between prime inputs and registers or between registers and prime output (Fig. 4) in combinational circuits. In order to get an accurate delay bound, we consider the possibility of signal propagation with multiple transitions at some logic stage or at prime inputs. That is, our delay model will depend on more complex variables.

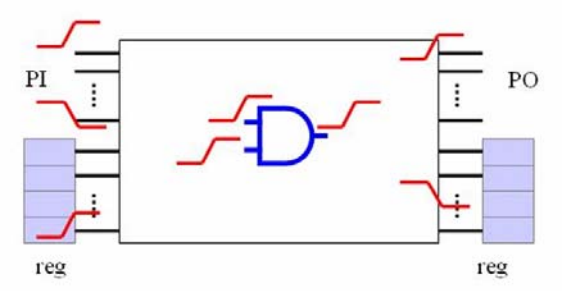


Fig. 4 Simulation with Multiple Input Transitions

Our timing analysis method is the same as STA. It does not need input patterns from the user. Instead, we use formal engine to find out proper patterns or transitions from all input patterns. By incremental ***Boolean constraints propagation (BCP)*** and delay calculation, we guarantee the correctness and robustness of our delay model.

In this thesis we focus on the behavior and data collection of **2-input gate**. We introduce AND2, OR2, NAND2, NOR2, INV and XOR2 gates into our consideration. Note that binate gate XOR2 would not propagate multi-transition signals so we just consider single transition when we meet an XOR2 on the concerned path. This is shown in Table.1. Considering constraints such as zero, one, rise, and fall to input pins, the output response would have **2^4 = 16** combinations as zero, one, rise, and fall. Table.2 is the 2-input NAND gate truth/transition table. However, we found that the combinational response can be classified as stable zero, stable one, ***rising with single input transition*** **(R_s)**, ***falling with single input transition*** **(F_s)**, ***rising with multiple input transitions*** **(R_m)**, ***falling with multiple input transitions falling*** **(F_m)**, and *hazard/glitch (*)* as shown in Tables below.

Table.1 2-Input XOR Gate Truth/Transition Table

	0	1	Rise	Fall
0	0	1	**R**	**F**
1	1	0	**F**	**R**
Rise	**R**	**F**	**0**	**1**
Fall	**F**	**R**	**1**	**0**

Green：Single input transition
Blue：Multiple input transitions

Table.2 2-Input NAND Gate Truth/Transition Table

	0	1	Rise	Fall
0	1	1	1	1
1	1	0	**F**	**R**
Rise	1	**F**	**F**	*
Fall	1	**R**	*	**R**

Green：Single input transition
Blue：Multiple input transitions

Reducing the complexity of logic gate output response is helpful for our delay calculations. Next step we would try to define certain delay variables to our timing model. As Table.3 shows, we have four kinds of delay variables — gate ***input transition time***, represented as Tr, gate input signal arrival time, represented as Arr, ***difference of input arrival times***, represented as Darr or Delta T, and finally the equivalent ***output load capacitance***, represented as Ceq. Notice that the Delta T can be zero, positive, or negative.

Delay Variables	Description
Transition Time (Tr x)	Transition time of input pin
Arrival Time (Arr x)	Arrival time of input signal
Difference of Arrival Times (Darr or △T)	May be positive or negative
Load Capacitance (Ceq)	Output load for logic gate

Table.3 Delay Variables and Description

3.2c Boundary Conditions

In Section 3.2 we have explained our delay variables. As we know, the more variables to the delay calculation, the more complexity in solving this problem. Fortunately, difference of input arrival times (**Darr**) can be simplified to reduce the delay variable combinations.

As Fig. 5 shows, when we consider 2-input unate gate, difference of input arrival times has some extremes. In one situation, we know that **Darr** can be treated as positive infinite when **Arr1 – Arr2 > Tr2**. In the other situation, **Darr** can be viewed negative infinite when **Arr1 – Arr2 < -Tr1**.

Extreme of **Darr** means that considering multiple input transitions is not necessary. Instead, the output response would be the same as single input transition. Sampling in the gap of **Darr** can help us to reduce the complexity of delay calculation.

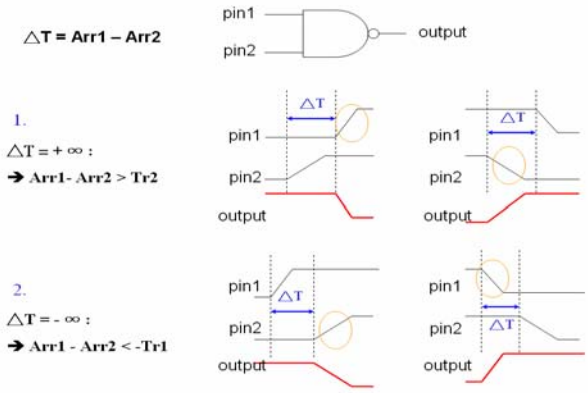


Fig. 5 Extreme of Delta T

3.2d Multiple Transition Timing Table

In the previous section we have explained the variables for delay calculation. Now we talk about how we implement it. As Fig.6 shows, for a 2-input unate gate, we construct a 4-dimensional timing table with four variables. Each variable have five indexes. Tr1 and Tr2 set up 5 x 5 = 25 timing tables. These timing tables have 5 x 5 = 25 delay information determined by Darr and Ceq.

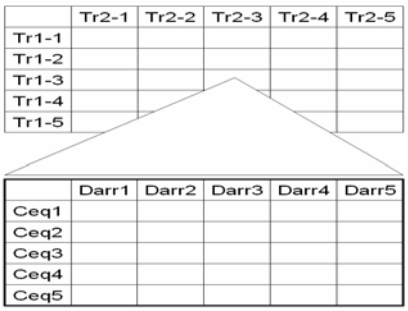


Fig. 6 Timing Table for Multiple Input Transitions

四、研究方法與成果

4.1 Problem Definition

4.1a Input of Our Problem

Our timing analysis method needs gate-level design files, including output load description. We also need both single and multiple transition delay tables in the cell library file to calculate delay information. Because there is no library file with multiple transition tables now, we construct it by conducting a great amount of SPICE simulations.

4.1b Output of Our Problem

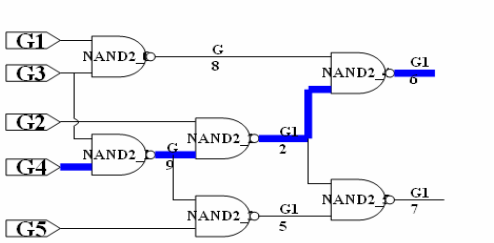
Our method would report various delay information corresponding to each stage. It would report critical path from either STA or DTA, current and maximum delay bound through timing analysis, change of delay bound or critical path, input patterns generated by SAT solver, and run time and memory usage in this analysis.

4.1c Assumptions and Simplifications in Our Method

In order to quickly a prototype of our algorithm, we consider only 2-input NAND gates. This is because that multiple transition tables are much more complex than single transition ones. Multiple input pins also leads to complex permutations on input transitions. We construct the prototype of our timing analysis method and hope that this method can be performed in real cases in the future.

4.1d An Illustrative Example

An example to demonstrate the multi-transition effect on the circuit delay is shown in Figure 7. The case “C17” is composed of six 2-input NAND gates. Each gate has its own load capacitance. In the figure we can see that the critical path from STA is a true path and it is the same critical one from our method. However, we can generate a test pattern that has two transitions on inputs G2 and G4 and makes the delay bound more critical than STA. Detailed experiments would be showed in following sections.



	STA	Our Method
Delay Bound	6.9089e-10 s	7.6646e-10 s

Input Pin	Pattern
G1	Zero
G2	Rise
G3	One
G4	Fall
G5	Zero

Fig.7 Example for Test Case

4.2 Algorithm and Framework Architecture

4.2a Main Algorithm

Fig. 8 is the pseudo code of our main algorithm and Fig. 9 shows its flow chart. We first parse circuit into the predefined circuit base data structure. Next step we get the initial critical path of the circuit by performing STA [8] [9]. Before we add Boolean constraints to this path, we do false path checking to guarantee this path is a true path [10].

```
mainAlgorithm(){
01. Circuit construction;
02. Perform STA and get critical path from STA;
03. False path checking to critical path
04. If (SAT)          go to step 06;
05. Else              Report false path, go to step 02 and
                      get next critical path;
06. For each gate in critical path
07.     Add proper Boolean constraints to this gate;
08.     If (SAT)      go to step 11;
09.     Else          go to step 07;
10. Solve a pair of input patterns from SAT solver;
11. Perform DTA;
12. If (critical path change)          go to step 03
13. Else if (get new delay bound )     go to step 15
15. Stable, end;
}
```

Fig. 8 Main Flow Algorithm

After all constraints are confirmed, we can get a pair of input patterns from SAT solver. In this way we can verify timing information by performing DTA with these patterns [11]. If critical path is changed, we should return to previous stage and try again until we get a new delay bound.

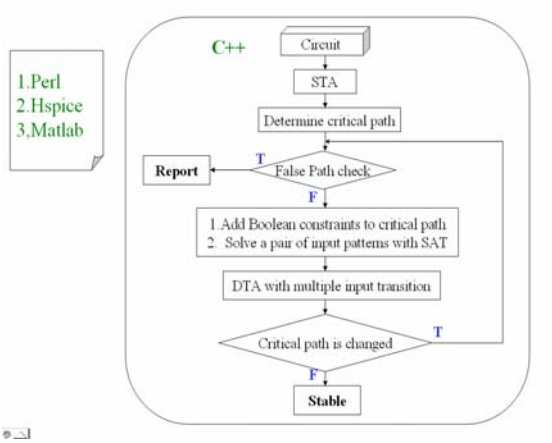


Fig. 9 Flow Chart of Main Algorithm

4.2b Framework Architecture

Fig.10 is the developed framework architecture. It contains several components such as ***front-end parser***, ***timing engine***, and ***SAT engine***.

Front-end parser parses circuit file and converts the circuit into the defined circuit base data structure.

Timing engine is divided into STA and DTA. It evaluates timing information and determines circuit critical path.

SAT engine is composed of SAT solver and Boolean constraints. It would be helpful for DTA and finding out the critical patterns.

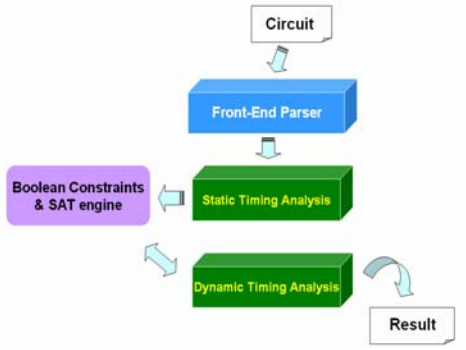


Fig.10 Framework Architecture

4.3 Experimental Results

In this section we show our new timing analysis experiments. We have several tables to point out delay bound difference of STA and our new algorithm. Moreover we have performed random simulations to the same test cases, to figure out the relationship of efficiency and accuracy.

4.3a Delay Bound Refinement with New Algorithm

Table.3 is the delay bound refinement with our new algorithm. We can see that our algorithm always finds out more critical input constraints, that is, our method finds out more critical delay bound that STA. Average improvements now is up to 3 percentage.

Name	Gate/Level	STA (s)	NTA (s)	Diff (%)
C17	6 / 3	6.91e-10	7.66e-10	10.9%
C001	14 / 4	9.29e-10	9.69e-10	4.3%
C002	17 / 5	1.08e-09	1.07e-09	---
RnGen_3_3_3	8 / 3	7.51e-09	False Path	
Rn_9_3_11	58 / 11	1.45e-09	1.49e-09	2.3%
RnGen_15_4_10	63 / 10	1.16e-09	1.24e-09	6.9%
Rn_20_10_15	133 / 15	2.28e-09	2.29e-09	0.4%
RnGen_1000_200_10	3662 / 10	1.67e-09	1.71e-09	2.4%
RnGen_4500_300_10	3272 / 10	1.29e-10	1.37e-09	6.2%
RnGen_2000_200_10	5801 / 10	1.77e-09	1.82e-09	2.8%
RnGen_100_1000_10	8259 / 10	2.87e-09	3.0e-09	4.5%
RnGen_6000_1000_10	13418 / 10	1.64e-09	1.68e-09	2.4%
RnGen_5000_500_10	15807 / 10	1.88e-09	1.88e-09	0%
RnGen_2000_1000_10	19490 / 10	2.32e-09	2.32e-09	0%
RnGen_1000_2000_10	24418 / 10	2.77e-09	2.79e-09	2.6%
RnGen_2800_3000_10	33961 / 10	2.68e-09	2.72e-09	1.5%
RnGen_5000_3000_10	37850 / 10	2.44e-09	2.48e-09	1.6%

Table.3 Delay Bound of STA & New Algorithm

4.3b STA & Random Simulation

Table.4 is random simulation experiments. In these experiments, delay bound is determined by random input assignments. In order to move up delay bound, random simulation needs to perform DTA more times. After time expense, random simulation still can't find the critical delay bound.

Name	Gate/Level	STA (s)	Iterations	RS (s)	Diff (%)
C17	6 / 3	6.91e-10	11968	3.46e-10	-49.9%
C001	14 / 4	9.29e-10	11595	8.02e-10	-13.7%
C002	17 / 5	1.08e-09	12738	8.92e-10	-17.4%
RnGen_3_3_3	8 / 3	7.51e-09	10072	4.81e-10	-93.6%
Rn_9_3_11	58 / 11	1.45e-09	14950	1.19e-09	-17.9%
RnGen_15_4_10	63 / 10	1.16e-09	18135	9.65e-10	-16.8%
Rn_20_10_15	133 / 15	2.28e-09	16626	1.40e-09	-38.6%
RnGen_1000_200_10	3662 / 10	1.67e-09	1085	1.25e-09	-25.1%
RnGen_4500_300_10	3272 / 10	1.29e-10	1956	1.25e-09	-3.1%
RnGen_2000_200_10	5801 / 10	1.77e-09	1221	1.62e-09	-8.5%
RnGen_100_1000_10	8259 / 10	2.87e-09	1065	2.0e-09	-30.3%
RnGen_6000_1000_10	13418 / 10	1.64e-09	1042	1.3e-09	-20.7%
RnGen_5000_500_10	15807 / 10	1.88e-09	2302	1.79e-09	-4.79%
RnGen_2000_1000_10	19490 / 10	2.32e-09	1005	2.16e-09	-6.9%
RnGen_1000_2000_10	24418 / 10	2.77e-09	1738	2.37e-09	-14.4%
RnGen_2800_3000_10	33961 / 10	2.68e-09	2057	2.21e-09	-17.5%
RnGen_5000_3000_10	37850 / 10	2.44e-09	1520	2.28e-09	-6.6%

Table.4 Delay Bound of STA & Random Simulation

Fig.11 is the delay bound histogram of three algorithms. FSTA means our new timing analysis algorithm (formal static timing analysis), and we can see that delay bound magnitude is "FSAT > STA > random simulation". These results meet our purpose of new algorithm.

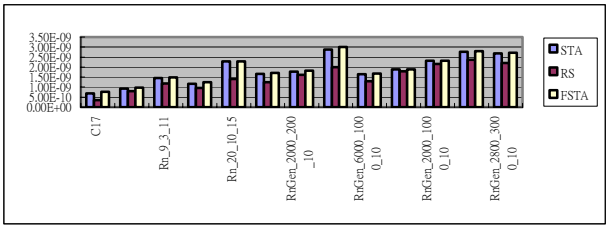


Fig.11 Histogram of Three Algorithms

4.4c Timing Analysis with All Solutions

Previous section we have explained our binary decision tree for side input pin Boolean constraints. Through our experiments (Fig.12 & Table.5), we know that the first solution is always with the most critical delay bound, corresponding to other solutions.

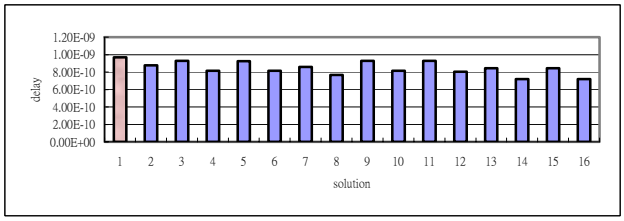


Fig.12 C001.v All-Solution Delay Bounds

There is an observation. If we run our timing analysis algorithm with all solutions, we will cost much time than the one with one solution. This is because the SAT solver would handle the most efforts during timing analysis. It doesn't make sense for our purpose and there are still some cases which max delay bound is not the first determined solution. In this way we must try to modify our algorithm, to explore delay bound and perform SAT solver with reasonable times.

Name	Max Arrival Time (s)	Solution Number	Diff (%)	Run One (s)	Run All (s)
C17	7.66e-10	1	---	0.00	0.00
C001	9.69e-10	1	---	0.01	0.02
C002	1.07e-09			0.02	0.03
RnGen_3_3_3	---	---	---	---	---
Rn_9_3_11	1.49e-09	1	---	0.02	0.06
RnGen_15_4_10	1.24e-09	1	---	0.01	0.02
Rn_20_10_15	2.29e-09	1	---	0.09	0.48
RnGen_1000_200_10	1.71e-09	1	---	1.25	29.66
RnGen_4500_300_10	1.51e-09	5	10.2%	2.6	56.28
RnGen_2000_200_10	1.82e-09	1	---	2.06	178.53
RnGen_100_1000_10	3.01e-09	1	---	13	52.42
RnGen_6000_1000_10	1.68e-09	1	---	5.98	239.23
RnGen_5000_500_10	1.88e-09	1	---	8.02	532.14
RnGen_2000_1000_10	2.32e-09	1	---	9.34	114.35
RnGen_1000_2000_10	2.79e-09	1	---	18.8	242.56
RnGen_2800_3000_10	2.72e-09	1	---	21.52	1516.94
RnGen_5000_3000_10	2.52e-09	5	1.6%	19.42	2870

Table.5 Timing Analysis with All Solutions

4.4d Critical Path Refinement

Table.6 is the critical path refinement of test cases. This table represents that our new algorithm can find out false path problem and determine most critical path.

Name	Gate/Level	False Path	Change Critical Path
C17	6 / 3	No	No
C001	14 / 4	No	Yes
C002	17 / 5	No	Yes
RnGen_3_3_3	8 / 3	Yes	---
Rn_9_3_11	58 / 11	No	No
RnGen_15_4_10	63 / 10	No	No
Rn_20_10_15	133 / 15	No	No
RnGen_1000_200_10	3662 / 10	No	No
RnGen_4500_300_10	3272 / 10	No	No
RnGen_2000_200_10	5801 / 10	No	No
RnGen_100_1000_10	8259 / 10	No	No
RnGen_6000_1000_10	13418 / 10	No	No
RnGen_5000_500_10	15807 / 10	No	No
RnGen_2000_1000_10	19490 / 10	No	No
RnGen_1000_2000_10	24418 / 10	No	No
RnGen_2800_3000_10	33961 / 10	No	No
RnGen_5000_3000_10	37850 / 10	No	No

Table.6 Critical Path Refinement with Test Cases

4.4e Found Critical Input Constraints

Here we recorded input constraints with maximum delay bound for all test cases. As Table.7 shows, these input switches would be more complex than STA, and actually STA can't determine these input constraints.

Name	PI Zero	PI One	PI Rise	PI Fall
C17	2	1	1	1
C001	0	1	2	5
C002	0	5	2	3
RnGen_3_3_3	---	---	---	---
Rn_9_3_11	3	5	0	1
RnGen_15_4_10	10	3	1	1
Rn_20_10_15	9	6	4	1
RnGen_1000_200_10	991	3	1	5
RnGen_4500_300_10	4493	4	2	1
RnGen_2000_200_10	1984	8	3	5
RnGen_100_1000_10	71	9	4	16
RnGen_6000_1000_10	5990	1	4	5
RnGen_5000_500_10	4989	4	0	7
RnGen_2000_1000_10	1972	21	2	5
RnGen_1000_2000_10	976	14	3	7
RnGen_2800_3000_10	2774	20	5	1
RnGen_5000_3000_10	4982	8	5	5

Table.7 Prime Inputs Constraints

4.4f Run Time Comparison

Table.8 is run time comparison of three algorithms. It is quite obvious that random simulation costs the most time, and our method is nearly as fast as STA.

Name	Gate/Level	STA Time (s)	NTA Time (s)	RS Time (s)
C17	6 / 3	0.00	0.00	2.22
C001	14 / 4	0.01	0.01	5.41
C002	17 / 5	0.02	0.02	2.21
RnGen_3_3_3	8 / 3	0.00	---	3.4
Rn_9_3_11	58 / 11	0.02	0.02	10.74
RnGen_15_4_10	63 / 10	0.01	0.01	12.33
Rn_20_10_15	133 / 15	0.08	0.09	34.86
RnGen_1000_200_10	3662 / 10	1.03	1.25	44.05
RnGen_4500_300_10	3272 / 10	0.91	2.6	62.42
RnGen_2000_200_10	5801 / 10	1.41	2.06	59.94
RnGen_100_1000_10	8259 / 10	12.85	13	29.81
RnGen_6000_1000_10	13418 / 10	2.31	5.98	58.91
RnGen_5000_500_10	15807 / 10	3.81	8.02	106.26
RnGen_2000_1000_10	19490 / 10	7.63	9.34	237.27
RnGen_1000_2000_10	24418 / 10	17.91	18.8	107.84
RnGen_2800_3000_10	33961 / 10	18.67	21.52	204.26
RnGen_5000_3000_10	37850 / 10	13.0	19.42	159.82

Table.8 Run Time Comparison

Fig.12 shows the expected tendency of STA, random simulation, and our new algorithm. We can see that STA has only constant delay bound. Delay bound of random simulation and our new algorithm can be improved, but simulation would cost much time. The value of our algorithm are not only improving delay bound in short time but also finding out critical input patterns for post-simulation.

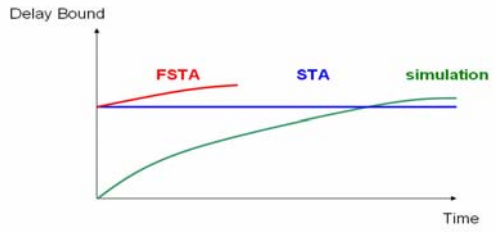


Fig.12 Tendency of Three Algorithms

五、Conclusion and Discussion

In this year, we implemented our proposed timing analysis algorithm. Current framework can parse test cases with net list format which is directly transformed from verilog and our method supports 2-input gates (and/or/nor/nand/xor) now. This method guarantees accurate delay bound in this situation, solves false path problem, and finds out critical test patterns for post-simulation. These contributions are valuable because current timing analysis tool don't provide these functions.

We completed the prototype of our timing engine. It provides user to perform STA, false path check, and perform our timing analysis algorithm in order to find out accurate delay bound with critical input constraints.

In the future we plan to do advance research for proposed algorithm. We want to research delay behavior with logic gate which fanin number is more than two. To achieve this, current look-up-table methodology would not work out because the table size would grow exponentially and take too much effort to take care. Thus, a generalized form of library might be required. If we have patience and make effort to model this problem, hopefully it could be solved. Furthermore a new format of technology file can be established at the same time.

Our work will also contribute to the publications in the coming year. We will target on the major EDA conference such as International Conference on Design Automation (ICCAD, deadline in mid-April) and Design Automation Conference (DAC, deadline in late November). At the end, the work will be summarized and submitted to selected journals.

六、 參考文獻

1. D. Lee, V. Zolotov, D. Blaauw, "Static Timing Analysis using Backward Signal Propagation", DAC 2004.
2. J. Cong, L. He, K. Y. Khoo, C. K. Koh, and Z. Pan, "Interconnect Design for Deep Submicron ICs", Proc. IEEE Int'l Conf. on Computer-Aided Design, San Jose, California, pp.478-485, November 1997.
3. 陳麒旭, "靜態時序分析 (Static Timing Analysis) 基礎及應用", http://www.chip123.com/article/Static Timing Analysis01.htm
4. MiniSAT page, http://www.cs.chalmers.se/Cs/Research/FormalMethods/MiniSat/
5. Eun Sei Park, M.Ray Mercer, "An Efficient Delay Test Generation System for Combinational Logic Circuits", Annual ACM IEEE Design Automation Conference
6. Perl 入門與實作. Http://linux.tnc.edu.tw/techdoc/perl_intro/
7. HSPICE simulation, http://www.ee.vt.edu/~ha/cadtools/hspice/hspice.html
8. MATLAB programming, http://www.cs.nthu.edu.tw/~jang/mlbook/
9. W.B. Jone, W.S. Yeh, C. W. Yeh*, and S.R. Das**, "An Adaptive Path Selection Method for Delay Testing", Dept. of CS&IE, Natinal Chung-Cheng U., Taiwan, R.O.C.*, Dept. of EE, U of Ottawa, Ottawa, Ontario, K1N 6N5, Canada**.
10. Luis Guerra e Silva, Joao Marques-Silva, L. Miguel Silveira, Karem A. Sakallah, "Satisfiability models and algorithms for circuit delay computation", ACM Transactions on Design Automation of Elecronic Systems (TODAES) 2002.
11. Seiji Kajihara, Masayasu Fukunaga, Xiaoqing Wen, Toshiyuki Maeda, Shuju Hamada, Yasuo Sato, "Path delay test compaction with process variation tolerance", Annual ACM IEEE Design Automation Conference, 2005
12. 黃俊輔, "Quick RTL Synthesis for Design Analysis and Verification", 國立台灣大學電機工程學研究所碩士論文
13. 李晨豪, "Test Pattern Generation for Maximum Circuit Delay under Simultaneous Input Transition Model", 國立台灣大學電子工程學研究所碩士論文

出席國際學術會議心得報告

計畫編號	NSC 95-2221-E-002-396
計畫名稱	FSTA：正規靜態時序分析技術(II)
出國人員姓名 服務機關及職稱	周俊男
會議時間地點	美國聖地亞哥 U.S. San Diego
會議名稱	設計自動化會議 Design Automation Conference
發表論文題目	

一、參加會議經過

今年的設計自動化會議在美國聖地亞哥會議中心(San Diego Convention Center)舉行。我們先搭飛機前往洛杉磯，接著在當地租車後再開車到聖地亞哥參加會議。設計自動化會議不愧是電子設計自動化領域中最大且最重要的國際會議，與會人數相當的多且參展廠商也超過250 家，規模相當的龐大卻又不失其條理。

不同的會議(session)在不同的會場同時舉行，我參加了與我研究相關的會議，如 Formal and Semi-Formal Verification Techniques(Session 5), Statistical Techniques for Timing Analysis and Design (Session 14), TLM: Crossing Over from Buzz to Adoption (Session 25), Verification Coverage: When is Enough Enough ? (Session 41), Dynamic Verification of Processors and Processor-Based Designs (Session 48)等，除了聆聽其他學校和研究機構的研究方向和研究成果之外，我也去展示會場參加 Gary Smith 主講的演講並參觀各個參展廠商，聆聽他們的產品介紹和實際的 Demo，了解工業界的需求與趨勢走向。

在這會議的期間當中，認識了幾位在這領域的知名前輩，像是 Laung-Terng Wang (the President and CEO of SYNTEST) 和 Yu-Chin Hsu (Vice President of Research & Development of NOVAS)，能有機會與他們討論並給予我建議和指導，真是在下的榮幸，另外也認識許多從台灣來的朋友，像是蔡永平總經理(茂積股份有限公司)、鄭裔文工程部經理(茂積股份有限公司)、陳紀綱技術副理(工研院系統晶片科技中心)、施泳千專案經理(亞睿資訊股份有限公司)等。

二、與會心得

在會議的這幾天當中，無論是跟本計畫有關的時序分析(timing analysis)方面，抑或是跟本實驗室有關的驗證(verification)與電子系統階層(ESL)驗證等方面，受益良多，尤其是對於目前大家針對這些方面的趨勢走向更為清楚。

像是關於時序分析而言，今年在會議中發表的相關論文，大致上都是針對統計靜態時間分析(Statistical Static Timing Analysis)為主軸，簡寫為 SSTA，像是在 Session 14 當中的”Comparative Analysis of Conventional and Statistical Design Techniques“就是在最佳化問題上去比較 SSTA 和其他方法的差異，而在相同 Session 14 中的其他論文，如”Fast Second-Order Statistical Static Timing Analysis Using Parameter Dimension Reduction”、”Non-Linear Statistical

Static Timing Analysis for No-Gaussian Variation Source”和”Beyond Low-Order Statistical Response Surfaces: Latent Variable Regressions for Efficient, Highly Nonlinear Fitting”，都是針對 SSTA 的 scalability 提出他們自己的見解並提出方法加以改善，另外在 Session 45 中的”A Framework for Accounting for Process Model Uncertainty in Statistical Static Timing Analysis”也有類似的概念在其中。

另外關於展示場的參觀部分，與我們實驗室研究相關的公司產品，其實不難發現有一大部分的公司都著重在如何讓模擬速度加快，而另外一大部分則強調電子系統階層的開發、驗證與整合。

除了上述兩點之外，我感受最深刻的是，想要站上國際舞台，語言能力一定要夠好，尤其是英文的聽和說，在整個會議過程當中，無論是聆聽別人的言論，抑或是想要詢問問題等，都有表達溝通上的阻礙，英文實在是有待加強。

就整體而言，參加這次盛大的國際會議，不僅是開拓自己學術的視野、增廣見聞外，更讓我了解整個電子自動化領域的生態和文化，是個難忘與值得的經歷。