國立中興大學資訊科學與工程學系

碩士學位論文

循 序 硬 體 木 馬 偵 測 方 法 之 研 究

A Study on Sequential Hardware Trojans Detection

指導教授：王行健 Sying-Jyan Wang
研 究 生：張源亨 Yuan-Heng Chang

中華民國一百零八年八月

國立中興大學 資訊科學與工程學系

碩士學位論文

題目：循序硬體木馬偵測方法之研究

A Study on Sequential Hardware Trojans Detection

姓名：張源亨 學號：7106056011

經 口 試 通 過 特 此 證 明

論文指導教授 王行健

論文考試委員 王行健

李承毅

鄭經華

中華民國 108 年 7 月 22 日

致謝

感謝指導教授王行健老師兩年來的苦心栽培，讓我學習到更多的專業知識與研究精神，經過老師的鞭策後使我對 VLSI 測試領域更加的專精，在每個禮拜報論文的過程中提高了找出重點和歸納以及口語表達的能力，感謝老師的教導讓我受益良多。

感謝應延、冠伶、國揚、彥霖、建德學長姊，在我一年級剛進入實驗室時給予我許多幫助，例如在課業上及研究中遇到瓶頸的時候能夠不吝嗇地給予建議，也感謝同學育聖、慈恆在求學路上一起成長互相勉勵，還有三位學弟衍彰、奕緯以及承軒，雖然你們平日都很忙，但也很感謝你們在生活上的幫忙。

最後感謝的我家人在求學路上無條件的支持我，也讓我沒有經濟上的壓力，能夠專心在研究上，最後祝福所有曾經幫助過我的貴人們，希望你們都能心想事成。

摘要

近年來無晶圓廠半導體產業及政府機構已經開始擔憂及研究在積體電路供應鏈中可能被插入硬體木馬來竄改電路，導致電路功能異常或重要資訊被竊取。

硬體木馬的偵測大多數是藉由觀察到電路有錯誤的輸出信號或透過側頻道分析(Side channel)觀察是否有異常的側頻道信號產生。因此，啟動木馬電路所花費的時間對於驗證者是一個非常重要的問題。

在此篇論文中，我們提出了一個自動化的方法加速硬體木馬的偵測。本方法針對轉換機率較低的信號線，找出適合的位置插入邏輯閘及控制信號，以提升電路的轉換機率，使電路中的每一條信號線轉換機率高於臨界值，以利於組合木馬及循序木馬的檢測。實驗結果顯示此篇提出的方法面積開銷非常小且能增加循序木馬電路的活躍性，減少啟動木馬所需的時間。

Abstract

In recent years, manufacturers and governments have studied the possibility of Hardware Trojan insertions to tamper original circuit among the supply chain of integrated circuits. Hardware Trojans lead to incorrect functionality of circuits and leakage of important information.

Most of the Hardware Trojan detections are based on observation of unexpected output signals for circuit and side channel analysis which detects the abnormal signals of side channels. Thus, the time for an activation of a Hardware Trojan is a significant issue for Hardware Trojan detectors.

In this thesis, we propose an automatic method to accelerate the Hardware Trojan detection. In the proposed method, logic gates are inserted into proper locations of a circuit with corresponding control signals. Therefore, the probabilities of signal transitions are increased so that they are higher than given threshold, which improves the detectability of both combinational and sequential Hardware Trojans. Experimental results show that the proposed method increases the activity of Hardware Trojans and reduces the required time for Hardware Trojan activations while the hardware overhead is low.

目次

圖目次

表目次

第一章　簡介

1.1 研究動機與目標

IC 產業的快速發展是全球化導致的結果，由於全球競爭激烈，一些 VLSI 設計和製造可能不得不轉移到各個國家以降低成本，有目的的攻擊者可能利用這種現象來擾亂整個供應鏈[1]。硬體安全面臨的主要威脅包括偽造(counterfeiting)、逆向工程(reverse engineering)及 IC 篡改。其中攻擊者將惡意的硬體嵌入到原始電路中進行篡改，這種電路被稱為硬體木馬(Hardware Trojan)[2]-[6]。首先，IC 產品的設計和製造通常涉及許多層面，包括設計公司、EDA 供應商、IP 供應商、代工廠、封裝測試公司等，在所有階段都有機會修改產品的一部分，因此，對電子系統的安全性產生了嚴重的隱憂。

硬件木馬檢測是一個極具挑戰性的問題，傳統的結構和功能測試無法有效解決它。木馬電路具有隱蔽性，只會在極少數情況下觸發。通常木馬的設計使它們大部分時間處於沉默狀態，並且相對於原始電路而言可能尺寸非常小，這些因素表明了它們很可能連接到具有可控制性或可觀察性較低的線[7]-[9]，也就是木馬電路的輸入會接到原始電路中轉換機率低的線，以減少其對電路側頻道(side channel)的影響。為了提升硬體的安全性，文獻中也提出了許多偵測木馬的方法[10]-[20]。透過側頻道分析(Side channel)去觀察是否有不正常的側頻道信

號產生；或者當木馬被觸發後，可以在輸出觀察其錯誤。

本篇論文目的為減少硬體木馬可能被植入的地方。首先要找出可疑的線，將其定為轉換機率低的目標點，再插入相對應的控制點來修改電路，設法提高此線的轉換機率。最後，必須加入控制輸入連接到所有插入的控制點。我們可以利用額外的控制信號來控制電路中所有修改後的部分。

1.2 內容大綱

本論文共分為五個章節，第一章簡介硬體木馬的背景及本篇論文的目標。第二章介紹知識背景以及簡述實作的方法。第三章介紹本篇尋找目標點及插入控制點的實作方式。第四章為實驗結果的說明與分析。最後則是本篇論文的總結。

第二章　背景知識與簡述實作方法

2.1 硬體木馬的種類與架構

硬體木馬(Hardware Trojan)通常是由觸發邏輯(trigger)和 payload 組成。觸發邏輯的輸入來自原始電路，payload 的輸入來自觸發邏輯的輸出及原始電路的信號。觸發邏輯用來表示在特定的條件下會啟動木馬，而 payload 用來表示要執行的惡意功能，如圖 1 所示:

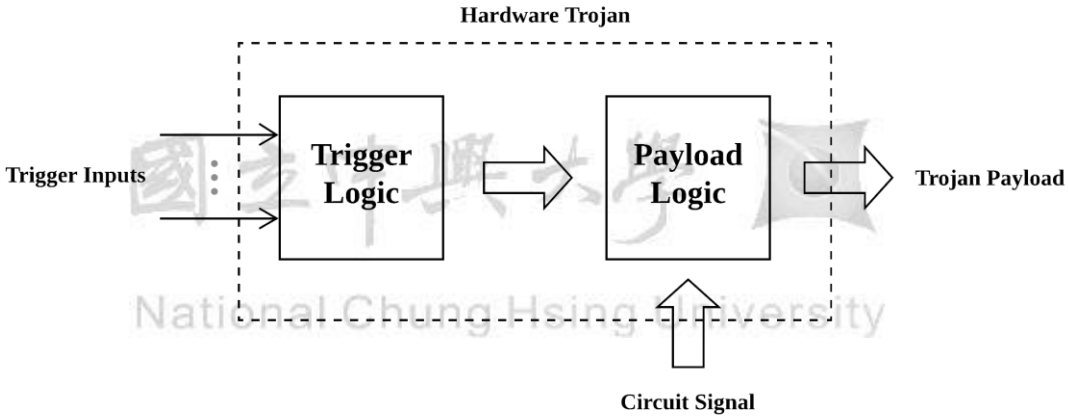

圖 1：典型的硬體木馬

另外根據觸發邏輯的設計，又可將木馬電路分為組合木馬與循序木馬，我們會在下一段簡單介紹。

2.1.1 組合木馬

組合木馬不會包含任何的記憶元件(memory components)，如圖 2 所示，邏輯閘 A、B、C、D 為觸發邏輯，邏輯閘 E 為 payload，另

外木馬電路上方的虛線表示原始電路中的連接，當 Trigger 1 至 Trigger 4 依序為(1011)時觸發條件會被滿足，木馬會被啟動，導致不如預期的結果，反之 Trigger 1 至 Trigger 4 不為(1011)，則硬體木馬的部分是完全不影響電路的正常情況。

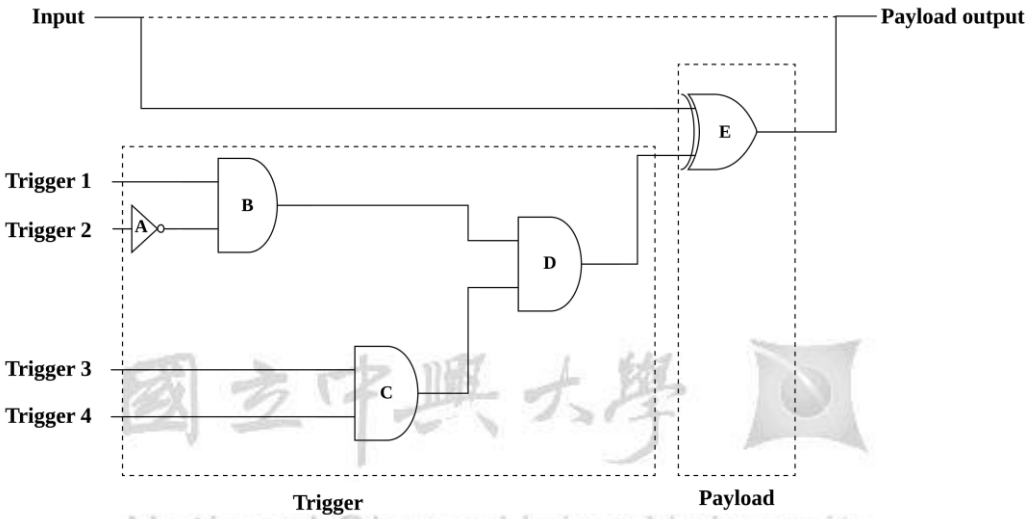

圖 2：組合木馬例子

2.1.2 循序木馬

如圖 3 所示，當 Trigger 1 至 Trigger 4 依序為(1111)時可以觸發計數器一次，必須成功觸發 2^k 次才能使 payload 被啟動。

由上例子可見，攻擊者所安裝的額外硬體木馬，一般情況下不會影響電路的正常運作且具有非常好的隱密性，若能被當成木馬的觸發條件，肯定要是 0 或 1 出現的機率趨近於 0 的訊號線。

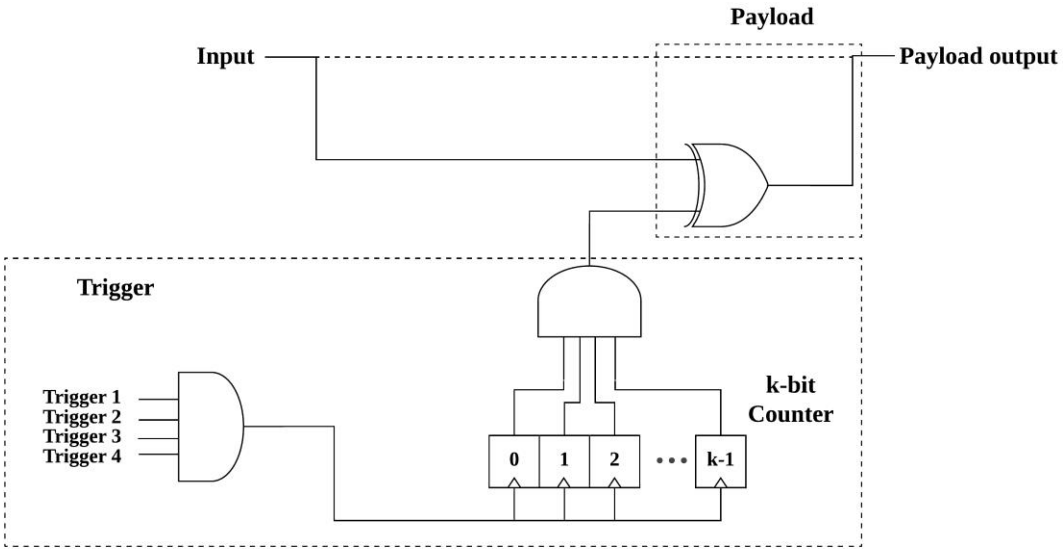

圖 3：循序木馬例子

2.2 相關研究

根據[21]的分析，木馬必須放置在隱蔽性高且可控性高的位置，此篇考慮了三種方向。第一個為木馬必須放置在極少切換的線上，此線可能常駐為 0 或 1；第二點是插入的木馬可利用老化的技術來避開漏功率的檢測；最後木馬電路也要盡量放置在重新收斂路徑(reconvergent paths)中延遲較低的路徑上，避免產生額外的延遲。

在[22]中作者考量電路中各種雜訊，提出一個方法來產生正確 IC 的功率指紋(power fingerprint)，接著應用隨機測試向量於待驗證 IC (IUA, IC-Under-Authentication)，在正確 IC 和 IUA 的功率分佈之間產生可測量的差異來偵測木馬。而第[23]篇提出的方法是基於分析晶片電源端(power ports)的 I_{DDT} 電流，在實際測量之前對每個 IUA 執行校準過程以減輕製成變異影響。

[24]所提出偵測木馬的方法是使用 path delay，使用工具去得到電路中每一個邏輯閘的延遲資訊。若安裝額外的硬體木馬就一定會有額外的延遲。[25]分析了啟動電路中功能性木馬所需的時間，並透過轉換機率低的線新增一個 dummy scan Flip-Flop 和一個邏輯閘縮減觸發木馬的時間，同時增加電路的轉換機率。

2.3 控制輸入(control input)與控制點(control point)

我們首先要找出轉換機率低於臨界值的線，將它們設為可疑的目標點後，再利用本篇提出的方法縮減目標點。作法是將目標點的輸入來源插入對應的控制點(control point)，這些控制點皆會被一條主要輸入(primary input)線控制，這條線稱為控制輸入，另外控制點又可分為 AND 控制點 及 OR 控制點，如圖 4:

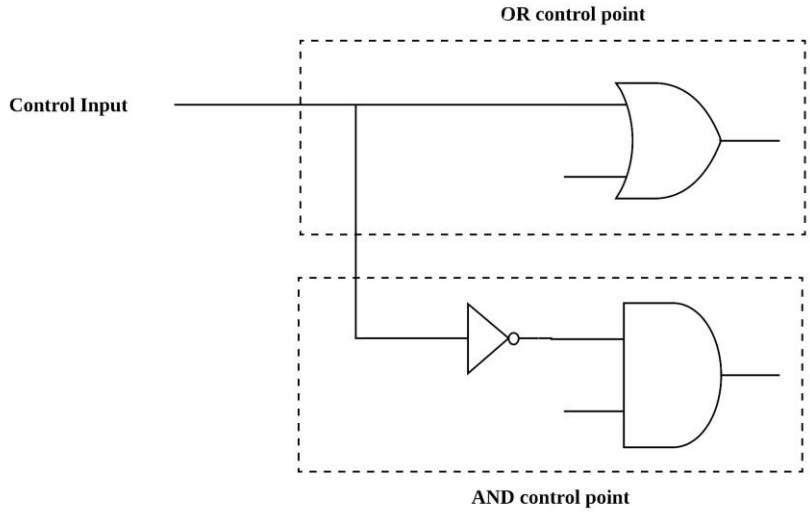

圖 4：OR 控制點及 AND 控制點

在正常模式下，控制輸入固定輸入為 0，因此控制點會被忽略，不影響原本電路功能。

在測試模式下，我們隨機指定控制輸入為 0 或 1，OR 控制點能提升邏輯值 1 出現的機率，而 AND 控制點能提升邏輯值 0 出現的機率，我們能透過此方式來縮減目標點。另外插入控制點會增加性能和面積開銷，所以必須使插入的控制點數量最小化[26]-[29]。

2.3.1 插入 AND 控制點

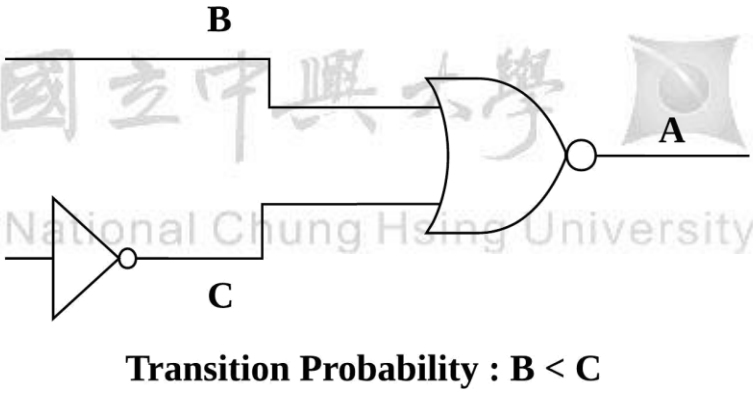

圖 5：容易產生 0 的目標點

以圖 5 為例，假設 A 為電路中的一個目標點。而 A 為目標點的原因是因為它經常產生 0，這意味著 A 的兩個輸入來源有一個較常為 1，因此我們可以在轉換機率較低的輸入來源插入控制點，使 A 產生 0 的機率降低。如果我們直接在目標點處插入控制點，控制點之後的信號線轉換機率會提升，但控制點之前信號線的轉換機率依

舊不會改變。

圖 6 是圖 5 插入控制點後的電路。AND 控制點的兩個輸入來源分別為 B 及控制輸入。在我們插入 AND 控制點後，A 產生 1 的信號提高了，因此 A 的轉換機率也會提升，若 A 的轉換機率高於臨界值，A 就不再屬於目標點了。

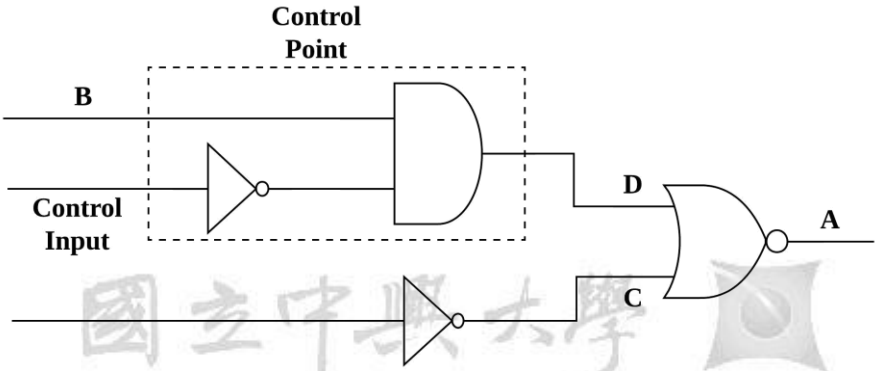

圖 6：插入 AND 控制點

2.3.2 插入 OR 控制點

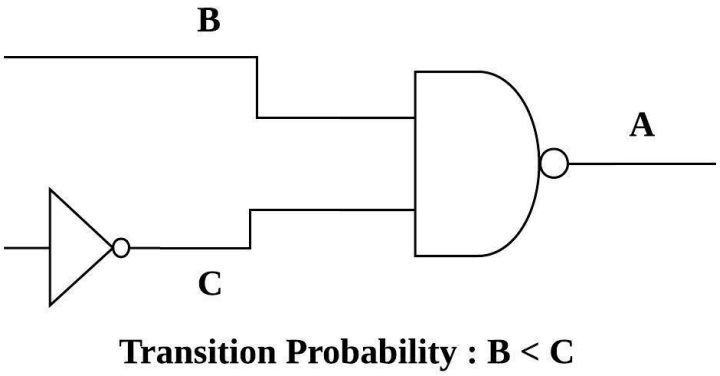

圖 7：容易產生 1 的目標點

以圖 7 為例，假設 A 為電路中的一個目標點。而 A 為目標點的

原因是因為它經常產生 1，這意味著 A 的兩個輸入來源有一個較常為 0，因此我們可以在轉換機率較低的輸入來源插入控制點，使 A 產生 1 的機率降低。

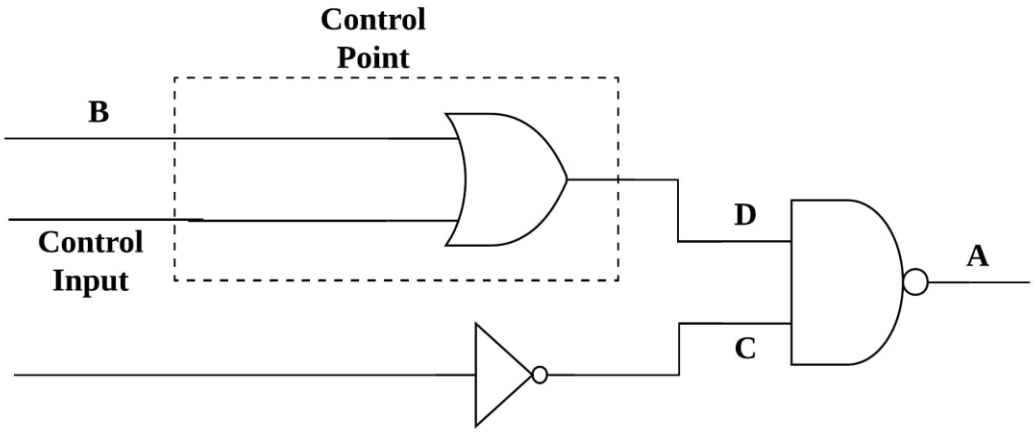

圖 8：插入 OR 控制點

圖 8 是圖 7 插入控制點後的電路。OR 控制點的兩個輸入來源分別為 B 及控制輸入。在我們插入 OR 控制點後，A 產生 0 的信號提高了，因此 A 的轉換機率也會提升，若 A 的轉換機率高於臨界值，A 就不再屬於目標點了。

最後，目標點的輸入來源可能會有相依關係，沒有辦法單靠一個控制輸入來提升目標點的轉換機率，因此可能會使用到一個以上的控制輸入，我們會在下一章說明。

第三章　實作方法與實驗流程

由於木馬電路通常是透過轉換機率非常低的信號線來觸發，為了更容易地偵測木馬電路，我們將電路中轉換機率低於臨界值的信號線稱為目標點，並插入對應的控制點提升目標點的轉換機率，使木馬電路更容易被啟動。

3.1 透過隨機邏輯模擬得到所有目標點

首先我們定義轉換機率(transition probability)為每條線 0→1 的機率加上 1→0 的機率。每次邏輯模擬時主要輸入(primary input)0 與 1 的機率約為 0.5。隨機執行測試電路一定次數，電路中每條線的轉換機率會趨近一固定值。

隨機模擬結束後，我們將設定一個臨界值(threshold)，若信號線的轉換機率低於此臨界值則被稱為目標點。本論文中，轉換機率的臨界值 σ 皆設為 0.03。臨界值越高木馬電路越容易被啟動，但需要處理的目標點會變多，而本篇的目標就是用控制點來減少目標點，使木馬電路更容易被觸發，接下來的段落中我們將介紹目標點之間的關係。

3.2 目標點之相依關係

因為我們會根據目標點來修改電路，目標點越少則插入的控制

點就越少，面積開銷也會越小。我們也發現如果兩個目標點是相依關係，則只需改變靠近主要輸入(primary input)目標點的轉換機率，就能有效影響另一目標點的轉換機率。相依的結構又可分為完全相依及部分相依。

3.2.1 完全相依之目標點

如圖 9 所示，假設 A 和 B 都是轉換機率小於臨界值的目標點。因為只經過一個 NOT 邏輯閘，所以 A 和 B 的轉換機率相同，如果提升 A 的轉換機率則 B 也會跟著改變，所以當兩個目標點分別為 NOT 邏輯閘的輸入和輸出時，我們只保留輸入的目標點。

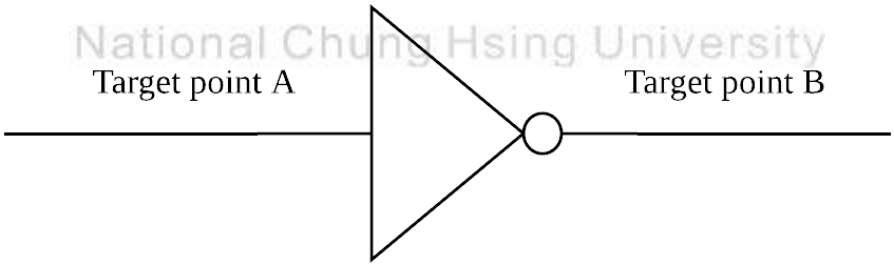

圖 9：完全相依之目標點

如果電路是一連串的 NOT 邏輯閘，信號線皆為目標點 A 到 N，根據完全相依的關係，只需要保留目標點 A 即可，如圖 10。

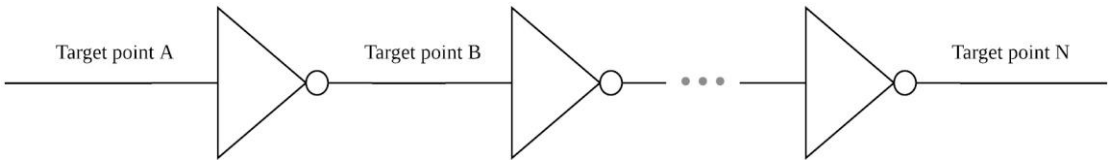

圖 10：多個完全相依之目標點

3.2.2 部分相依之目標點

當 AND、OR、NAND、NOR 邏輯閘的任一輸入為目標點且輸出也為目標點，稱為部分相依。

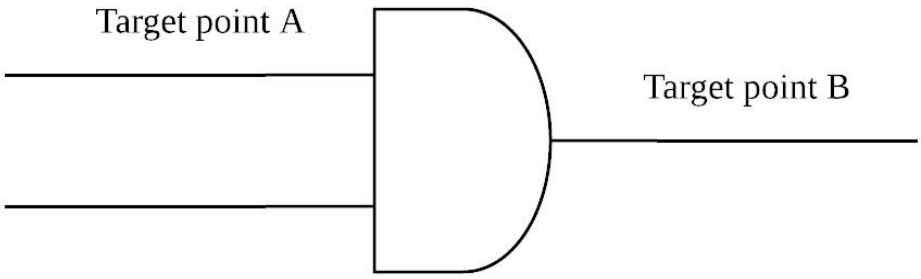

圖 11：部分相依之電路架構

以圖 11 為例，A 與 B 均為目標點。如果 A 經常為 0，將使得 B 亦經常為 0，要使 B 為邏輯 1 的時候，A 也必須為 1，因此 A 和 B 多少有相依關係。

假設目標點 A 邏輯值 0 的機率非常高，則目標點 B 邏輯值 0 的機率也會非常高，因此改善 A 的轉換機率能提升 B 的轉換機率。反之 A 邏輯值 1 的機率非常高，則 B 為目標點的可能性非常低，因此可將目標點 B 移除。

實作中我們會刪除相依情況的目標點來減少處理的複雜度，使插入的控制點能越少越好。

3.3 相同來源之目標點

若兩個以上的目標點具有相同來源，此來源很有可能是造能目標點產生的原因，因此我們會分析相同來源的目標點，只需插入一個控制點就能同時解決兩個以上的目標點，減少額外增加的面積開銷。

我們首先要考慮邏輯閘的種類，必須要是相同種類的邏輯閘才有辦法在共同來源插入控制點來提升轉換機率，如 AND 邏輯閘和 NAND 邏輯閘可以在共同來源插入控制點，OR 邏輯閘和 NOR 邏輯閘可以在共同來源插入控制點。

輸入線的轉換機率也必須考量。以圖 12 為例，如果相同來源 Net 2 的轉換機率越低，越有可能是產生目標點的原因，因此我們會在相同來源前加入控制點；反之若相同來源 Net 2 的轉換機率越高，越不可能是產生目標點的原因。在本篇中，相同來源的轉換機率和其他目標點的輸入相比只要不是最高的，皆會被插入控制點。

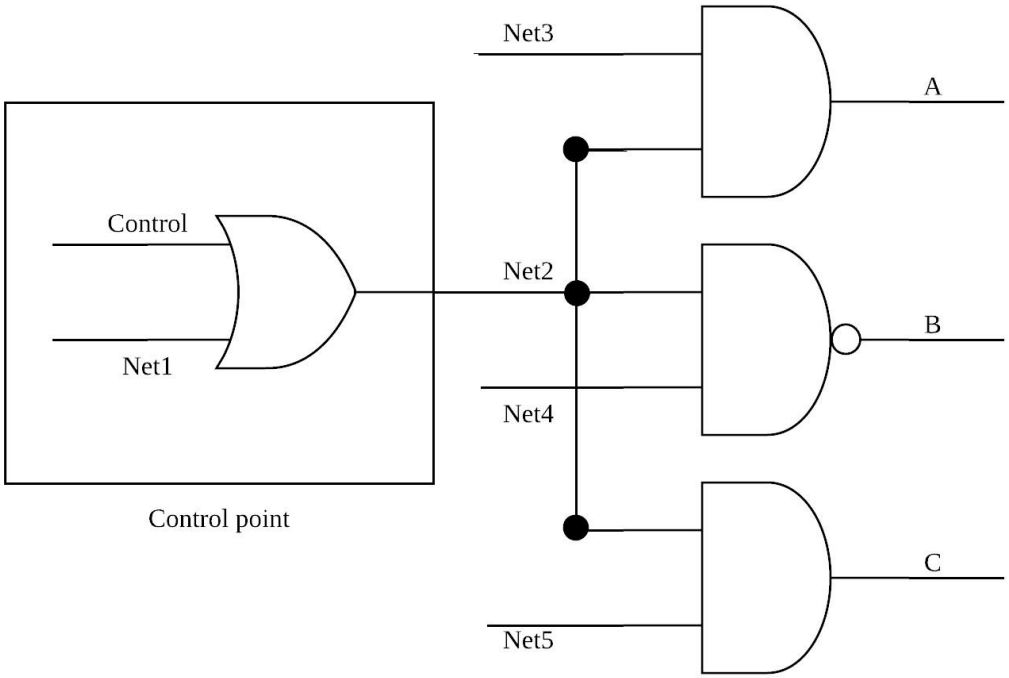

圖 12：相同來源修改後的電路架構

3.4 相依及相同來源之目標點

相依及相同來源的目標點是指目標點相依及兩目標點有相同來源，並且邏輯閘種類必須相同，如圖 13 所示:

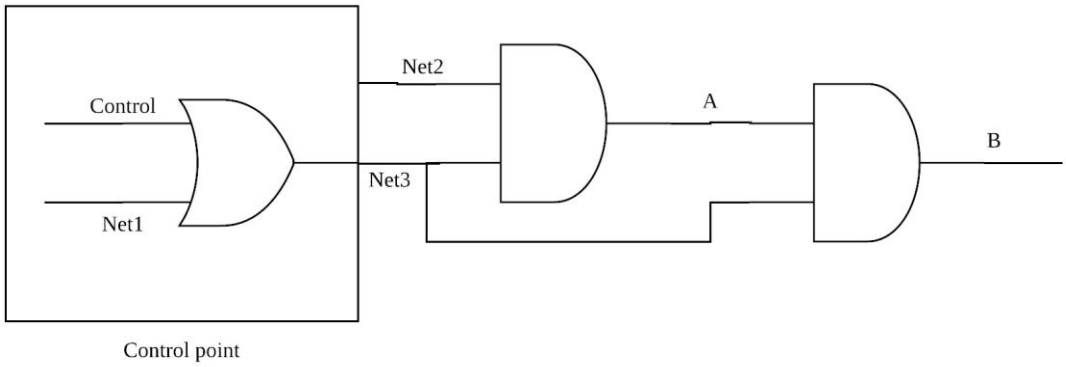

圖 13：相依及相同來源修改後的電路架構

在圖 9 中，A 和 B 為目標點。因為 A 和 B 邏輯值 1 的機率很低，因此我們只需在相同來源處(Net 2)加入一個控制點，就能同時

提升兩目標點邏輯值 1 的機率。

3.5 獨立之目標點

沒有明顯關係的目標點我們稱為獨立的目標點。由於這些目標點的關係沒有那麼明確，每個獨立的目標點會直接插入控制點來提升轉換機率，根據目標點輸入(inputs)的數量，插入的控制點數量也不同。

3.5.1 目標點輸入信號線不超過三條

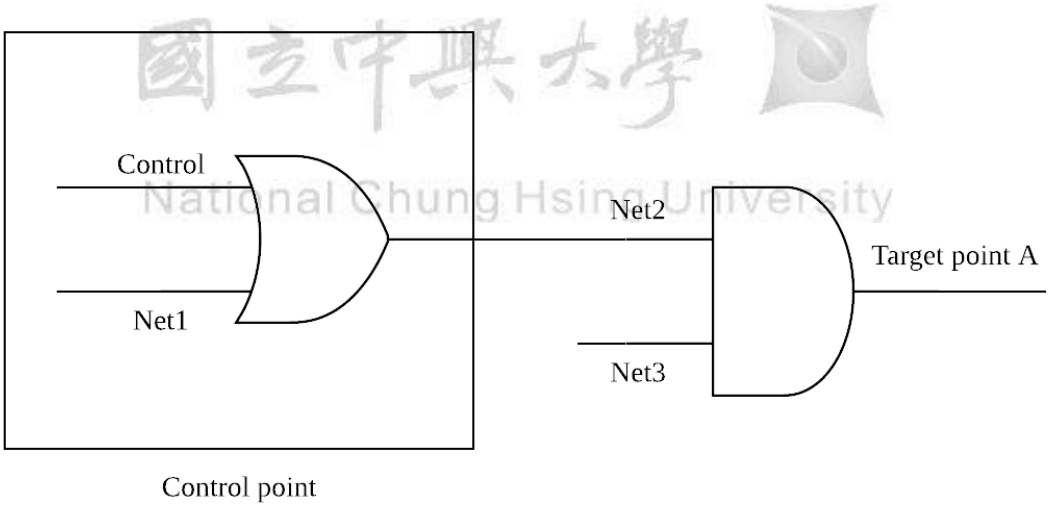

圖 14：輸入信號線不超過三條

在圖 14 中，A 為目標點且 Net 2 的轉換機率小於 Net 1，Net 2 較有可能是影響 A 的主要原因，因此直接在 Net 2 前插入一個目標點提升 A 的轉換機率。

3.5.2 目標點輸入信號線超過三條

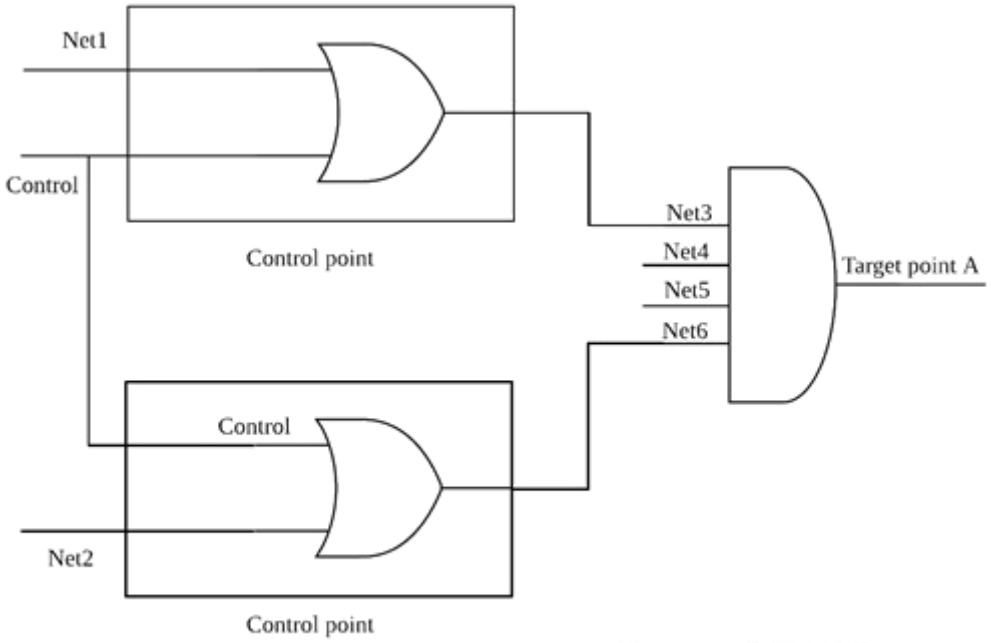

圖 15：目標點信號線超過三條

如果目標點的輸入信號線超過三條，我們會找出轉換機率最低的兩條線來插入控制點。如圖 15 所示，A 為目標點且輸入信號線的轉換機率為 Net 3 < Net 6 < Net 4 < Net 5，因此我們選 Net 3 及 Net 6 插入控制點。

3.6 剩餘之目標點

經過以上步驟後，還未能解決的目標點被稱為剩餘之目標點，為了防止硬體木馬插入在這些目標點，我們會特別針對剩餘目標點產生測試向量來觸發這些點。

3.7 硬體木馬的觸發條件

在這一段我們將討論本篇方法用於觸發組合木馬電路及循序木馬電路的機率:

3.7.1 四個觸發條件的組合木馬

四個觸發信號組成的木馬如圖 16 所示。假設沒有使用本篇方法修改電路且 Trigger 1 至 Trigger 4 為邏輯值 1 的機率都小於 5×10^{-5}，木馬觸發被觀察到的機率將小於 $(5\times10^{-5})^4$ ，因此木馬被偵測的機率非常低。

若使用本篇方法修改電路，使每條輸入信號線為 1 的機率高於 3×10^{-2}，轉換機率小於臨界值的線也會產生特定的測試向量加入邏輯模擬，因此木馬被觀察到的機率將大於 $(3\times10^{-2})^4$。

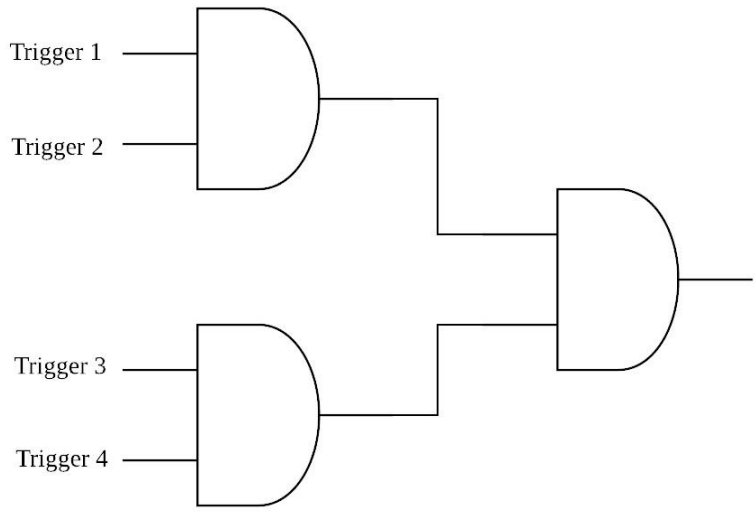

圖 16：4 個觸發條件的組合木馬

3.7.2 四個觸發條件的循序木馬

若木馬電路是屬於計數器類型且觸發條件都是相同的，如圖 17 所示:

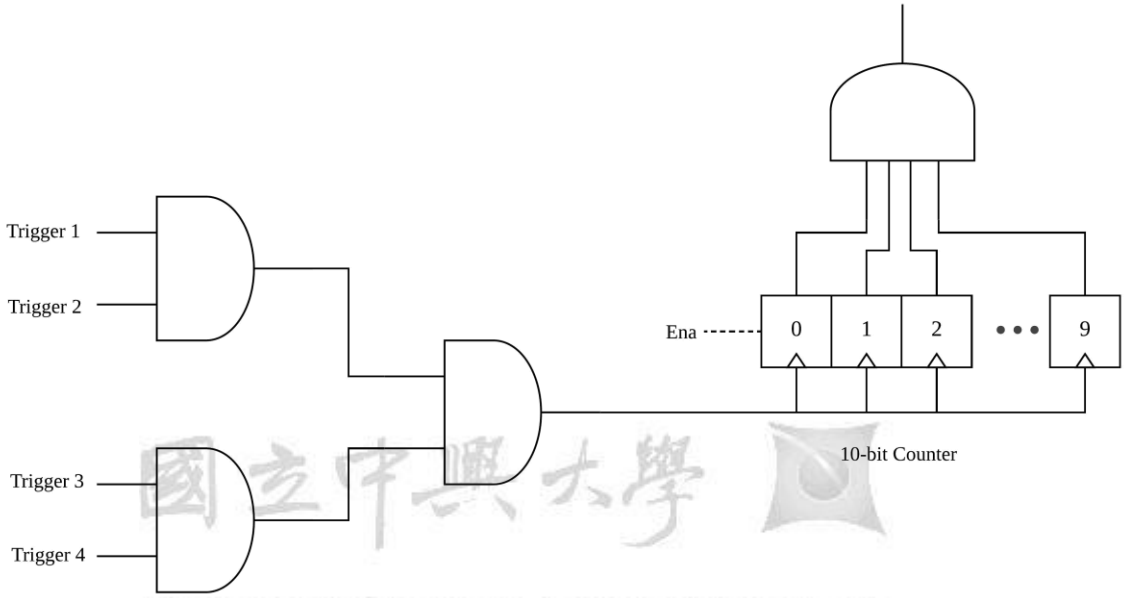

圖 17：4 個觸發條件的計數器循序木馬

若要觸發此種硬體木馬 Trigger 1 至 Trigger 4 需要為 1，必須成立 2^N 次才會使硬體木馬被成功的觸發。假設沒有使用本篇方法修改電路且觸發信號 1 至 4 為邏輯值 1 的機率都小於 5×10^{-5}，則計數器計數一次的機率將小於 $(5\times10^{-5})^4$。觸發木馬前計數器必須數 2^{10} 次，因此若沒有使用本篇方法時觸發木馬的機率將小於 $(5\times10^{-5})^4\times2^{10}$。若使用本篇方法修改電路，使每條輸入信號線為 1 的機率高於 3×10^{-2}，轉換機率小於臨界值的線也會產生特定的測試向量加入

邏輯模擬，因此木馬被觀察到的機率將大於 $(3\times10^{-2})^4 \times 2^{10}$。

3.7.3 有限狀態機型的循序木馬

有限狀態機木馬和計數器木馬的差異在於計數器木馬每一次觸發成立的條件都相同，只要完成一定次數即可，而有限狀態機木馬每一組觸發成立的條件都不一樣，像是一個有限狀態機。

一個有限狀態機木馬範例如圖 18 所示。圖中每一點代表木馬電路的一個狀態，每次木馬狀態轉移的觸發條件不相同，且是有順序性的，當前一個組合觸發條件成立之後才會移動到下一個階段，意味著此種木馬被觀察到的機率極低。假如有限狀態機設計複雜且龐大，幾乎不可能使硬體木馬被驅動，則此種木馬對原始電路產生危害的機率也微乎其微。

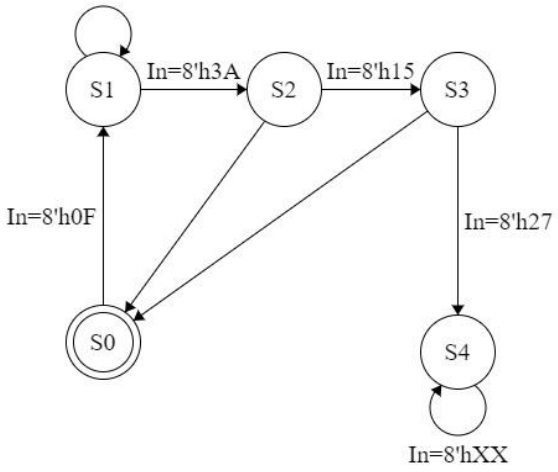

圖 18：4 個觸發條件的不規則循序木馬

3.8 實驗流程

根據圖 19 所示，首先透過邏輯模擬的方式找出所有目標點並記錄目標點總數為 Count，找出有相同來源且相依的目標點插入適當的控制點，接著刪除彼此相依的目標點，因為相同來源且相依的目標點包含於相依目標點，因此不能直接刪除相依的目標點。接著處理共同來源目標點及剩餘之目標點，最後再次模擬電路記錄新的目標點總數為 updatedCount，重複以上步驟直到目標點無法再減少為止。

另外臨界值 α 是指我們能容忍未能解決的目標點的最大數量，如果目標點大於 α，則新增一條主要輸入控制接腳重作以上步驟，如果目標點小於 α，只需要產生能觸發這些目標點的測試向量，最後將這些測試向量加入邏輯模擬的向量集中，有效觸發電路中的木馬電路。

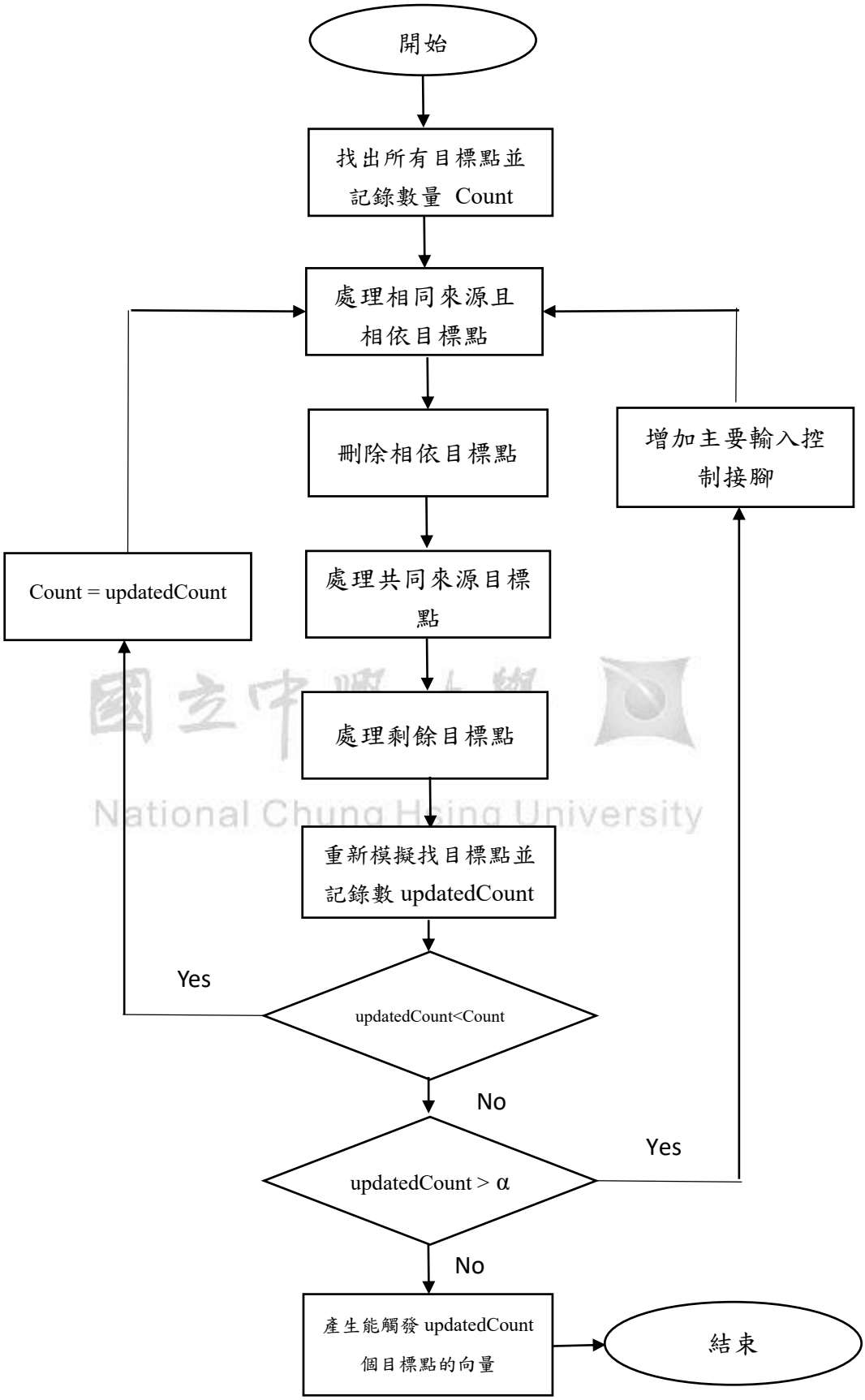

圖 19：實驗流程圖

3.9 實作方法

圖 20 是此篇方法的虛擬碼:

```
procedure: HTDetection
Input:   BenchmarkCircuit,
Output: ModifiedCircuit
1   TargetPoint ← Ø //用來儲存目標點
2   Graph = ConstructGraph(BenchmarkCircuit)//把benchmark檔轉換成圖
3   TargetPoint = Simulation(Graph, σ);//模擬電路找出目標點, σ = 0.03
4   DeletedPoint ← Ø //用來儲存相依且共同來源目標點
5   for each nodeA in TargetPoint
6     DeletedPoint.add(FindCommonSourceAndDepedent(nodeA)) //找相依且共同來源目標點
7   End
8   InsertControlPoint(DeletedPoint);//根據相依且共同來源目標點插入適當控制點
9   Update(Graph)//更新電路結構
10  TargetPoint – DeletedPoint //刪除相依且共同來源目標點
11  TargetPoint – Dependent(TargetPoint)//刪除相依目標點
12  DeletedPoint ← Ø //用來儲存共同來源目標點
13  for each nodeA in TargetPoint
14    DeletedPoint.add(FindCommonSource(nodeA)); //找共同來源目標點
15  End
16  InsertControlPoint(DeletedPoint)//根據共同來源目標點插入適當控制點
17  Update(Graph)//更新電路結構
18  TargetPoint – DeletedPoint //刪除共同來源目標點，剩餘獨立目標點
19  InsertControlPoint(TargetPoint)//  根據獨立目標點插入適當控制點
20  Update(Graph)//更新電路結構
21  NewBenchmark = GraphToBenchmark(Graph)//將電路結構轉成benchmark檔
22  Return ModifiedCircuit
23
24
```

圖 20：實作方法

第四章 實驗結果

我們使用提出的程式模擬電路找出目標點，分析所有目標點插入對應之控制點，可以明顯地消除電路中可疑的位置，成功地減少木馬可能存在的地方。並且使用 Synopsys Design Compiler 評估額外面積花費。

4.1 電路規格

我們取 ISCAS'89 [30]中六個電路來做實驗，電路規格如表 1。在接下來的實驗中，因為 ISCAS'89 是循序電路，我們假設所有暫存器皆是掃描暫存器，並且偵測木馬的行為是在測試模式下進行，因此可以把所有暫存器視為一個輸入。

表 1：ISCAS'89 電路規格

Circuit	# of gates	# of primary inputs	# of primary outputs	# of FFs
s5378	2779	35	49	179
s9234	5597	36	39	211
s13207	7951	62	152	638
s15850	9772	77	150	534
s38417	22179	28	106	1636
s38584	19253	38	304	1426

4.2 分析目標點

在此篇中我們將轉換機率低於臨界值的信號線視為目標點 (臨界值設為 $\sigma = 0.03$)，假設某條信號線的轉換機率為 10^{-4}，平均來說我

們需要模擬 10000 次才可能觸發一次轉換，如果轉換機率被提升為 0.03，模擬 33 次就可能觸發一次轉換，後面的實驗我們將轉換機率的臨界值設為 0.03。

另外我們也統計了在不同臨界值所需處理的目標點數量，如表 2，臨界值越高須要處理的目標點就越多，插入的控制點也越多，面積開銷也會增加，但電路整體的轉換機率越高。反之臨界值越小插入的控制點越少，額外面積開銷也越少，電路整體的轉換機率較低。

表 2：不同臨界值產生的目標點數量

電路	臨界值			
	0.001	0.03	0.05	0.1
s5378	47	215	285	597
s9234	207	684	801	945
s13207	604	1258	1327	1415
s15850	147	915	1040	1266
s38417	529	1157	1452	2080
s38584	917	2000	2219	3020

為了觀察電路的轉換機率分布，我們用 30000 次的隨機邏輯模擬(輸入邏輯值“1” 和 “0”機率相同)產生轉換機率。如圖 21 所示，在 s38584 電路中有 8980 條信號線轉換機率為 0.45 至 0.5，而我們的目標是轉換機率 0 至 0.03 的 2000 條信號線。

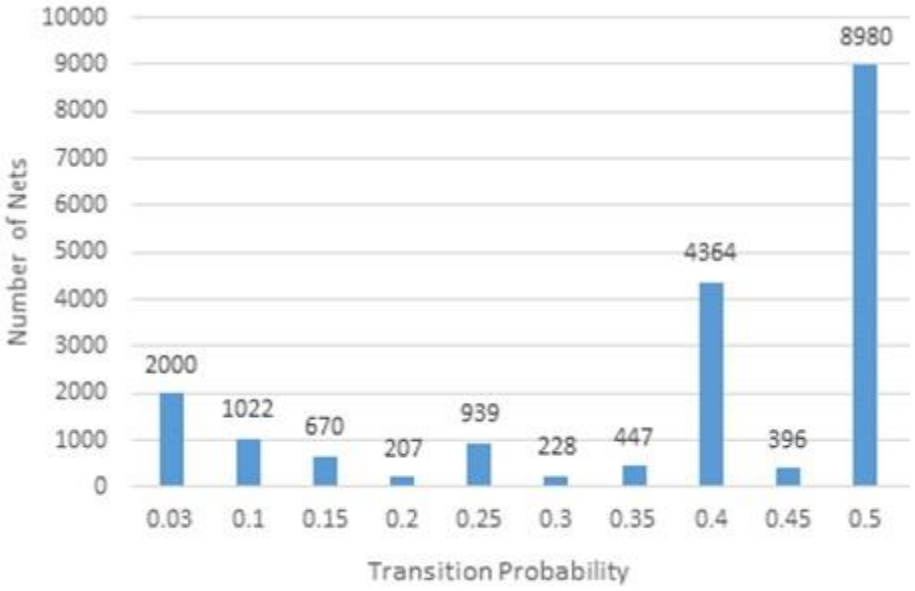

圖 21：s38584 轉換機率分布圖

4.3 實驗評估

4.3.1 插入控制點後目標點數量

找出目標點後，會根據目標點的關係插入適當的控制點，接著再次模擬電路來觀察實驗結果。表 3 為插入控制點後的實驗結果，第一行為電路名稱，第二行是插入控制點前目標點的數量，第三行是使用本篇的方法後未能處理的目標點的數量，第四行是控制輸入(control input)的數量，最後是插入控制點後額外的面積開銷。

另外我們將 α 設為 100，在插入控制點後，如果沒辦法繼續減少目標點且目標點還維持在 100 個以上，我們會加入一個新的控制輸入信號，利用另一條控制輸入來減少目標點。

表 3：插入控制點後目標點數量

σ = 0.03, α = 100				
電路名稱	原來目標點數量	剩餘目標點	控制輸入數量	額外面積開銷
s5378	215	7	1	6%
s9234	684	7	1	7%
s13207	1258	4	2	7%
s15850	915	23	1	5%
s38417	1157	91	1	3%
s38584	2000	78	2	4%

4.3.2 剩餘目標點

剩餘目標點沒辦法藉由改變電路結構提升轉換機率，因此我們會針對剩餘目標點產生罕見的邏輯值 0 或 1 的輸入向量。在表 4 中，第二行為電路剩餘目標點的數量，第三行表示剩餘目標點中 redundant net 的數量。

以 s5378 電路為例，有 7 個剩餘目標點，但有 1 個目標點是 redundant net，所以會額外產生 6 個輸入向量，把這些向量隨機插入測試集合中就能觸發這些目標點，如果想觸發多次則可以重複插入這些向量。

表 4：剩餘目標點

電路名稱	剩餘目標點	redundant net 數量
s5378	7	1
s9234	7	1
s13207	4	0
s15850	23	3
s38417	91	0
s38584	78	53

4.3.3 計數器型木馬模擬結果

為了評估實驗結果，我們會將原始的電路及處理後的電路插入循序木馬，如圖 22，這是一個 10bits 計數器型的循序木馬，計數器從 0 增加到 1023 木馬電路才會被觸發。

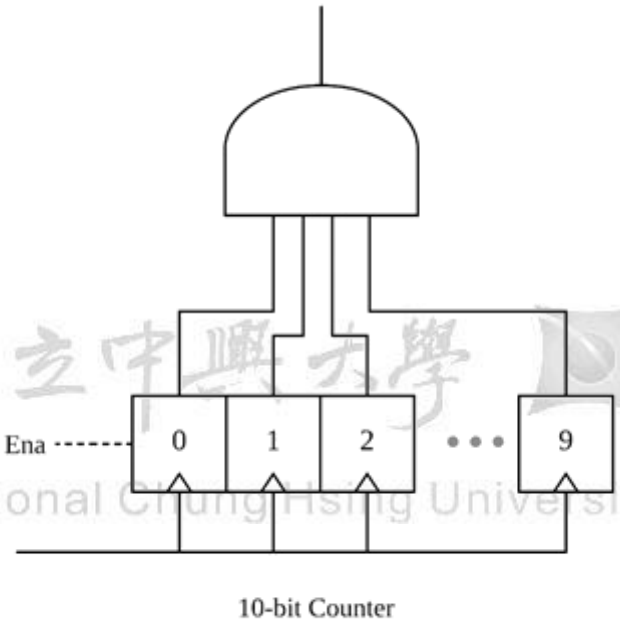

圖 22：10bits 計數器循序木馬

接著選出轉換機率低的信號線插入木馬電路，執行隨機邏輯模擬直到此木馬被驅動。表 5 為在 s5378 電路中隨機選出 10 條不同的轉換機率低的信號線插入木馬電路並執行隨機模擬直到木馬被觸發所需的模擬次數，另外因為電路結構的關係，某些插入後的木馬電路可能無法被觸發表示為 X。

表 5：s5378 模擬結果

電路名稱	信號線	未使用本篇方法的隨機模擬次數	使用本篇方法後的隨機模擬次數
s5378	n964gat	1038090	5617
	n145gat	259781	9197
	n968gat	687971	6787
	n111gat	542408	4048
	n870gat	682960	6930
	n2701gat	X	9233
	n445gat	644588	6531
	n197gat	524085	34535
	n1359gat	67965	66670
	n2986gat	11186474	11858

表 6 是 ISCAS'89 電路做以上步驟的平均結果，從結果可看出使用此方法後模擬次數少了 100 至 1000 倍左右。圖 23 用來表示使用本篇方法後模擬次數減少倍率的盒狀圖，縱軸為減少的倍率取對數，橫軸為基準電路，圖中的橫線為減少倍率的中位數，又號為減少倍率的平均數， 從圖中我們能看出減少的倍率範圍從 10 倍至 10000 倍。

表 6：10bits 計數器木馬模擬結果

電路名稱	未使用本篇方法的平均隨機模擬次數	使用本篇方法後的平均隨機模擬次數	減少倍率
s5378	1737147	16141	108x
s9234	10951520	10842	1010x
s13207	3524848	11281	312x
s15850	8873815	6993	1269x
s38417	838436	22314	38x
s38584	2940439	4515	651x

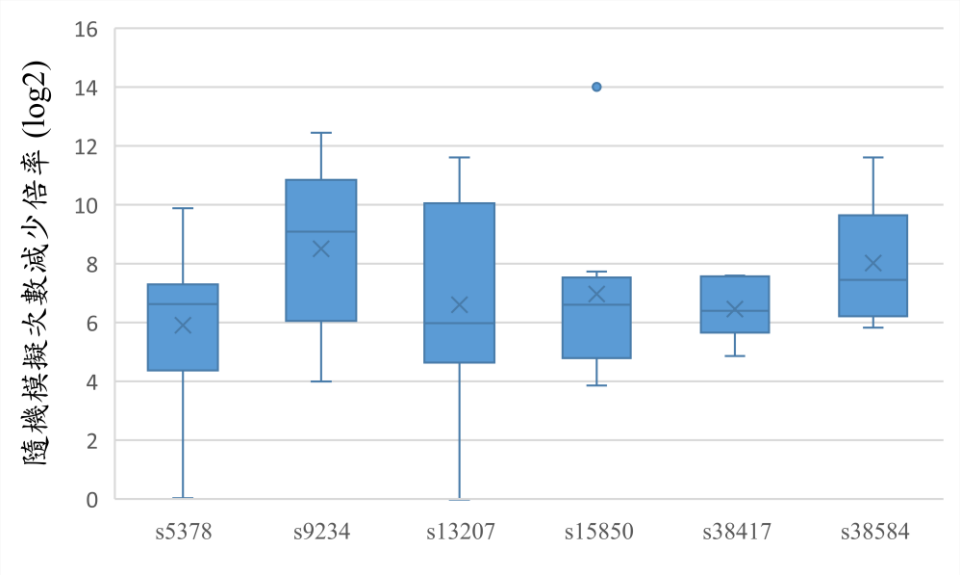

圖 23：計數器木馬隨機模擬次數減少倍率

4.3.4 有限狀態機型木馬模擬結果

為了評估實驗結果，我們設計一個簡單的有限狀態機的循序木馬，此木馬電路有四個狀態為 S_0 至 S_4，8-bit 輸入 In，當觸發條件成立時會進入下一個狀態，到達 S_4 時木馬會被驅動。如圖 24 所示:

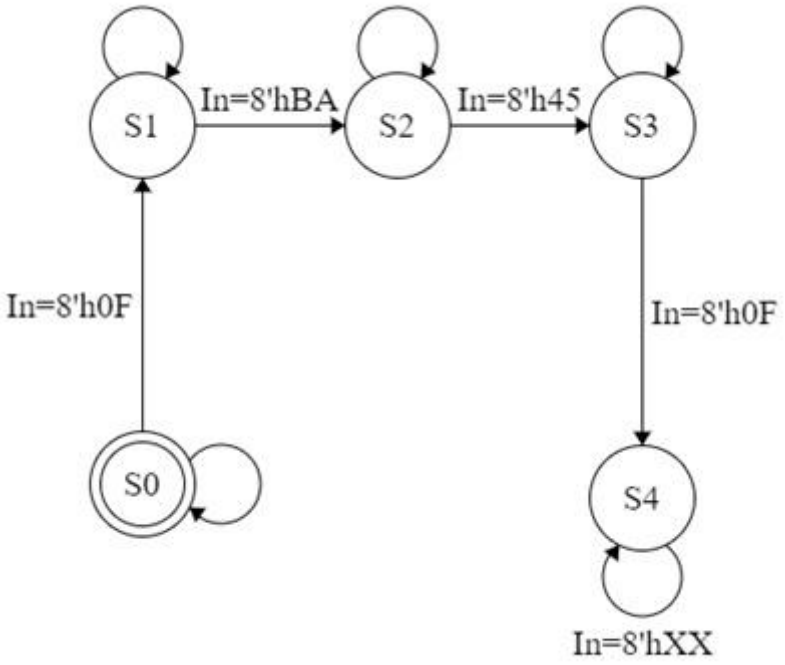

圖 24：4 個觸發條件的木馬狀態圖

為了讓木馬能有效地被觸發，我們從原始電路中隨機選出兩條罕見信號線接到 In 前兩個位元中來隱蔽木馬，其它位元為一般信號線，接著用隨機邏輯模擬來處發電路，各個電路跑五次模擬的平均結果如表 7:

表 7：有限狀態機木馬平均模擬結果

電路名稱	未使用本篇方法的平均隨機模擬次數	使用本篇方法後的平均隨機模擬次數	減少倍率
s5378	167436	6370	3x
s9234	27440	17997	15x
s13207	41724	16650	3x
s15850	39108	1312	30x
s38417	136378	1711	80x
s38584	10448	3437	3x

圖 25 用來表示使用本篇方法後模擬次數減少倍率的盒狀圖，從圖中我們能看出減少的倍率範圍從數十倍至數百倍。

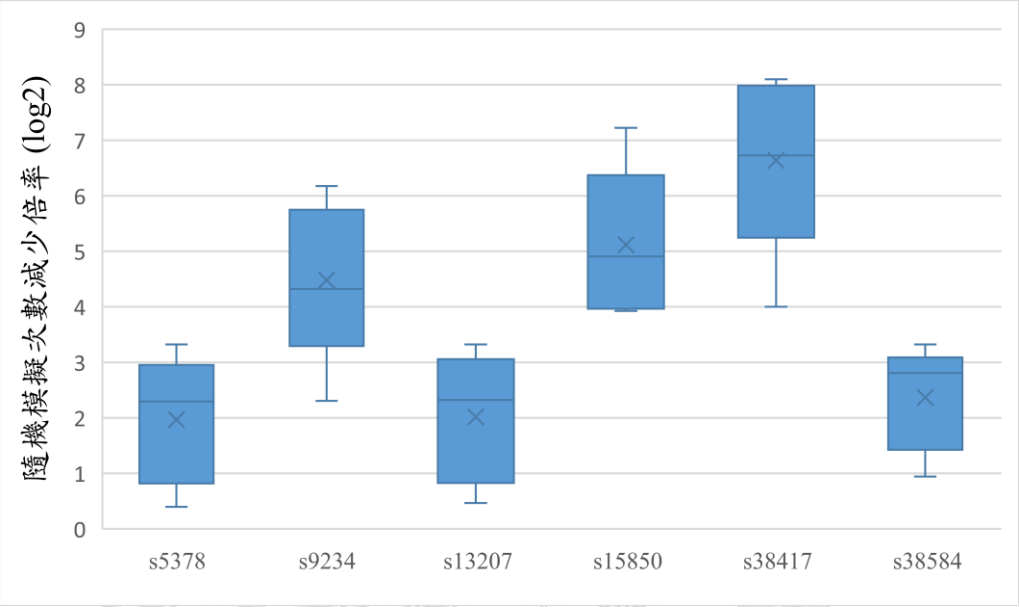

圖 25：有限狀態機型木馬隨機模擬次數減少倍率

從實驗結果可以得知，電路的大小與模擬的次數沒有直接關係，此種木馬必須考量木馬插入的位置，當木馬的輸入信號線觸發機率越低木馬越難觸發，也須考量信號線之間的關係以及木馬電路本身的設計，都可能是影響模擬次數的因素。因此，當此種木馬設計較複雜或木馬本身設計不良，木馬幾乎不可能驅動，對電路本身也只是無害的額外硬體木馬而已。

4.4 測試向量

我們使用 Atalanta-M ATPG Tool [31]產生原始及修改後電路的測試向量。因為 Atalanta ATPG 是針對組合電路產生測試向量的工具，

因此在使用之前必須先將 ISCAS'89 的循序電路(Sequential circuit)轉變為組合電路(Combinational circuit)。

如圖 26，組合電路與循序電路不同的地方在於循序電路有記憶體元件，其輸出與前一次的輸入狀態有關。因此我們須將 pseudo primary input 轉換成 primary input 及 pseudo primary output 轉成 primary output 就能將循序電路視為組合電路。

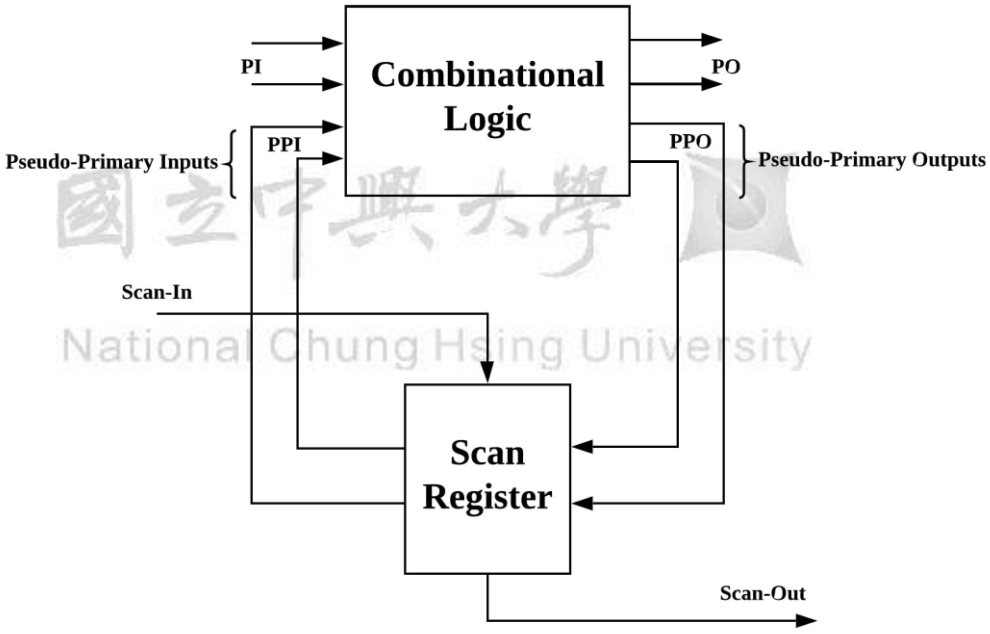

圖 26：循序電路轉成組合電路

表 8 為壓縮後原始電路及修改後電路對於 stuck-at fault 的測試向量數量，因為增加了額外的主要輸入信號，且這個額外的主要輸入信號通常會接到可控制性低的網路，使 ATPG 的困難度降低，所需的測試向量數量會下降。由此可知，增加控制輸入及控制點，可降

低硬體木馬的風險，也有助於減少測試向量的數量，但會增加額外的面積開銷及繞線的困難度。

表 8：測試向量之數量

電路名稱	原始測試向量	修改後測試向量
s5378	252	221
s9234	368	340
s13207	463	441
s15850	425	395
s38417	953	919
s38584	647	612

第五章 結論

本篇論文提出了對於快速偵測循序硬體木馬的研究，目的是要大量減少偵測硬體木馬的時間。我們先分析了 ISCAS'89 電路每條線的轉換機率以及可能插入木馬的位置。接著分析目標點，將其分類成相依及共同來源、相依、共同來源和剩餘之目標點。最後我們將會插入控制點到目標點來提升轉換機率，一旦木馬插入到這些位置，木馬的啟動時間會大幅縮減，因而被檢測出來。從 s38417 插入木馬後的模擬結果可得知，此方法能有縮減木馬電路的啟動時間。當木馬電路規模較小不容易透過側頻道信號方法檢測時，此方法能有效的預防以及偵測木馬。

雖然本篇方法會增加硬體面積，但能有效的偵測到循序木馬，最後我們也使用 Atalanta-M ATPG 觀察測試向量的變化，結果發現修改後的電路測試向量也會略為減少。

參考文獻

[1] M. Tehranipoor, H. Salmani, X. Zhang, M. Wang, R. Karri, J. Rajendran, and K. Rosenfeld, “Trustworthy hardware: Trojan detection and design-for-trust challenges,” *IEEE Computer Magazine*, 44(7):66 - 74, 2011.

[2] Y. Jin, N. Kupp, and Y. Makris, “Experiences in hardware Trojan design and implementation,” in *Proc. IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*, pp. 50 - 57, 2009.

[3] M. Tehranipoor and F. Koushanfar, “A survey of hardware Trojan taxonomy and detection,” *IEEE Design & Test of Computers*, pp. 10 - 25, 2010.

[4] G. T. Becker, A. Lakshminarasimhan, L. Lin, S. Srivathsa, V. B. Suresh, and W. Burelson, “Implementing hardware Trojans,” in *Proc. IEEE International Conference on Computer Design (ICCD)*, pp. 301 - 304, 2011.

[5] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan “Hardware Trojan attacks: threat analysis and countermeasures,” in *Proc. IEEE Special Issue on Trustworthy Hardware*, pp. 1229 - 1247, 2014.

[6] J. Cruz, Y. Huang, P. Mishra, and S. Bhunia, “An automated configurable Trojan insertion framework for dynamic trust benchmarks,” in *Proc. IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1598 - 1603, Mar. 2018.

[7] X. Wang, M. Tehranipoor, and J. Plusquellic, “Detecting malicious inclusions in secure hardware: challenges and solutions,” in *Proc.*

IEEE International Workshop on Hardware-Oriented Security and Trust (HOST), pp. 15 - 19, 2008.

[8] M. Banga and M. S. Hsiao, "A novel sustained vector technique for the detection of hardware Trojans," in *Proc. IEEE International Conference VLSI Design*, pp. 327 - 332, 2009.

[9] R. S. Chakraborty and S. Bhunia, "Security against hardware Trojan through a novel application of design obfuscation," in *Proc. IEEE/ACM International Conference on Computer-Aided Design* (ICCAD), pp. 113 - 116, 2009.

[10] M. Banga and M. S. Hsiao, "A region based approach for the identification of hardware Trojans," in *Proc. IEEE International Hardware-Oriented Security and Trust (HOST)*, pp. 40 - 47, 2008.

[11] S. Patnaik, N. Rangarajan, J. Knechtel, O. Sinanoglu, and S. Rakheja, "Circuit camouflage integration for hardware IP Protection," in *Proc. ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 97 - 102, 2018.

[12] F. K. Lodhi, I. Abbasi, F. Khalid, O. Hasan, F. Awwad, and S. R. Hasan, "A self-learning framework to detect the intruded," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1702 - 1705, 2016.

[13] K. Hasegawa, M. Yanagisawa, and N. Togawa,"Trojan-feature extraction at gate-level netlists and its application to hardware-Trojan detection using random forest classifier," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1 - 4, 2017.

[14] S. J. Wang, J. Y. Wei, S. H. Huang, and K. S. M. Li, “Test generation for combinational hardware Trojans,” in *Proc. 2016 IEEE Asian Hardware-Oriented Security and Trust (AsianHOST)*, pp. 1 - 6.

[15] M. Patterson, A. Mills, R. Scheel, J. Tillman, E. Dye, and J. Zambreno, “A multi-faceted approach to FPGA-based Trojan circuit detection,” in *Proc. IEEE VLSI Test Symposium (VTS)*, pp. 1 - 4, 2013.

[16] J. Rajendran, V. Jyothi, and R. Karri, “Blue team red team approach to hardware trust,” in *Proc. IEEE 29th International Conference on Computer Design (ICCD)*, pp. 285 - 288, 2011.

[17] S. Bhasin and F. Regazzoni, “A survey on hardware Trojan detection techniques,” in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2021 - 2024, 2015.

[18] K. Hasegawa, M. Yanagisawa, and N. Togawa, “A Trojan-invalidating circuit based on signal transitions and its FPGA implementation,” in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1 - 5, 2018.

[19] Y. Huang, S. Bhunia, and P. Mishra, “MERS: statistical test generation for side-channel analysis based Trojan detection”, in *Proc. ACM Conference on Computer and Communications Security*, pp. 130 - 141, 2016.

[20] J. Cruz et al., “Hardware Trojan detection using ATPG and model checking”, in *Proc. IEEE International Conference VLSI Design*, pp. 91 - 96, 2018.

[21] S. Wei, K. Li, F. Koushanfar and M. Potkonjak, “Hardware Trojan

horse benchmark via optimal creation and placement of malicious circuitry," in *Proc. IEEE/ACM Design Automation Conference (DAC)*, pp. 90 - 95, Jun. 2012.

[22] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using IC fingerprinting," in *Proc. IEEE Symposium on Security and Privacy*, pp. 296 - 310, 2007.

[23] R. Rad, X. Wang, J. Plusquellic, and M. Tehranipoor, "Power supply signal calibration techniques for improving detection resolution to hardware Trojans," in *Proc. IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 632 - 639, 2008.

[24] J. Li and J. Lach, "At-speed delay characterization for IC authentication and Trojan horse detection," in *Proc. IEEE International Workshop on Hardware-Oriented Security and Trust (HOST)*, pp. 8 - 14, 2008.

[25] H. Salmani, M. Tehranipoor, J. Plusquellic, "A novel technique for improving hardware Trojan detection and reducing Trojan activation time," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* pp. 112 - 125, 2012.

[26] B. H. Deiss, P. M. Trouborst, and M. H. Schulz, "Test point insertion for scan-based BIST," in *Proc. International European Test Conference*, pp. 253 - 262, 1991.

[27] M. Nakao, K. Hatayama, and I. Higashi, "Accelerated test points selection method for scan-based BIST," in *Proc. IEEE Asian Test Symposium (ATS)*, pp. 359 - 364, 1997.

[28] F. Brglez, "On testability of combinational networks," in *Proc. IEEE*

International Symposium on Circuit and Systems (ISCAS), pp. 221 - 225, 1984.

[29] N. Tamarapalli and J. Rajski, “Constructive multi-phase test point insertion for scan-based BIST,” in *Proc. IEEE International Test Conference (ITC)*, pp. 649 - 658, 1996.

[30] http://www.pld.ttu.ee/~maksim/benchmarks/iscas89/bench/

[31] http://ddd.fit.cvut.cz/prj/Atalanta-M/index.php?page=main