系統晶片強韌度驗證分析工具平台開發

陳信宇
中華大學資訊工程學系
300 新竹市香山區五福路二段 707 號
m09502044@chu.edu.tw

陳永源
中華大學資訊工程學系
300 新竹市香山區五福路二段 707 號
chenyy@chu.edu.tw

***摘要*—智慧型嵌入式系統常應用在一些需要高可靠度的領域，例如汽車駕駛系統以及智慧型機器人，同時智慧型系統的設計也大量的應用系統晶片的設計概念。製程進步已經進入到深次微米時代，高密度的系統晶片容易受到輻射線及電磁波的干擾，而降低晶片可靠度。使得系統開發者在開發新的智慧型系統時，必須嚴格審視系統晶片可靠度的相關問題。**

在本篇論文裡，我們以不更改目標系統晶片架構的前提之下，利用系統晶片軟體實踐錯誤注入的方式，去針對目標系統晶片內的處理器所擁有的暫存器作錯誤注入模擬實驗，並利用 FMEA 概念和失效類型的分類，以及實驗結果紀錄，提出一個失效類型分類模型，並根據分類模型開發一套自動化錯誤注入及失效類型分析工具。透過精簡化的使用者參數設定，讓系統開發者能在系統設計流程初期就可以透過此工具進行大量的模擬錯誤注入實驗，並分析實驗數據轉換成目標系統的強韌度資訊，讓系統開發者在系統設計流程初期就能掌握目標系統的強韌度和脆弱點，並依照目標系統對於可靠度的要求程度，針對脆弱點來加入有效的容錯機制。不僅協助設計者提高目標系統的可靠度，也可以提昇系統晶片可靠度設計及驗證的效率。我們以 ARM 為基礎的系統晶片來做實驗，透過設定少量的使用者自訂參數，來自動化完成大量的暫存器錯誤注入實驗。由實驗結果得知，錯誤一旦發生在暫存器單元時，系統可能產生八種不同的反應，包括七種不同的失效類型以及沒有造成任何失效。同時我們也可以統計出七種不同的失效類型以及沒有造成任何失效的機率分布，作為目標系統的強韌度參考。所開發的工具除了提供非侵入式方式進行模擬錯誤注入實驗，並且利用自動化實驗以及分析的方法來大大降低分析成本，提昇智慧型系統設計及驗證流程的速度。

一. 簡介

隨著嵌入式系統應用的普及化，複雜的智慧型系統已經成為人們日常生活不可或缺的一部分。例如智慧型手持裝置，車用電子控制裝置，航太設備…等等。而智慧型系統在設計流程上面普遍的加入系統晶片。由於人們更加的依賴電子產品，當智慧型系統應用的領域與人身安全有直接相關時，系統晶片的可靠度便成為不可忽視的重要議題。

當晶片製程進入深次微米時代之後，系統晶片的設計越來越複雜，包含的電晶體也越來越多，因此也越來越容易受到輻射線或是電磁波的干擾而出現軟性錯誤，特別是在暫存器以及記憶體中[1]。因此，系統晶片開發者必須要在設計初期，即能針對系統晶片強韌度做一完整的驗證分析，以利開發者決定是否需要加入容錯機制來提高系統晶片的可靠度。

所以，在決定是否將容錯機制加入系統晶片時，有兩個重要的課題必須先解決：一是如何在系統開發設計初期驗證目標系統的強韌度；二是如果需要加入容錯機制時，如何更快速有效的提供系統晶片對於失效模式以及效應分析(Failure Mode and Effect Analysis)的相關資料，讓設計者知道系統晶片的脆弱點，以及如何有效的加入容錯機制到系統晶片，降低系統晶片的開發、設計及晶片成本。

對於以上問題，要驗證目標系統的可靠度最直接的方法就是對目標系統進行錯誤注入實驗，透過實驗觀察系統遭受到錯誤污染時，所反

應的系統行為去分析出目標系統的強韌度以及脆弱點。因此，我們需要一套自動化的實驗以及驗證分析工具，將錯誤注入到目標系統的方式來快速驗證系統可靠度並找出系統的脆弱點。

二. 軟體實踐錯誤注入方法相關研究

錯誤注入模擬實驗是一個可以驗證系統可靠度的方法，主要是透過將錯誤注入到目標系統的方式，來模擬當系統遭受到輻射線或者是電磁波…等干擾系統運作時，觀察系統對於各種不同類型的錯誤表現出來的錯誤行為。而目前主要的錯誤注入方法有三種：

i. 實體錯誤注入法
ii. 模擬基礎錯誤注入法
iii. 軟體實踐錯誤注入法 (Software-Implemented Fault Injection , SWIFI)。

在(i)實體錯誤注入法中，這方法是透過一些特殊功能的硬體將錯誤注入目標系統，或是將目標系統直接用輻射線照射或電磁波干擾，並觀察目標系統在執行過程中受到雜訊污染後，所產生的錯誤行為。此類方法的優點是有較快的實驗模擬速度以及實驗精確度，但缺點是需要花費較高的實驗相關設備成本、實驗平台開發時間以及在實驗過程中可能造成目標系統的損壞。

在(ii)模擬基礎錯誤注入法中，這方法是透過電路模擬軟體模擬目標系統環境，例如 VHDL simulator。由於目標系統是在模擬的環境下，因此在做錯誤注入實驗時，可能會降低精確度與模擬速度，其優點是利用模擬環境可以大大降低開發成本及時間。

(iii)軟體實踐錯誤注入法，此方法是用軟體實作的方法去達成模擬錯誤注入，而目標系統的形式可以是實際硬體平台或是軟體模擬的系統，比較沒有限制。也因為是用軟體模擬的方式，在實作錯誤注入工具上會比較快速容易，也不需要改變目標系統架構或是額外的增加硬體，在設計錯誤注入工具的複雜度也會比較低，並且因應不同的錯誤注入模擬需求會比較容易修改，而且針對於不同平台的可攜性會比較高。

在先前 SWIFI 的相關研究中[2]，提到了一個利用 UNIX 的系統函式 fork()來產生母程序以及子程序的概念來模擬測試程式在目標系統執行遭遇到錯誤時，系統行為會如何反應。而在[3] FERRARI (Fault and ERRor Automatic Real-time Injector) 這篇論文中，更確實的說明出，利用系統函式 fork()產生子程序來模擬測試程式執行，而母程序則是用來監督子程序的執行過程，當程式執行到預先設定好錯誤注入中斷點時，母程序便透過系統函式 ptrace()來控制子程序，暫停子程序的執行，並根據設定好的錯誤型態(fault type)來針對子程序所使用的暫存器或者是記憶體空間作錯誤注入。而此工具對於暫時性錯誤以及永久性錯誤的實作提出了可行性的方法。此工具只針對程式計數器(program counter)來執行錯誤注入實驗，主要是要了解當錯誤影響到程式執行流程時，系統的錯誤反應，以及了解錯誤偵測的覆蓋率，但由於錯誤注入目標只有一個，當錯誤發生在其他可能的暫存器或者記憶體空間時，我們無法知道系統會出現何種錯誤行為。而在[4]中，作者提出了一套叫做 Xception 的錯誤注入工具軟體，這套工具利用了目標系統上面的內建的例外處理機制來啟動中斷並且執行錯誤注入，在設計工具的階段必須中斷處理器(interrupt handler)，在系統資源有限的嵌入式系統上，任意的更改目標系統架構，是否會降低系統穩定度以及提高工具設計的複雜度。

在[5]中，作者仔細介紹了如何利用作業系統，以及所提供的系統函數，來設計軟體錯誤注入工具，並在文中提到一些利用作業系統，以及系統函數庫，來設計工具時，所需要注意的點：

i. 在時間精確度方面，由於作業系統對於時間的解析度不一定相同，所以在時間精確度的設定上要特別留意。

ii. 在利用 ptrace()函數為主的方法中，會使用到大量的內文切換(content switch)。因此若測試程式執行時，所佔用的記憶體區塊太大的話，將會導致錯誤注入工具在執行中會變得很慢。

作者也提到利用系統函數 ptrace()，來設計錯誤注入工具時，可能會有的優缺點。

優點：

i. 工具的執行檔很小，不會佔用太多的目標系統資源
ii. 易於移植，針對以 UNIX 為基礎之不同的系統，只要稍微修改工具即可移植
iii. 在進行每一次實驗時，不需更動測試程式的原始碼內容，或是重新編譯測試程式。

在[6]中，提到了利用一些晶片內建的 on chip debugger，在非侵入式的情況之下，利用軟體錯誤注入的方式並搭配內建的除錯電路去針對系統內部資訊作存取及改寫。此方法可以在不需中斷測試程式的情況之下來驗證系統的容錯能力，但這並不是每一個晶片系統都有內建除錯電路。若目標系統的晶片沒有內建除錯電路，我們不一定還能正常使用此工具。

由於應用領域的不同，上述的文獻大多都是探討在高速工作站的系統或是具有內嵌除錯電路的系統上進行實驗。而在嵌入式系統廣泛運用的今日，上述提供的方法似乎沒有辦法提供一套完整的方案來完全地協助使用者，針對不同的目標系統進行錯誤敏感度實驗，並且針對大量實驗的結果執行自動失效類型分類與強韌度分析。

因此，在本篇論文中，我們以非侵入式的方式，以不更改目標系統的架構以及資源為前提，採用軟體實踐錯誤注入的方法開發一套適用於一般系統晶片的自動化軟體錯誤注入工具，並利用 FMEA 相關概念設計自動化錯誤管理分析工具，透過大量錯誤注入實驗的自動化分類以及分析，來提供一個針對於一般系統晶片在開發與驗證容錯設計的一個完整解決方案。

三. 自動化軟體灌錯分析工具架構

3.1 錯誤注入流程

為了完成上述的目標，我們採用軟體實踐錯誤注入的方法，並且以非侵入式的做法以及不改變目標系統晶片架構為原則去設計工具。我們利用以 Unix 為基礎的系統環境，利用作業系統所提供的系統函式 fork()，產生出子程序以及母程序。子程序為錯誤注入目標程序，負責執行錯誤注入實驗所要測試的目標程式;而母程序則是錯誤注入監控程序，負責處理錯誤注入實驗的一些必要的前置參數設定及更改暫存器或記憶體的內容。當母程序自動設定完參數之後，首先執行無錯誤注入(fault free)實驗，此實驗的結果紀錄將會當成之後進行實驗結果分類及分析的對照組。接著，母程序使用到另外一個系統函式 ptrace()，透過 ptrace()這個系統函式，母程序可以監督並控制子程序的執行狀況，並且透過 GETREGS 、 SETREGS 及 、 PEEKTEXT 、 POKETEXT…等 ptrace()函式參數的設定，使得母程序可以在特定的事件被觸發之後，暫停子程序的執行，並且存取在子程序中，目標程式所使用的暫存器以及記憶體中的內容值，進而進行錯誤注入。將錯誤注入到目標程式之後，母程序恢復子程序的執行，並紀錄每一次實驗的結果。

3.2 失效類型分類及分析方法

為了要取得高可信度的數據來分析目標系統的強韌度以及脆弱點，我們必須依靠大量的錯誤注入實驗，並從實驗結果觀察系統運作時受到干擾的反應現象，來做分類並統計歸納。因為不同的測試程式特性，以及錯誤注入的時間點及發生錯誤的暫存器位置的差異，我們可以歸納出系統運作時遭受干擾時所產生的八種反應 如圖 1 所示。

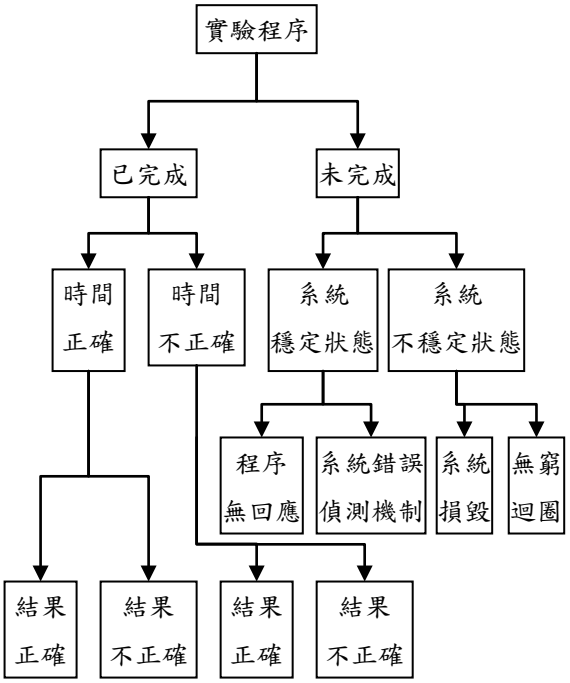

圖 1. 失效類型分類圖

首先,第一層先將實驗程序分成已完成跟未完成，第二層分為四個項目，根據已完成的部份分成時間正確以及時間不正確類型，未完成的部份則是分成系統穩定狀態以及系統不穩定狀態類型，而最後總共可分為如圖 1 所示的八種不同類型。

- 時間正確結果正確：此類型代表注入的錯誤並沒有造成任何失敗，代表這個錯誤是無效性的錯誤。
- 時間正確結果不正確：此類型代表注入的錯誤造成程式的輸出結果遭到污染，但是程式仍然在正確的時間點結束。
- 時間不正確結果正確：此類型代表注入的錯誤造成程式的執行時間不正確，但是程式的輸出結果是正確的。
- 時間不正確結果不正確：此類型代表注入的錯誤造成程式的執行時間不正確，而且程式的輸出結果是錯誤的。
- 程序無回應：此類型代表錯誤注入到暫存器之後，程式運行的必要資料被注入的錯誤污染而導致整個程式不正常終止。但此時作業系統仍然可以正常運行。
- 系統錯誤偵測機制：此類型是代表錯誤注入到程式裡面所產生的錯誤反應，剛好被作業系統的偵錯機制偵測並處理。
- 無窮迴圈：此類型代表錯誤注入到迴圈相關的判斷變數上面，導致產生迴圈運算無法結束跳出而進入無窮迴圈。
- 系統損毀：此類型是代表注入的錯誤讓整個系統出現無法恢復的錯誤，導致系統必須重新開機方可恢復。

上述八種類型中,包含了七種不同的失效類型以及一種沒有造成任何失敗的類型。而有了上述的失效類型分類的樹狀圖之後，接下來我們必須從大量的實驗結果紀錄檔中，去區分每一個實驗紀錄檔是屬於何種類型，進而可以反推在實驗過程中，系統受到干擾時，當下所反應出來的行為。因此我們以下列四項所描述的特徵，去觀察每一個錯誤注入實驗結果記錄檔，並以一開始執行無錯誤注入(fault free)實驗所產生的記錄檔，當作比對的對照組。

i. 子程序的結束狀態

子程序是屬於異常結束或是正常結束

ii. 子程序的結束時間

為排除作業系統對於工作排程，所造成的時間延遲，我們執行數次的無錯誤注入實驗，來找出子程序結束時間的共同參考值，並以此時間共同參考值，檢視錯誤注入實驗記錄檔案中的子程序結束時間。若不同，則判斷此錯誤發生，造成程式執行時間的異常。

iii. 測試程式的輸出結果

比較錯誤注入實驗結果記錄檔的測試程式輸出結果，是否與對照組相同。

iv. 實驗結果記錄檔案的容量

為了能區別出程式系統損毀類型及無窮迴圈失效類型，我們除了觀察上述三項特徵之外，並加入實驗結果記錄檔案容量作為第四項特徵。在子程序結束狀態這項特徵中，若呈現無結束狀態訊

息，則其失敗類型可能是系統損毀失效類型、或無窮迴圈失效類型。在這二種失效類型中，由於系統損毀失效類型不會產生執行結果記錄檔案，而無限迴圈則是會產生數量龐大的不正確資訊。所以我們需要利用實驗結果記錄檔案的容量，來做為這二種失效類型的區分特徵。若實驗記錄檔的容量為不為零，則代表此次實驗因受到錯誤污染，而產生子程序無止盡執行的系統行為，導致實驗記錄檔案容量爆增，我們將之歸類於無窮迴圈失敗類型。實驗記錄檔的容量為為零，則代表子程序損毀無法回傳結束狀態，甚至為監督者的母程序所產生的既有訊息也沒有出現在記錄檔案。進而可推斷，當此次實驗及遭遇到的錯誤干擾若出現在實際操作的系統上，作業系統可能因此錯誤而導致無法回應，必須重新開機。

在錯誤注入實驗結果記錄檔所記錄的資訊中，我們找到上述四種特徵。我們可透過這四種特徵去區分出八種不同的系統反應，而這四種特徵在八種不同系統反應所代表的意義如下：

◆ 時間正確結果正確(Correct Time Correct Data , CTCD)：

子程序的結束狀態方面，此分類是屬於正常結束。子程序的結束時間方面，是屬於正確的。在測試程式輸出結果方面，是屬於正確的。在實驗結果記錄檔案容量方面，是屬於正確的。

◆ 時間正確結果不正確(Correct Time Incorrect Data , CTID)：

子程序的結束狀態方面，此分類是屬於正常結束。子程序的結束時間，是屬於正確的。在測試程式輸出結果方面，是屬於不正確的。在實驗結果記錄檔案容量方面，是屬於正確的。

◆ 時間不正確結果正確(Incorrect Time Correct Data , ITCD)：

子程序的結束狀態方面，此分類是屬於正常結束。子程序的結束時間方面，是屬於不正確的。在測試程式輸出結果，是屬於正確的。在實驗結果記錄檔案容量方面，是屬於正確的。

◆ 時間不正確結果不正確(Incorrect Time Incorrect Data , ITID)：

子程序的結束狀態，此分類是屬於正常結束。子程序的結束時間方面，是屬於不正確的。在測試程式輸出結果方面，是屬於不正確的。在實驗結果記錄檔案容量方面，是屬於正確的。

◆ 程序無回應(Process Hang , PH)：

子程序的結束狀態方面，此分類是屬於不正常結束。子程序的結束時間方面，由於是不正常結束，故程式結束時間無法做為判斷依據。在測試程式輸出結果方面，由於是不正常結束，測試程式輸出結果不一，故無法做為判斷依據。而在實驗結果記錄檔案容量方面，由於程序無回應是屬於程式受到干擾而無法繼續執行，但是作業系統仍然是正常執行的狀態，所以實驗結果記錄檔案容量不會爆增。

◆ 無窮迴圈(Infinite Loop , IL)：

子程序的結束狀態方面，此分類是屬於沒有結束。子程序的結束時間方面，由於是沒有結束，故程式結束時間無法做為區分參考。在測試程式輸出結果方面，由於是不正常結束，測試程式輸出結果因受干擾，而可能產生大量無意義資訊，故測試程式輸出結果無法做為區分參考。在實驗結果記錄檔案容量方面，由於無限迴圈會造成實驗結果記錄檔案的容量，因為多了很多無意義的資料而造成爆增的現象。

◆ 系統錯誤偵測機制(System Error Detect Mechanism , SEDM)：

子程序的結束狀態方面，因為錯誤被作業系統的偵查錯誤機制偵測並排除，所以此分類是屬於正常結束。子程序的結束時間方面，由於子程序被作業系統提早結束的關係，所以程式結束時間非常短，是屬於不正確的。在測試程式輸出結果方面，由於作業系統已經透過偵查錯誤機制將子程序安全而快速的結束掉了，因此子程序所執行的測試程式輸出結果也一併視為是錯誤而不會輸

出，所以在測試程式輸出結果方面是屬於不正確的。而在實驗結果記錄檔案容量方面，因為只有一些母程序所產生之少量既有訊息，為了提高工具分析準確度，避免與 ITID 失敗類型產生誤判，除了判斷測試程式輸出結果沒有內容之外，並與無錯誤注入實驗檔案容量做比對，若屬於系統錯誤偵測機制失敗類型，則實驗結果紀錄檔案容量會低於下限值，所以在實驗結果記檔案容量中也是屬於不正確的。

◆ 系統損毀(System Crash , SC)：

子程序的結束狀態方面，若目標系統無法回應，或是產生出來的實驗記錄檔案沒有結束資訊，則代表模擬程式執行的子程序受到嚴重破壞而無法記錄結束狀態，甚至注入的錯誤直接造成工具或者是目標系統崩毀，所以在四個特徵中皆是屬於不正常的狀態。

我們將以上的敘述以表格的方式呈現，‘V’代表在範圍內或合理，‘X’代表在範圍外或不合理，‘◎’代表無法作為區分參考，如表 1 所示：

表 1. 失效類型特徵表

	子程序結束狀態	子程序結束時間	測試程式輸出結果	實驗結果紀錄檔容量
CTCD	V	V	V	V
CTID	V	V	X	V
ITCD	V	X	V	V
ITID	V	X	X	V
PH	X	◎	◎	V
IL	X	◎	◎	X
SEDM	V	X	X	X
SC	X	X	X	X

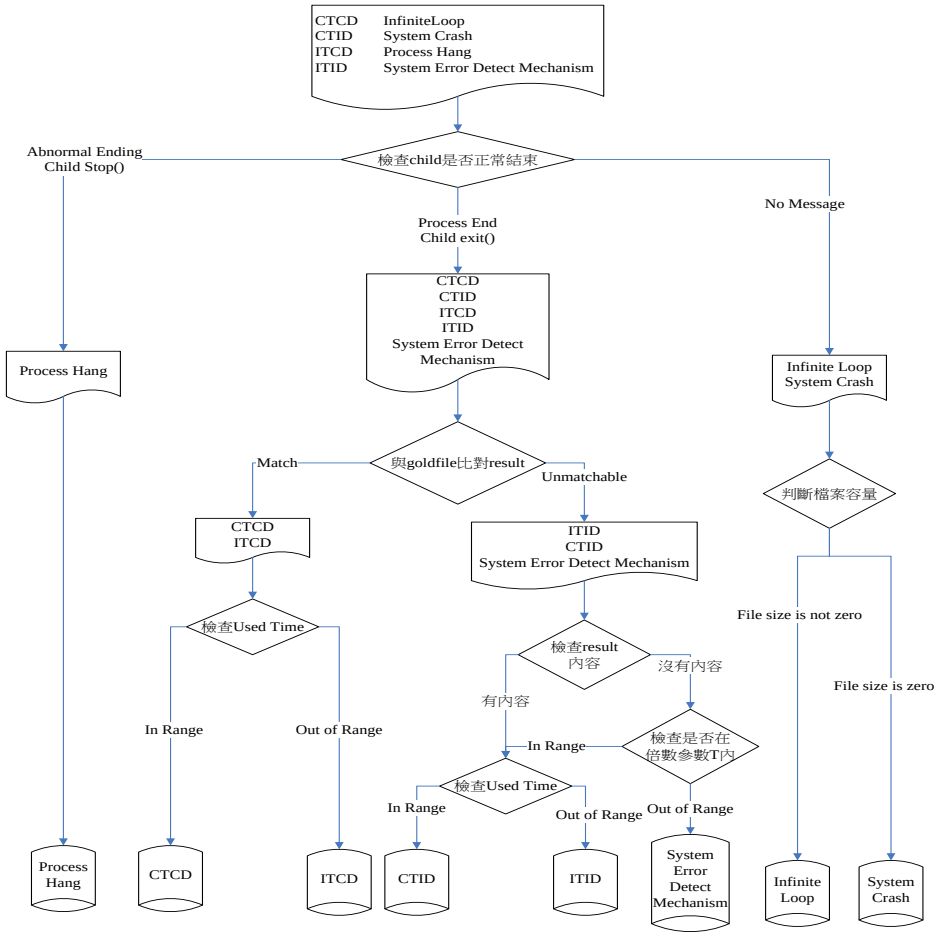

圖 2.自動化分析模型流程圖

依照上述的分類法則，我們制定出一個自動化的分析模型(Automatic Analysis Model)。此分析模型是將上述的分類法則以程式流程圖及一些程式技巧的撰寫方式表示，並且轉換成與工具結合的模組，以進行錯誤注入實驗之後的自動化分析與分類。圖 2 展示分析模型流程圖。在這邊值得一提的是，在分析錯誤類型的特徵中，工具必須定義一組合理的時間範圍來區分每一次的實驗結果是否落在合理的時間範圍內。而由於作業系統在執行程式的過程中，會因為工作排程以及一些內文切換的動作，使得每一次的執行結束時間皆有些許不同，所以合理的時間範圍必定不是為一個固定的值。因此，我們使用了蒙地卡羅模擬法(Monte Carlo Simulation)的概念，透過統計的方式來找出一組合理的正確時間範圍。

蒙地卡羅模擬法，是基於大數法則的實證方法，當實驗的次數越多，其平均值也就會越趨近於理論值。因此我們透過大量蒐集並統計無錯誤注入實驗結果的結束時間，來找出合理的時間範圍。

1. 首先，我們先透過執行無錯誤注入實驗，並擷取每一次實驗的程式執行時間，執行次數預設值為 50 次。執行完成之後，我們在這 50 組的程式執行時間中，找出最大值以及最小值，並且個別存在 Time_MAX1 以及 Time_min1 兩個變數中，而完成一整個程序我們稱作是一個 Run。
2. 將 Time_MAX1 以及 Time_min1 分別複製一份到 Time_MAX2 以及 Time_min2 兩個變數中。Time_MAX2 以及 Time_min2 分別儲存目前找到的合理時間最大值以及最小值。
3. 之後，再次執行一個 Run，可以得到另一組的時間範圍存放在 Time_MAX1 以及 Time_min1 中。
4. 將 Time_MAX2 跟 Time_MAX1 比較大小，若 Time_MAX1 > Time_MAX2，則改寫目前存放的最大合理時間。
5. 將 Time_min2 跟 Time_min1 比較大小，若 Time_min1 < Time_min2，則改寫目前存放的最小合理時間。
6. 定義一個變數 judge 記錄收斂次數，若在執行一個 run 之後，發現上述步驟 4 及步驟 5 並沒有改寫之動作，代表這一次的 run 是在收斂範圍內，將 judge + 1，並進行下一次實驗。若下一次實驗的步驟 4 或步驟 5 有改寫的行為，則 judge 歸零。
7. 當 judge = 2 的時候結束蒙地卡羅模擬，換句話說，必須找到連續兩次都落在統計的合理時間範圍之內才代表接近理論值的時間範圍。

透過圖 3，可以很清楚知道程式的演算法運作…

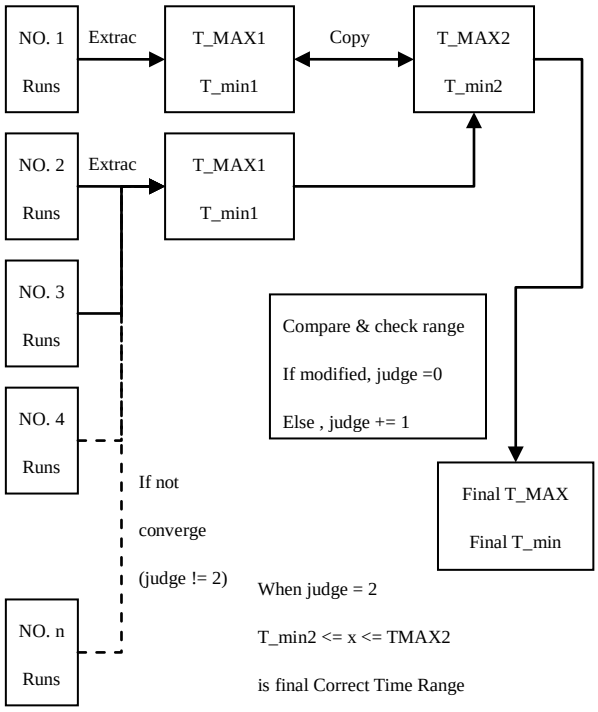

圖 3. 蒙地卡羅模擬法流程

透過上述蒙地卡羅模擬法的概念可以找出逼近母體的合理時間範圍，而越逼近母體的合理時間範圍，在判斷程式結束時間的特徵上可大幅

提高準確度。

這邊有兩個參數會間接影響蒙地卡羅模擬法準確度：

- 一個 Runs 所包含的執行次數
- 判斷收斂的程度(judge 變數)

在 runs 執行次數方面，若次數太小，則有可能每一次的 run 中，所表現出來的時間差距不足以模擬真實情況。

而在 judge 變數上，若次數太少，則判斷收斂的程度則有可能不夠大。

雖然加大上述兩個變數的值，皆有助於逼近理論的合理時間範圍，但卻會大幅降低分析效能，因此必須設定一個合理的參數，來取得一個平衡點。

3.3 自動化錯誤管理分析工具

根據上述的四項特徵以及分析模型，我們可以將每一次的實驗結果所反應出來的系統行為做正確的分類。而為了提高分析準確度，每一次實驗皆需要大量的實驗數據來佐證。因此，我們為了降低分析成本(analysis cost)，而設計了一個自動化錯誤注入及管理分析工具，並且精簡化使用者參數設定，讓系統開發者能夠透過此工具進行大量錯誤注入模擬實驗，進而快速的掌握目標系統的強韌度資訊以及脆弱點，進而協助系統開發者提高目標系統的可靠度，以及驗證的效率。

圖 4 為自動化錯誤注入及管理分析工具的設計流程圖：

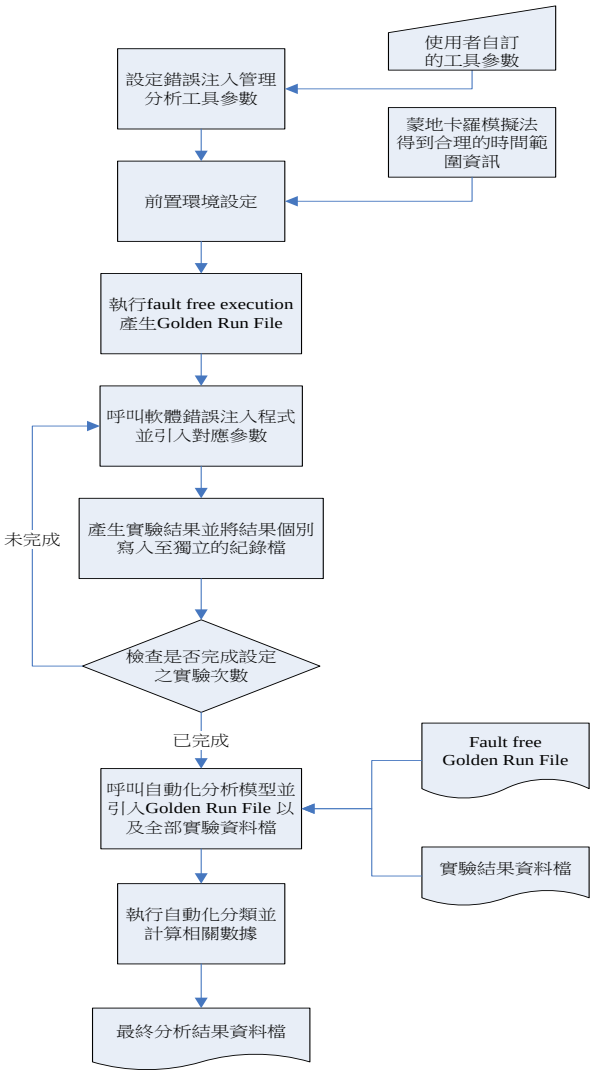

圖 4. 自動化錯誤注入及管理分析工具流程圖

在這自動化工具的流程中，第一步就是要針對使用者對於實驗的參數需求去初始化工具，而使用者要設定的參數如下:

i. 測試程式
ii. 錯誤注入實驗執行次數
iii. 目標暫存器
iv. 錯誤注入觸發時間

引入完使用者參數並完成工具前置設定之後，接下來就準備進行錯誤注入實驗。首先會先執行無錯誤注入實驗，產生的無錯誤注入實驗記錄檔案稱作 Golden Run File，裡面所紀錄的資訊用來當作之後分析用的對照組。

產生出 Golden Run File 之後，接著準備進行錯誤注入實驗，工具可以根據剛剛使用者設定的測試程式以及錯誤注入實驗的執行次數…等參數進行一連串的錯誤注入實驗，並且將個別的實驗結果紀錄到不同的檔案。完成錯誤注入實驗之後，工具接著引入自動化分析模型，而自動化分析模型的分類法則是套用 **3.2** 章節所提到的分析法則，並且帶入無錯誤注入實驗記錄檔案當作對照組，將產生出來的大量實驗記錄檔案進行自動化分析及分類。

工具將實驗記錄分析結果並分類完成後，接著彙整數據，並計算各種失敗類型的機率分佈，並將資料輸出成結果資料檔以利系統開發者參考。

四. 實驗結果

在[1]中提到，由於soft error發生在暫存器的機率相當高，所以我們把實驗目標定在目標系統晶片的暫存器檔案上面，並且透過大量的自動化錯誤注入及分析工具來驗證系統晶片的強韌度。本篇論文使用(CDK)[1]嵌入式系統開發平台來做為軟體錯誤注入實驗的環境，而整個平台包括ARM926EJ-S的核心以及Open Linux 2.6.19 為平台的作業系統[7]，並選擇ARM處理器中，在使用者模式下的R0~R17 暫存器進行實驗。我們將以下列實驗來展示軟體灌錯管理分析工具如何自動化完成灌錯實驗以及分析數據。

4.1 蒙地卡羅模擬法實驗合理化時間的差異性

由於作業系統在執行程式的過程中，會因為工作排程以及一些內文切換的動作，使得每一次的執行結束時間皆有些許不同，所以使用蒙地卡羅模擬法來取得合理的時間範圍。而實驗中有兩個參數，分別是 judge 以及 runs 所包含的個數，這兩個參數的設定將會影響合理時間範圍的正確性，我們將 judge 設定為 2，分別觀察當 runs 為 5，10，50，100 時，對於合理時間判斷的準確度，實驗結果如圖 5 所示：

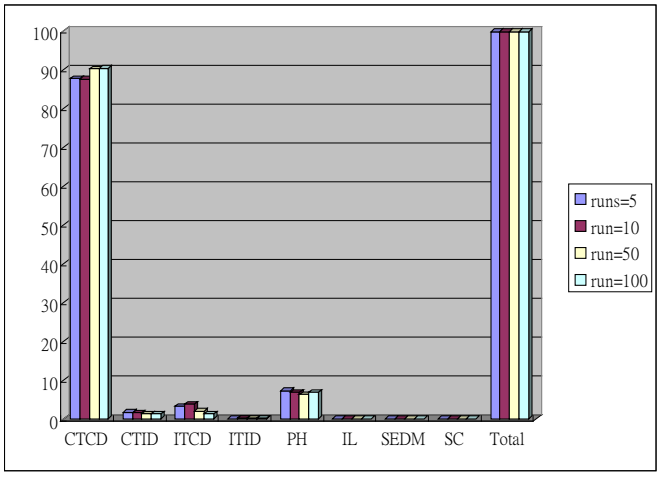

圖 5. 不同參數設定使用蒙地卡羅模擬法之差異圖

由圖 5 的數據可以發現，當 runs 越高，則 CTCD 的精確度也越高，而 ITCD 的數量也會明顯變少。此數據可以驗證使用蒙地卡羅模擬法的概念可以提昇合理化的時間範圍準確度，而提高每一個 runs 所要執行的次數則會使分析工具的效率些微下降。

4.2 個別暫存器之注入錯誤分析實驗

實驗設定為 50 × 50 的矩陣相乘運算執行 1000 次實驗，目標暫存器為 R0，錯誤注入時間範圍為 0us ~ 250us 之間隨機產生。

將整套自動化軟體錯誤注入管理分析工具在目標系統上面執行錯誤注入實驗，執行 1000 次錯誤注入實驗及分析資料總共耗時 15 分鐘，而分析結果如下：

在完成 1000 次的錯誤注入實驗以及自動化分析之後，工具會在目標系統上面建立七種不同的目錄，分別表示失敗類型以及一種沒有造成任何失敗的類型，如圖 6 所示：

[1] Socle Technology – SoC Platform Solution and Service Company

```
[Socle@DK /]$ ls
CTCD                          home
CTID                          lib
ITCD                          linuxrc
ITID                          lost+found
Infinite_Loop                 matrix_multiplication_ARM.out
Process_Hang                  mnt
SWIFI_beta.out                private
System_Crash                  prob_analysis_ARM.out
System_Error_Detect_Mechanism proc
analysis_result.txt           root
bin                           sbin
boot                          sys
dev                           test_4.sh
etc                           tmp
get_file.sh                   usr
gold.txt                      var
[Socle@DK /]$
```

圖 6.系統行為分類目錄

在圖 6 中，包含了八個不同的系統行為目錄，以及一個最後的分析結果資訊(analysis_result.txt)，裡面記載著實驗結果各種的數據，以及系統的可靠度資訊，協助系統開發者更快速的了解目標系統的強韌度以及脆弱點。

分析結果資訊紀錄如下圖：

```
Successful Experiment = 1000
Fail Experiment =0
Num_CTCD = 934
Num_CTID = 8
Num_ITCD = 35
Num_ITID = 0
Num_PH = 23
Num_IL = 0
Num_SEDM = 0
Num_SC = 0
CTCD = 93.40000 %
CTID = 0.80000 %
ITCD = 3.50000 %
ITID = 0.00000 %
Process Hang = 2.30000 %
Infinite Loop = 0.00000 %
System Error Detect Mechanism = 0.00000 %
System Crash = 0.00000 %
```

圖 7. 分析結果資訊紀錄

此次實驗的分析結果如圖 8 所示：

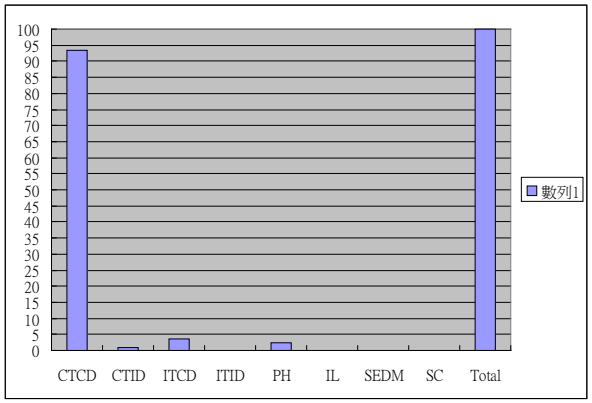

圖 8.實驗分析結果分佈圖

由圖 8 的數據可以看出，目標系統在執行 50 × 50 矩陣相乘時，當暫存器 R0 受到干擾時，仍然有 93.4%的機率是不受影響的，只有 0.8% 的機率會造成程式輸出結果出錯。這數據告訴系統開發者，R0 受到干擾時會導致系統出現資料錯誤失敗行為的機率約 6.6%，此數據將可以提供系統開發者，針對系統可靠度機制開發的一個參考。

另一方面，在數據中我們可以看到有 0.8%的機率會造成時間正確結果不正確的失效類型，我們透過自動化工具所分類好的目錄進去檢視 CTID 中，檢視被分類到 CTID 類別中，實驗結果失敗的情況，如下圖 9 所示：

```
50 50 50 50 50 50
50 50 50 50 50 50
50 50 50 50 49 50
50 50 50 50 50 50
50 50 50 50 50 50
50 50 50 50 50 50
50 50 50 50 50 50
50 50 50 50 50 50
50 50 50 50 50 50
```

圖 9.CTID 類型的實驗結果

在完全沒有任何失效的類型中，50 × 50 矩陣相乘的測試結果輸出為一個 50 × 50 的二維陣列，每一個元素的數字皆為 50，而當測試程式遭受到錯誤干擾後，有可能會造成如圖 9 中圈起來的部份，測試程式的輸出值因為系統遭受到污染而輸出錯誤的答案。

4.3 暫存器檔案注入錯誤分析實驗

接著我們進行一個實驗，以整個暫存器檔案為一個元件進行軟體錯誤注入實驗，來模擬當智慧型系統暫存器檔案遭受錯誤干擾時，系統所呈現的反應為何。實驗設定如下：

- 測試程式：50 × 50 矩陣相乘
- 實驗次數 10808 次
- 每次實驗中，目標暫存器檔案 R0 ~ R17 隨機選擇
- 錯誤觸發時間：測試程式執行時間範圍內隨機選擇

實驗數據如圖 10 所示：

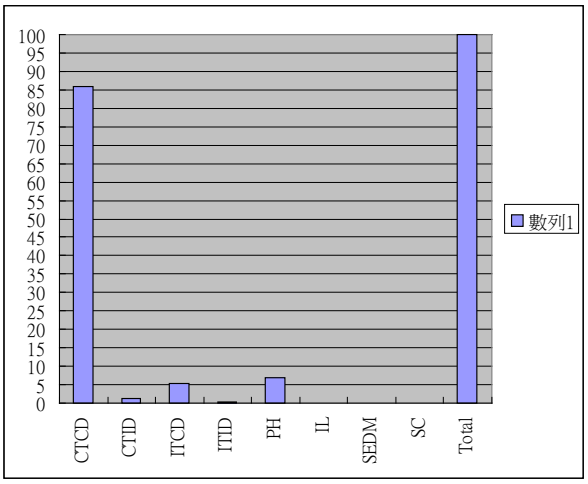

圖 10 暫存器檔案綜合錯誤注入實驗結果

在圖 10 中我們可以透過工具，很快的得到暫存器檔案的錯誤注入實驗分析結果。由數據可以很清楚了解，當系統遭遇到輻射線或電磁波干擾時，有 86%的機率是不會有影響的，換言之有 14%的機率是會造成系統失效的類型出現。而在各失效類型中，又以程式無回應這種失敗類型為最高，達到約 7%的機率會造成程式無回應，次之則是輸出結果錯誤，而這些綜合數據也可以提供系統開發者一個資訊去了解目標系統整體對於執行測試程式中遭受到錯誤污染，系統整個的失效類型分佈。

五. 結論

在本篇論文中，我們以非侵入式的原則及不更改晶片系統的架構之下，採用軟體實踐錯誤注入的方式，完成一套自動化的軟體灌錯管理及分析工具，並且以 FMEA 的錯誤類型分類概念去設計一個自動化分析模型並套用到工具中。藉由此工具的自動化實驗分析，系統開發者可以在開發階段早期就透過工具了解到目標系統的強韌度資訊以及脆弱點，並且透過精簡化的使用者參數設定以及高程度的自動化分類以及分析，可以使用大量的實驗數據來提高分析結果的精確度以及大幅降低分析成本 (analysis cost)。在智慧型嵌入式系統大幅應用的今日，高度自動化的實驗分析工具有助於系統開發者降低系統的開發以及驗證成本，並且降低 Time to Market 所需要的時間成本，使得智慧型嵌入式系統能夠越來越可靠及人性化。

六. 參考文獻

[1] Alfredo B, Paolo P. “Fault injection techniques and tools for embedded systems reliability evaluation”, Kluwer Academic, 2003

[2] Raphael R. Some*, Won **S.** Kim, Garen Khanoyan, Leslie Callum, Ani1 Agrawal, and John J. Beahan “A Software-Implemented Fault Injection Methodology for Design and Validation of System Fault Tolerance”, Int. Conference on Dependable Systems and Networks, pp. 501-506, July 2001

[3] G.A. Kanawati, N.A. Kanawati, and J.A. Abraham, “FERRARI: A Flexible Software-Based Fault and Error Injection System,” IEEE Trans. on Computers, vol. 44, no. 2, pp. 248-260, Feb. 1995.

[4] J. Carreira, H. Madeira, and J.G. Silva,

"Xception: A Technique for the Experimental Evaluation of Dependability in Modern Computers," IEEE Trans. on Software Eng., vol. 24, no. 2, pp. 125-136, Feb.1998.

[5] Sieh, "Fault-Injector using UNIX ptrace Interface", Internal Report 11/93, IMMD3, Universität Erlangen-Nürnberg, 1993.

[6] Fidalgo, A.V.; Alves, G.R.; Ferreira, J.M., "Real time fault injection using a modified debugging infrastructure", 12th IEEE Int. On-Line Testing Symposium, Page(s):6,10-12 July 2006.

[7] http://www.socle-tech.com.tw