新一代運動控制系統設計方法和整合環境的介紹

陳國任

壹、前言

對於使用於精密切削或高速工具機上的運動控制系統，以系統的控制性能而言通常需滿足下列要求：

一、 快速移動，也就是極短的上升時間。
二、 如爲高精密定位，則需爲零穩態誤差。
三、 不能擁有或只允許極小的超越量
四、 到達穩態時間很短，在幾個取樣區間後，即和參考訊號完全一致。

除了上述的要求外，運動控制系統也需滿足和實際環境間的互動所造成的複雜行爲與滿足來自外界的服務要求，最後也需符合各種不同的時間限制。這些要求皆和系統真正的時態特性(Timing Behavior)有關，而不是只考慮控制演算法的演算邏輯即可。因此運動控制系統通常擁有即時(Real-time)和多工(Multi-tasking)的特性，此外系統通常也具有事件驅動(Event-driven)的特徵。

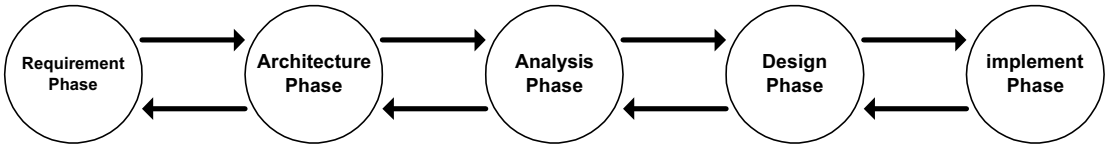
但傳統的運動控制系統設計方法並無法滿足此類系統的真正需求，除了系統特殊的運作方式外，在於傳統運動控制系統的發展過程中，存在的問題使得最終系統無法滿足最初的規格要求。這些問題主要在於運動控制系統通常在完成硬體架構和軟體撰寫後，才能經由實際測試評估系統的真正性能和可靠度，但此時的性能並不能保證必如原先所規劃的結果。此外當系統的真正性能無法滿足原設計目標時，這些測試結果一般並沒有提供足夠的資訊，可作爲系統中包括軟、硬體和控制演算法改進的參考。

爲了解決上述問題，新一代的運動控制系統整合發展環境提供了以模型爲基礎的架構(Model-Based Architecture)，具有廣泛、有彈性、富有成效等特色，取代傳統的工程設計方法，以克服在控制系統設計上的關鍵挑戰，諸如定義、解決標準的、或是進階的控制問題，並可自動從系統模型中產生內嵌式程式碼。因此整合式的運動控制系統發展環境需提供下列功能：

一、 可利用已發展模型爲基礎，快速而正確地發展系統模型。
二、 以互動的方式爲動態或事件驅動系統建立模型，並加以模擬與分析。
三、 全自動地從系統模型中產生 C 程式碼或其他必須的程式碼
四、 在建立高成本的原型之前，可反覆對系統設計結果進行測試並加以改善。

貳、目前系統設計方法發展現況和缺陷

目前對實作於單一微處理器中的運動控制系統，皆假設硬體的性能足以滿足實際的需求，因此並不考慮硬體架構對控制行爲的影響，而只專注於控制演算法的設計與軟體的撰寫；另外在單純的單工環境中，實際運作時各伺服迴路間並不會相互影響。下圖即爲一般對於運動控制系統的發展步驟。



但對於實作於單一微處理器中的多個伺服迴路來說，複雜的運作方式就難以保證軟、硬體的性能必能實作控制演算法，造成此種結果的主要原因爲硬體和即時核心於運作時產生不同的時間延遲與必須處理的突發事件；此外，也可能是系統所提供的資源無法滿足運作時的實際需要。但傳統的分析方法只能就即時系統是否發生死結(Deadlock)的現象進行討論，並無法獲得並分析下列幾個問題：

一、 運動控制系統因多工特性所造成的各種時間延遲現象，包括從取樣動作開始至完成控制訊號輸出之間的時間長度，和對緊急事件從發生至開始處理的反應速度(Responsiveness)。多工環境並無法保證這些行爲皆可在同一取樣區間或要求時間內完成。

二、 在多工環境下運動控制系統實際的取樣區間可能因系統排程的原因，無法爲固定值而只能侷限於某一範圍內。

三、 運動控制系統的強健性質(Robust)是指系統對週遭事件的反應所造成的時間延遲現象，並不會對控制系統的性能造成無法容忍的影響，但傳統設計方法並無法評估此種特性。

四、 不同的排程演算法對上面所述問題所造成的影響

這些問題通常又和所使用的即時作業系統之特性有關，這些特性包括：排程演算法(Scheduling Algorithm)、信號傳送特徵(Signal feature)與等待時間(Latency)，另外也可能是額外的記憶體需求。

綜合上面所言，可將這些問題產生的原因歸納爲：

一、 缺乏完整的運動控制系統設計過程
從分析最初需求到建立性能符合要求的系統，完整的過程包括了發展規格、設計演算法、硬體設計、軟體設計、產品原形化、以及最後的產品測

試與驗證階段，這些階段所需完成的分析與設計內容並沒有被清楚定義。

二、 設計過程各步驟間存在的設計不連續性(Discontinuity)，這些不連續性包括：

(一) 設計規格的不連續性

在訂定運動控制系統設計規格時，較少考慮到軟體和硬體的架構對整體性能的影響，因此除了缺少對軟、硬體相關的發展規格外，傳統設計方法也無法得知軟、硬體性能對整體性能的影響。完整的設計規格需加入這方面的考慮。

(二) 設計流程的不連續性

除了設計規格的不連續性外，各自獨立的設計流程因沒有標準的步驟和驗證方法，加上各階段都缺少與前、後階段相互整合的成員與工具，致使其缺乏整體性，因此各流程往往無法充分且正確的反應前一步驟的發展成果。尤其是軟硬體通常有自己的發展規格，因此無法確保整合時的正確性與相容性，並經常造成不同工具間的錯誤轉譯。

三、 因爲運動控制系統的運作特性並缺乏適當的建模(Modeling)工具，很難建立系統真正的運作模型，因此無法具體描述設計成果並進行離線(Off-line)分析。

上述的問題將拉長產品的設計週期，並且任何在整合軟、硬體時所發生的問題都可能成爲整個發展計劃中的致命傷，所以我們必須重新思考傳統的設計流程。可靠的研發方式必須同時從觀念到軟硬體的各個面向都納入設計考量，過去的發展環境，當研發工程師選擇新的發展流程時，即使在相同的設計階段，此時所使用的研發工具通常只針對特殊設計領域，而不能應用於整個設計流程當中，並與其他工具互相整合，有效率的研發過程必須揚棄過去的做法。

參、軟、硬體整合技術的發展趨勢

爲了解決前面所談及的問題，與滿足此時的需求，急需有一致性的運動控制系統設計規格與標準的系統設計程序和整合環境。也就是說運動控制系統的設計過程包括了從設計概念分析、建立模型、系統模擬、產生程式碼、到內嵌控制系統的快速原型化等。而想要達成理想的發展週期，有賴於一個直覺式的、經得起各種考驗的系統層級發展環境。強調互動性與整合性的發展環境，能完全整合運算邏輯、C 程式碼、傳統即時控制系統軟體設計、與電子設計自動化(EDA)等工

具。

另外，以運動控制系統的發展層面而言，除了需考慮生產成本外，其他常見的設計需求尤其是嵌入式運動控制系統尚包括了[2]：

一、 運算處理能力

他代表了要完成工作所需的運算處理能力的量，一般我們用來比較運算處理能力的方法是測量 MIPS 值(Millions of Instructions Per Second，每秒可執行百萬個指令)，如果有兩個測量值分別爲 25MIPS 及 40MIPS 的處理器，我們就認爲後者的運算處理能力比較強。除此之外，處理器的其他重要特徵也要列入考慮，其中之一就是暫存器的寬度。當系統發展完成時，利用某些工具我們很容易可獲得某些服務的指令長度與可容許的最長執行時間，利用這兩項資料即可獲得運動控制系統所需的運算處理能力。

二、 記憶體

他代表了存放可執行碼與其處理資料所需的記憶體數量，尤其是 DSP-Based 的運動控制系統，硬體設計者通常必須在先前做出最佳預測，並且隨著軟體開發作號增加或減少記憶體的準備。而所需的記憶體數量將影響到記憶體的選擇。一般而言處理器暫存器長度通常決定了他可以存取的記憶體在數量上的上限。

三、 研發成本

他代表了硬體與軟體在設計過程所花的成本，對於硬體而言他通常是固定的成本，但對軟體而言則必須包含往後維護的支出，所以他的重點反而不是金錢，尤其是對大量生產的產品來說。

四、 生產數量

在生產成本與研發成本之間的權衡，通常受預期可生產即銷售數量的影響最深。例如要開發一個產量和市場皆小的產品，就要考慮製造此類產品的可行性。

五、 預期壽命

一個必須能持續運作多久，這將影響到設計決策的各個層面，就硬體元件的選取到系統在研發及生產上所需花費的成本，這些決策都會受到產品壽命的影響。

六、 可靠度

最終產品必須要可靠到何種程度，也會影響到硬體元件的選取及研發、生產成本，

這些額外的需求間接影響了所使用的工具與發展流程，因此新一代運動控制系統設計方法的主要發展概念與目標包括：

一、 軟、硬體整合技術

傳統的 EDA 解決方案對硬體執行層面的知識需有相當嚴格的要求，即使只是一個初期的系統設計工程師，也必須有深厚的背景。因此將造成控制演算法設計工程師與硬體設計工程師之間的藩籬，如果此設計週期中還包含軟體設計的部分，情況就更顯得更加複雜。但此種整合式硬體設計(Integrated Hardware Design)和 EDA 藉用整合式的設計概念，的確讓各自分離的工具在共同的產品模式上從事實體的硬體設計。這種設計方法確實達到了一定程度的進步，因爲它同時採用了視覺化圖形區塊(Visual Block-diagram)與標準化硬體描述語言(Standardized Hardware Description Language)。但這種 Top-down EDA 的設計方式，通常只能滿足硬體設計師的設計需求。但隨著科技的進步，軟體日益複雜，使軟體已快速成爲許多通訊產品、多媒體產品、與其他 DSP 應用產品設計週期中的主導角色，所以軟體設計所花費的時間與軟體工程師的生產力，才是決定整個設計週期的關鍵因素。

而就軟體設計的層面而言，則以系統層級設計工程師與 DSP 軟體實行設計工程師之間的隔閡最爲惱人。就運動控制系統軟體或相關控制演算法設計者而言，通常將自行設計、手工編碼的 C 程式碼，用編譯的方式或手工轉譯的方式轉成組合語言，來模擬、定義此演算法在某特定儀器設備上所可能表現的行爲模式。也因爲此軟體設計方式通常使用非物件化語法，使得這些 C 程式碼難以理解且較難重複利用，如此使得工程師們無法於設計與測試階段，快速輕易地展開演算法或調整設計，因此雖然這些手寫的 C 程式碼看起來能夠符合演算法設計的需求，但它其實已經限制了運動控制系統軟體的發展。

一個新的演算法編寫爲 C 或 Ada 程式碼等執行碼需要花上相當的時間，使用圖形區塊模式（Block-diagram）表示法的系統層級設計方式，就能讓軟體設計工程師只要在短時間內，完成繁複的設計與測試的工作。在理想的控制系統軟體設計模擬環境裡，必須同時涵蓋由下往上(Upstream，上溯至演算法設計)與由上往下(Downstream，往下至內嵌式運動控制系統軟體設計工具)兩種系統設計方式，如此才能簡易地結合使用者自訂之特殊 C 程式碼、提供完美的共同模擬(Co-simulation)、除錯介面，與全自動地將模型轉換成即時的可執行程式碼功能。可將模型自動轉換成可執行程式碼的另一個好處，在於設計結果不會因程式碼撰寫人員程度的不同而影響設計品質，另外也可大量縮短往後軟體的維護時間。

綜合上面所言我們除了需解決軟、硬體工程師各自面對的問題外，也需

整合兩者的發展結果。

二、 提供整合的發展環境

整合的發展環境必須能讓系統發展者們能夠同步地從事特定領域的研發，並擴展過去的研究成果。這個單一、互動的系統層級設計環境裡，結合了演算法設計工具、圖形區塊模擬工具、即時程式碼產生器、與各式分析工具，環境。

此整合的發展環境提供快速、準確、支援多種模擬型態(例如事件導向與時間導向)，並支援多種元件類型（如：軟體與能處理類比、數位、混合訊號的硬體）與提供直覺式使用者介面，讓使用者容易理解其背後的數學演算法與行爲模式的設計模型。

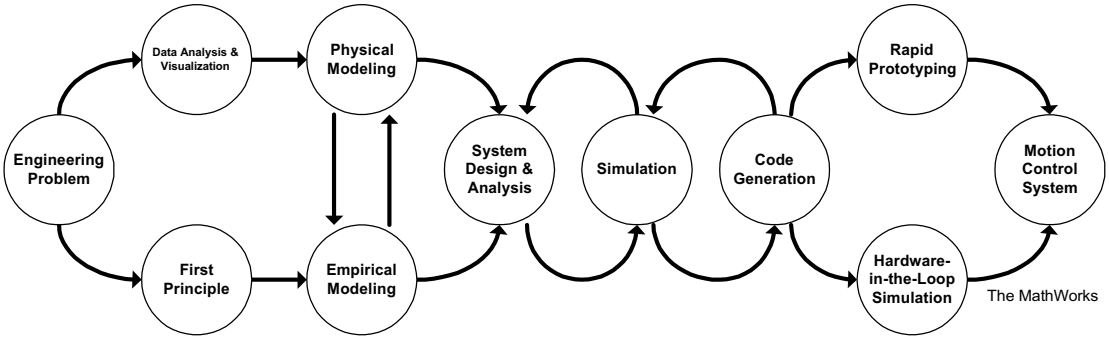
最後，一個良好的發展環境必須能夠儲存之前的設計成果，也就是運用程式設計介面，從函式庫裡呼叫各種過去編寫的程式碼。

肆、新一代的運動控制系統發展環境介紹

新一代的運動控制系統完整設計過程可區分爲下列步驟或階段(Phase)[1]：

一、 需求分析
二、 測量資料分析、視覺化與利用經驗法則建立系統模型
三、 提出初步解決方案並利用基本理論建立系統模型
四、 綜合二、三項結論已獲得系統模型並進行控制系統設計與分析
五、 系統模擬
六、 程式碼撰寫
七、 硬體迴圈模擬(Hardware-in-the-loop Simulation)
八、 快速原形化(Rapid Prototyping)

上述步驟或階段可使用下圖表示，這些步驟或階段並非全然獨立，此圖將可看出各步驟間的關係。



爲了滿足上述各階段的需求，經整合的新一代運動控制系統設計環境需包括資料分析、資料視覺化、演算法設計、模型模擬、產生程式碼等工具，這些工具依照功能的不同可區分爲四類，包括了：

一、 提供直覺式語言及高科技運算環境或平台，此平台支援核心數學運算能力並配合下面將提及進階的圖形工具，以便完成資料分析、資料視覺化、以及演算法與應用程式的設計。

二、 提供分析、模擬真實世界與動態系統的模型化設計環境，對建立在上述運算環境或平台上的應用或程式設計，提供圖塊式的介面工具。
整合環境將提供使用於建模、模擬與分析動態系統的互動式工具，此種工具將可建立系統的區塊圖模擬動態系統，並分析、驗證系統的性能以提昇原來的設計品質。
利用與運算環境或平台緊密的結合，將提供系統發展者獲得更寬廣的分析與設計視野。配合下面將提及的事件驅動系統圖形化模擬環境，將使得此類工具在發展運動控制系統時扮演不可或缺的交色

三、 提供模型化與設計事件驅動系統的圖形化模擬環境，尤其是在設計內嵌式運動控制系統與原型化邏輯驗證時，此種設計與模擬環境將有效縮短發展時程。
這類圖形化的環境通常以 Finite State Machine 爲基礎，使用狀態(State)和狀態轉移(Transition)定義系統，因此系統可使用圖形化的狀態轉移圖加以表示。目前功能完整的狀態轉移圖尚具備階層式與平行式架構，也包括自動連結與歷史紀錄的功能。
使用此類設計與模擬環境來發展事件驅動系統並不需具備 Finite State Machine 的觀念，配合下面將提及的程式碼產生器，將可有效率的自系統模型產生 C 程式碼，此種強大的功能將不只侷限應用於運動控制系統，在自動化生產、航太與通訊應用中的嵌入式產品經常使用此類工具。

四、 能夠在進行系統原型化、硬體迴圈模擬、與內嵌式系統的應用時，直接由圖表中，產生出爲您定作的 C 程式碼與或其他程式碼。
程式碼產生器能直接由圖形化的設計成果產生經最佳化、具可攜性的 ANSI C 程式碼。此外也可自動建立可在即時系統中執行的程式，或只是單獨模擬於非即時環境中。這些程式碼配合某些即時作業系統可運作於 PC、DSPs 或其他微控制器上，如此可滿足快速系統原形化

的目標。而產生的程式碼可應用於包括離散系統、連續系統或混合系統中。

某些程式碼產生器可利用動態機所定義的模型產生程式碼，而 Ada 程式碼產生器則能產生 Ada 83 與 Ada 95 原始碼；有些設計環境也提供嵌入系統程式碼產生器，可自動產生有效率、可嵌入的 ANSI C 原始碼。新一代的運動控制系統發展環境皆提供完整的程式碼產生器，以滿足各種不同的需求。

伍、結語

新一代的運動控制系統發展環境利用標準化的硬體與軟體描述語言，有效整合硬體與軟體的發展過程。這些描述語言獨立於發展平台與發展工具，因此只要是支援並符合這些描述語言的規定，並不限定何種發展工具，如此可增加系統發展工具選擇的多樣性。

這些硬體與軟體描述語言爲非程序化(Nonprocedural)程式語言，程序化程式語言即爲一般的程式語言，例如 C 或 PASCAL，用來指示處理器(Processor)依照所敘述的動作依序執行；非程序化程式語言則是用來定義要求的結果，再利用直譯器或編譯器來產生程式碼，此種語言爲架構運動控制系統 I-CASE 工具的基礎。完整的運動控制系統 I-CASE 工具除了包括了分析、設計、模擬工具與程式碼產生器外，更先進的發展環境尙包括儲存庫(Repository)與推理核心(Inference Engine)，如此除了可保存設計成果外，具備人工智慧的推理核心將可將降低在設計過程中所產生的人爲錯誤。

利用經圖形化的模擬機制與系統設計結果表示法，配合硬體迴圈模擬將可有效降低運動控制系統的發展時程與成本，而標準化的程式碼產生器將可消除因人爲因素，所造成軟體品質不一的現象。

國內鈦思科技[1]所代理的 The MathWorks 整合式設計環境爲一個系統層級設計工具，它提供了一個共同的設計環境，讓工程師們在發展運動控制系統時能夠同步地從事各領域的研發，並擴展過去的研究成果。這些工具包括了演算法設計工具、圖形區塊模擬工具、即時程式碼產生器、與各式分析工具，足以滿足設計師們的嚴格需求；此外，此系統層級設計環境並提供事件導向模擬、延伸的 DSP 與通訊設計工具箱、函式庫、開放的應用程式介面（API）等，在發展運動控制系統的過程中提供了整個設計團隊在從事系統層級設計的最佳環境，以我們的實際使用經驗，利用此發展環境的確縮短了產品的發展時程，並增加了產品的可靠度。有興趣的讀者可參考鈦思科技的網站，以獲得更進一步的資訊。

參考資料

[1] The MathWorks co.，http://www.mathworks.com；國內代理商：鈦思科技 http://www.terasoft.com.tw
[2] Michael Barr，”Programming Embedded Systems in C and C++”，O’REILLY