MIAT 系統設計與硬體高階合成方法論

陳慶瀚
義守大學電機工程學系
pierre@isu.edu.tw

摘要

由於缺乏一致性的、系統化的設計方法論，現有的資訊—電子產品研發工作常需大幅仰賴系統工程師的個別設計經驗，因此使得產品開發的設計時間、成本、可靠度和重覆使用性都難以掌握，而且團隊的研發也難以在嚴謹的、演繹的基礎(infrastructure)上進行分工、協同的設計。基於這些考量，我們提出一個名為 MIAT 的系統設計與高階合成方法論，此一方法論先採用階層式、模組化的功能架構設計方法—IDEF0。一個系統被由粗到細(coarse to fine)分解成一組獨立的功能模組，以便設計團隊得以進行協同設計；隨後針對每一個功能獨立的模組，我們使用 GRAFCET 圖形化工具建立其離散事件模型，也就是行為建模(behavior modeling)。第三個階段，根據一組合成的法則，所有 GRAFCET 模型得以轉譯為 VHDL 硬體描述語言的控制器硬體架構，最後結合一些通用的組合邏輯建構單元，便可實現完整的系統硬體高階合成。MIAT 方法論已經被應用在許多系統設計的問題上，其中包括了工具機控制器、模糊控制器、FIR 濾波器等系統的設計與電路合成；同時也成功地實踐在指紋辨識系統單晶片、加密晶片等商業化產品開發案例。

1.前言

近年來，以電腦、通信和消費性電子產品為首的 3C 產業快速發展，使得相關產品生命週期不斷縮短，導致產品的系統設計方法和技術也必須隨著不斷更替和改進，其中最明顯的例子就是系統單晶片(System-On-A-Chip)、快速原型化(Rapid Prototyping)、快速上市(Rapid Time-to-Market)的發展趨勢。

由於缺乏一致性的、系統化的設計方法論，現有的系統設計方式必須大幅仰賴系統工程師的個別設計經驗，因此使得產品開發的設計時間、成本、可靠度和重覆使用性都難以掌握。一旦系統設計完成後，還需進行系統實作(implementation)，以 3C 產業追求低成本、高性能、微型化的目標來看，以 IC 為目標的系統單晶片顯然是最具競爭力的技術解決方案。在 IC 產業，EDA（電子設計自動化）是 IC 設計的一致目標。目前在實體層級（Layout，半導體物性）的設計自動化程度已經很高，但在高階（high-level）系統層級的自動化程度仍是學術界和產業界共同努力的目標。

在各式的消費性電子產品的產品開發過程，控制器(controller)常扮演著最重要的關鍵角色之一，由於各種產品的功能、規格互異，因此絕大多數的控制器實現均由有經驗的電子工程師來進行，電子工程師的設計工作係依照資訊工程師的系統設計結果來進行，資訊工程師則憑藉系統工程師的系統分析和規格定義才能夠

從事系統的功能和限制。此一工作流程呈現明確、互補的分工，但也具有以下的限制：

—上游的產品設計、中游的資訊軟體設計和下游的電子設計三個領域需要有密切的溝通配合，但在現實的工業設計和製造環境中，卻顯示這樣的溝通配合並非總是可能的。特別在以中小企業為主體的台灣工業界，組織規模和知識管理的模式都不利於發展此種精細分工再整合的概念。

—上述設計流程是循環式的。如果有下游的設計環節發現問題，則可能需要檢討、甚至更動上游的設計，再重新投入新一輪的設計。如此一來，產品開發週期勢必大幅延宕，同時也導致成本增加。

—在這樣的設計流程中，由於缺乏一致性的、系統化的設計方法論 (Design methodology)，個別的設計者僅能以其專業的技能和經驗來解決問題，但卻難以累積形式化的設計知識，限制了知識技能重覆使用的可能性。

針對上述問題，本文提出一個系統設計與硬體高階合成的方法論，將不同的技術、工具和方法，整合在一個階層式、模組化和自動化的設計流程。藉由此一設計方法論，將有助於 3C 產品的設計自動化和快速原型化目標，以有效提升競爭力。

我們的方法論是以系統分析結果為出發點，首先進行系統功能架構的階層化和模組化，在此一階段，IDEF0[05][06][07]被採用作為主要的圖形設計工具。隨後針對每一個功能獨立的模組，我們使用 GRAFCET[08][09][10][11]建立其演算法模型，也就是行為建模(behavior modeling)，每一個 GRAFCET 模型都對應一個離散事件控制器。第三個階段，根據一組合成的法則，所有 GRAFCET 模型得以轉譯為 VHDL 硬體描述語言的控制器硬體架構，最後結合一些通用的組合邏輯建構單元，便可實現完整的系統硬體高階合成。其架構與流程如圖 1 所示:

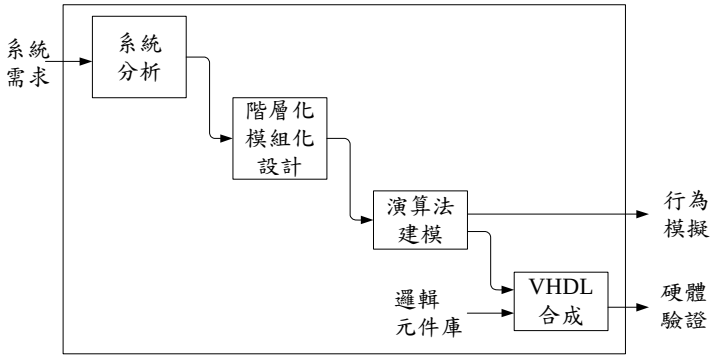


圖 1、系統設計與高階合成方法論

2.系統功能架構的階層式、模組化設計

由於產業界對系統種類的需求愈來愈多樣化、複雜化，因此對於設計一個複雜系統時，能夠針對系統的行為特性做階層化和模組化的分割，則有利於研發團隊進行多人的同時設計，具有縮短系統的開發時程、只需考慮各模組內的功能與行為特性、簡化了整個系統問題等優點。基於上述的目的，我們採用了

IDEF(Integrated Computer-Aided Manufacturing (ICAM) DEFinition) [12]中的IDEF0 技術來做系統的階層化和模組化，此技術以簡單圖形、文字描述與結構化分析的方式來表達出複雜系統的內部包含的功能模組(functional module)以及模組之間的連結關係(relationship)。進行系統模組化分割後，每個模組均視為一個演算流程的序向器(sequencer)，至於演算法的行為建模(behavior modeling)則交由第二階段的 GRAFCET 來進行。

IDEF 方法是由美國空軍於 1978 年至 1983 年執行 ICAM(Integrated Computer Aided Manufacturing)計畫發展而來。主要是使用結構化的繪圖形式來描述複雜的系統，以強化系統整合人員彼此的溝通。IDEF0 的基本概念源自於 Douglas T. Ross 在 1970 年代中期所提出的結構化分析及設計技術(Structured Analysis And Design Technique, SADT)[13][14]，目前也是美國國家標準及技術局(NIST)的第 FIPS183 號標準[15]，並且納入 ISO9000 規範之中。

IDEF0 以結構化、圖形化的方式將複雜系統功能間的關係與和功能有關的物件(Objects)、資訊(Information)表達出來[16]，以改善一般語言表達能力的不足。這種方法以結構化層級的方式描述系統流程，同時清楚的說明每一流程所需之輸入、輸出、控制、設備，使我們瞭解系統的功能與流程。IDEF0 除了繼承了 SADT 的圖形化模型表示法的優點外，更加上了較嚴謹的模式建構規範，因此極適合於團隊分工設計工作。

IDEF0 Model的建立是由功能方塊(Function Block)和連接這些方塊的箭頭(Arrows)所組成[17]，每一格方塊代表一個系統的活動(Activity)，方塊的名稱必須具有動詞性質[18]，用以描述系統中的過程(Process)、作業(Operation)或轉換(Transportation)如圖2 所示。箭頭是用來連結功能方塊，表示功能方塊之間的關係和描述執行系統功能所需之資訊和物件，箭頭的名稱必須具有名詞性質。箭頭資料可分成四類：輸入(Input)、輸出(Output)、控制(Control) 和機制(Mechanism)。輸入從方塊的左方進入，描述執行其功能所必須之資料或物件，例如原料、參考資料、輸入資料。輸出從方塊的左方離開，描述執行完功能後所產生的資料或物件，例如產品。控制從方塊上方進入，描述執行其功能過程中的限制條件或政策與原則，例如國際標準或交期。機制從方塊的下方進入，描述執行其功能所需的人員、設備、工具或其他資源。

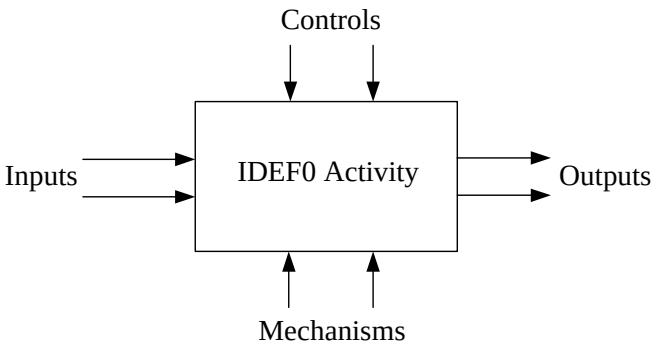


圖 2、IDEF0 基本模組功能方塊與箭頭意義

此外 IDEF0 有功能階層化分解(Decomposition)的特性，可視需要透過結構化分解將複雜系統往下展開成獨立的子功能模組結構，以便逐步描述出更細部的功能模型如圖 3 所示，以此簡化複雜系統設計的維護和群體協同(collective co-operative)設計工作。

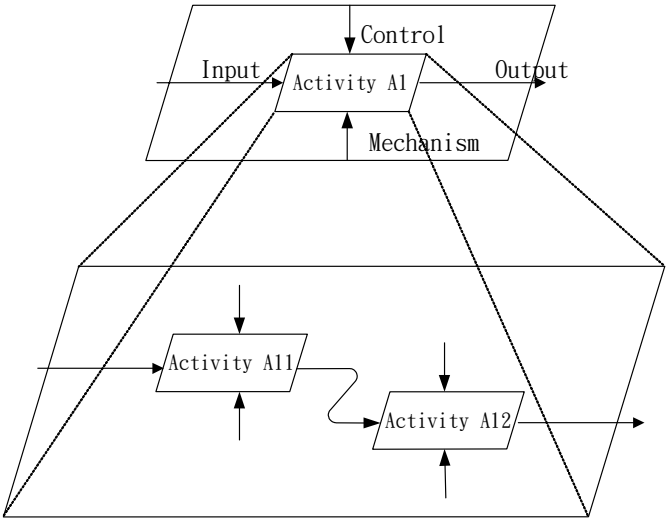


圖 3、IDEF0 階層化架構

每一個 IDEF0 模組皆可視為獨立的離散事件系統(Discrete Event System, DES)，系統的行為可以用一組循序演變的離散狀態表示，或者以演算法來描述，我們採用 GRAFCET 作為離散事件系統的建模工具。

3. 離散事件系統建模

一個能夠表達出複雜離散事件系統的輸入/輸出與狀態轉移關係的方法，必須符合下列的需求：

（1）能描述出離散事件系統的大量且循序發生的狀態。

（2）必須考慮事件平行發生(Concurrent)的情況，且能以簡單明瞭的方式表達。

（3）通常一個系統的狀態只會受到幾個輸入的影響，且只有一些輸出被改變，因此只描述因輸入改變而產生的局部行為。

（4）可以清楚的瞭解輸入與輸出的關係

針對上列的需求，我們採用 GRAFCET 模型作為離散事件系統的建模工具。GRAFCET 模型是由一個結合法國學界和工業界的團體所定義出來，1987 年成為國際標準的方法[19]。它的基本原理和元素是由 Petri nets[20]發展而來，用來描述循序行為與平行發生情況的一種圖形化模型。

3.1 GRAFCET 的基本構件

GRAFCET 圖形由三種構件所組成：

—步驟(step)

—轉移條件(transition)

—有向性的連結(directed connection)

一個 GRAFCET 至少包含一個狀態和一個轉移條件以上，連結時由狀態連結

到轉移條件或是由轉移條件連結到狀態。

步驟以一個方塊圖形表示。它有三種狀態，一個系統的初始步驟被表示為一雙方塊，如圖 4(a)；當步驟是活動的 (active) ，如 4 (b)，則在步驟中加入一個 token 標記；否則步驟就是閒置(inactive)的，如 4 (c)。

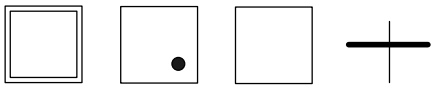


圖 4、步驟：(a)初始步驟(b)活動的步驟(c)閒置的步驟(d)轉移條件

轉移條件的表示符號為一橫槓，如 4(d)所示。轉移條件成立與否與接受 R_i(Receptivity)有關，定義為

$$R_i \in \{0,1\}$$

R_i 可以是 GRAFCET 輸入變數或是內部狀態的邏輯條件。

3.2 GRAFCET 的結構

一個 GRAFCET 最基本的結構是如圖 5 所示的建構單元(Building Block)。一個步驟 X0 經由步驟轉移條件 R_0 連接著另一個步驟 Xn，它表示條件 R_0 成立之前必須等待 X0 變成活動才能轉移狀態，狀態轉移後 Xn 將變成活動，而 X0 則變成閒置，如圖 5(a)。

0 1

Initial Active

(a) (b)

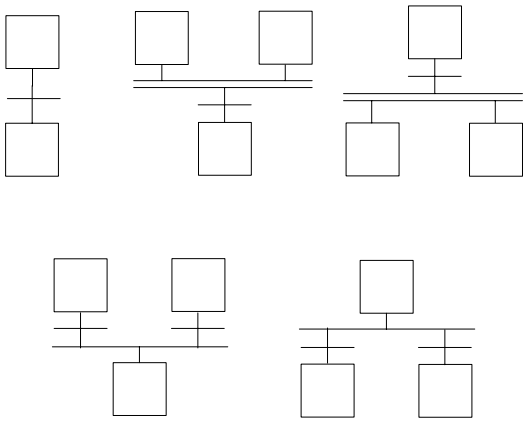


圖 5、GRAFCET 的建構單元

GRAFCET 提供一個平行架構的形式來描述事件平行發生的行為，如圖 5(b)和(c)。在圖 5(b)中，當兩個步驟{X0、X1}一起進入同一轉移條件(R_0)，這時就必須在步驟轉移之前加入一雙橫槓。它意味著在轉移條件 R_0 成立之前必須等待兩個步驟 X0、X1 皆變成活動狀態，才會發生狀態轉移，Xn 變成活動，而 X0、X1 同步變成閒置，這種模式稱為 Convergence AND。在圖 5(c)中，當轉移條件 R_0 後接有兩個步驟{X1、Xn}，這時就必須在步驟轉移之後加入一雙橫槓，表示轉移 R_0 成立之後兩個步驟{X1、Xn}將同時變成活動，X0 變成閒置，這種模式稱為 Divergence AND。

至於條件式的流程控制(Conditional Flow Control)，GRAFCET 提供了 Convergence OR 和 Divergence OR 兩種表達形式，如圖 5(d)和圖 5(e)。當有兩個轉移條件 R_0、R_1 連結到同一步驟 Xn，如圖 5 (d)，此即 Convergence OR 流程，它表示步驟 Xn 要轉變成活動只要任一轉移條件 R_0 或 R_1 成立即可。另外在圖 5(e)中，當一個步驟 X0 後接有兩個{R_0、R_1}或多個轉移條件時，表示步驟 X0 要轉變成閒置只要 R_0、R_1 任一轉移條件成立即可，此為 Divergence OR 流程。

圖 6 是一個離散事件系統的 GRAFCET 狀態演化實例。圖中的 GRAFCET 具有 4 個步驟與 3 個轉移條件，由(a)→(b)→(c)→(a)表現 GRAFCET 的狀態轉移情形。當系統一開始，系統狀態位於初始步驟 X0，如圖 6(a)，此時只有轉移條件 R_0 成立時才產生狀態轉移。轉移發生後，轉移條件 R_0 之前所連接的步驟 X0 將轉變成閒置，而轉移之後所連接的所有步驟 X1、X2 將轉變成活動。因此當轉移條件 R_0 成立時活動步驟由{X0}轉變成{X1, X2}，即圖 6(b)。此時，當轉移條件 R_1 成立時，活動步驟轉變成步驟{X1, X3}，如圖 6(c)。當轉移條件 R_2 成立時將會回到初始步驟 X0。

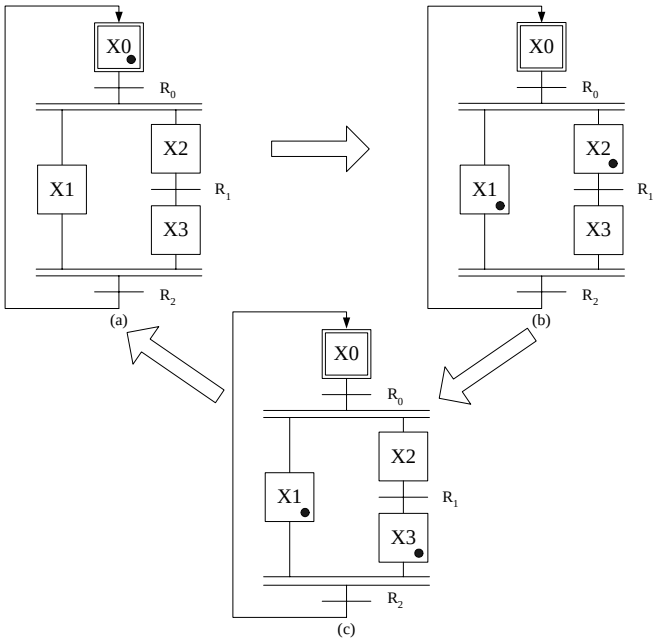


圖 6、GRAFCET 的狀態演化

4. VHDL 電路合成

VHDL 是由美國國防部(The United States Department Of Defense)於 1980 年提出一個名為極高速積體電路(Very High Speed Integrated Circuit, VHSIC)計畫而來，1982 年改良後的電路描述方式 VHDL (VHSIC Hardware Description Language)產生，1987 年成為國際電機電子工程協會(International Electrical & Electronic Engineering, IEEE)標準，即 IEEE1076 標準。1988 年，美國國防部規定所有官方的 ASIC 設計都必須以 VHDL 為設計描述語言。1993 增修為 IEEE1164 標準，1996 年，IEEE 再將電路合成的標準程式與規格加入至 VHDL 硬體描述語言之中，稱

之為 IEEE1076.3 標準。

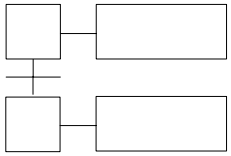
近年來，由於應用系統功能複雜度快速攀升，VHDL 因為可以直接針對電路的功能性(Functionalities)而非實體(Physical Device)進行設計，讓複雜的電路設計可以透過高階程式語言編譯方式，快速達成電路合成的目標，因此受到廣泛的採用。

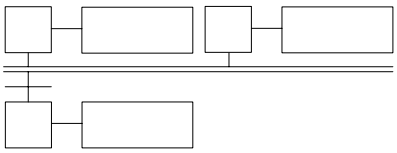
一個基本的 VHDL 程式模組包含了單體(Entity)和架構(Architecture)兩個主要部分。單體在 VHDL 語言中只是用來描述階層式方塊的介面(Interface)，即描述電路設計中的輸出與輸入訊號的宣告。架構描述(Architecture Description)則是用來描述電路單體的內部結構或行為。

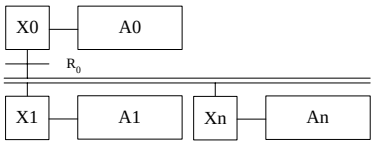
我們在前一階段建立了離散事件系統的建模方法。在此，我們將規劃一系列的 VHDL 合成規則，以便將 GRAFCET 模型中的每一個建構單元轉譯為 VHDL 程式碼，此一轉換過程是線性的、直覺的、且是自動化的。

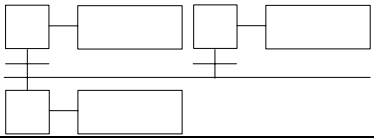
4.1 GRAFCET 建構單元的 VHDL 合成

圖 7 列出五種 GRAFCET 建構單元的 VHDL 程式碼的合成規則。我們在每一個步驟結合一個致動方塊(action block)，例如步驟 X0 對應致動方塊 A0，步驟 Xn 對應致動單元 An 等。這些致動單元可以是簡單的邏輯設定，或是基本的組合邏輯運算，也可以是 GRAFCET 子模組。

Primitive building block
 
	`if X0='1' and R0='1' then`
	`X0<='0'; Xn<='1';`
	`end if;`
	`A0 <= X0 ;`
	`An <= Xn ;`

Convergence AND
 
	`if X0='1' and X1='1' and`
	`R0='1' then X0<='0'; X1<='0';`
	`Xn<='1';`
	`end if;`
	`A0 <= X0 ;`
	`A1 <= X1 ;`
	`An <= Xn ;`

Divergence AND
 
	`if X0='1' and R0='1' then`
	`X0<='0'; X1 <='1'; Xn<='1';`
	`end if;`
	`A0 <= X0 ;`
	`A1 <= X1 ;`
	`An <= Xn ;`

Convergence OR
 
	`if X0='1' and R0='1' then`
	`X0<='0'; Xn <='1';`
	`elsif X1='1' and R1='1' then`
	`X1<='0'; Xn <='1';`
	`end if;`
	`A0 <= X0 ; `

	A1 <= X1 ; An <= Xn ;

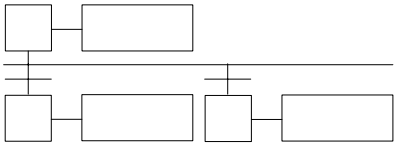
Divergence OR
 
	if X0='1' and R0='1' then
	X0<='0'; X1 <='1';
	elsif X0='1' and R1='1' then
	X0<='0'; Xn <='1';
	end if;
	A0 <= X0 ;
	A1 <= X1 ;
	An <= Xn ;

圖 7、GRAFCET 建構單元的 VHDL 程式碼合成規則

4.2 GRAFCET 模型的 VHDL 合成範例

我們以一個 GRAFCET 模型為例來展示完整的 VHDL 合成範例。如圖 8，首先列出系統的輸入輸出訊號：{X0, X1, X2, X3}，X0 是初始步驟，X1, X2, X3 分別對應致動單元 A1,A2,A3。致動單元的邏輯值為 1 代表該致動單元被啟動或開始執行運算。另外我們從圖中也可以列出三個轉移條件：R0,R1,R2。因此我們可以宣告電路的輸入/輸出介面規格如下：

```
entity Example is
port(
  clk,rst:in std_logic;
  A1: out std_logic;
  A2: out std_logic;
  A3: out std_logic;
  S0: in std_logic;
  S1: in std_logic;
  S2: in std_logic
 );
end Example;
```

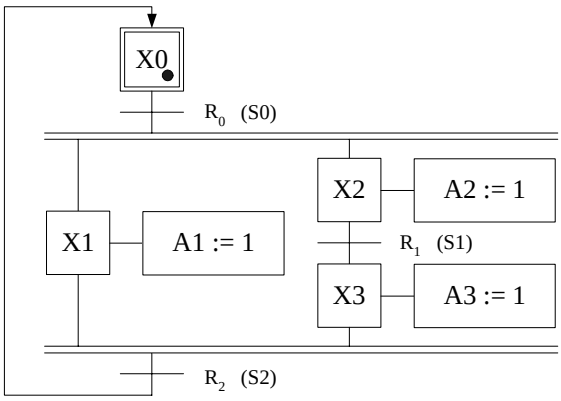


圖 8、GRAFCET 模型範例

至於系統的行為，可綜合圖 7 的合成規則，得到以下的的電路架構描述：

```
ARCHITECTURE action OF Example IS
  signal X0,X1,X2,X3: std_logic;
begin
  process(clk)
  begin
```

```
    if rst='0' then
      X0<='1';
      X1<='0';
      X2<='0';
      X3<='0';
    elsif clk'event and clk='1' then
      if X0='1' and S0='1' then
         X0<='0'; X1<='1'; X2<='1';
      elsif X2='1' and S1='1' then
         X2<='0'; X3<='1';
      elsif X1='1' and X3='1' and S2='1' then
         X1<='0'; X3<='0'; X0<='1';
      end if;
    end if;
    A1<=X1;
    A2<=X2;
    A3<=X3;
  end process;
end action;
```

5. 方法論應用實例

MIAT 系統設計方法論已經成功的應用在許多系統設計問題上，其中戴[]用以設計了一個結合 PI 控制與離散事件控制的鑽孔機控制器，並以 FPGA 實現其電路，最後驗證了硬體的可用性(validity)，其結果參見圖 9。

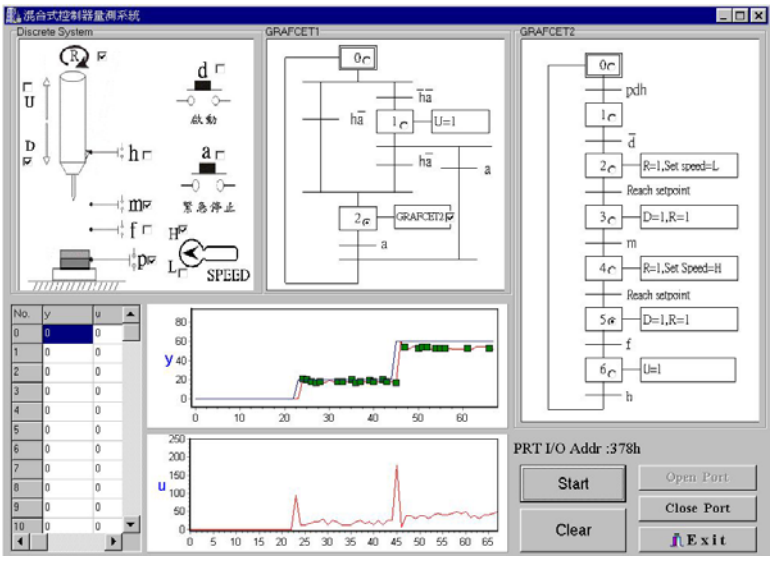


圖 9、MIAT 方法論應用在鑽孔機控制器設計與高階合成

此外 Chen 等[]以 MIAT 方法論設計了模糊控制器，其特色在於具有高度結構化和平行處理能力，並且針對不同的性能和成本考量，能夠輕易合成新的硬體電路，可符合快速原型化的工業需求。圖 10 展示模糊控制器的 IDEF0 的系統架構，圖 11 則展示其合成的電路方塊圖(block diagram)。

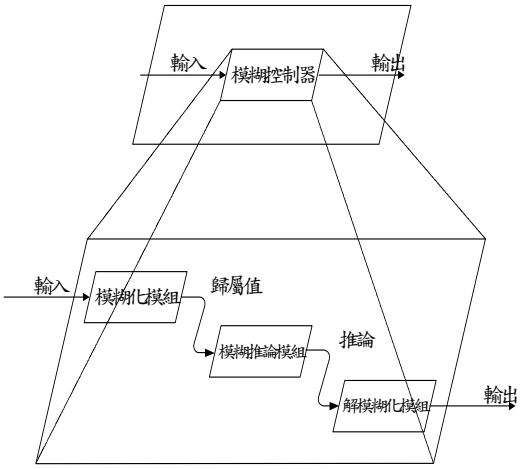


圖 10、模糊控制器的 IDEF0 系統架構

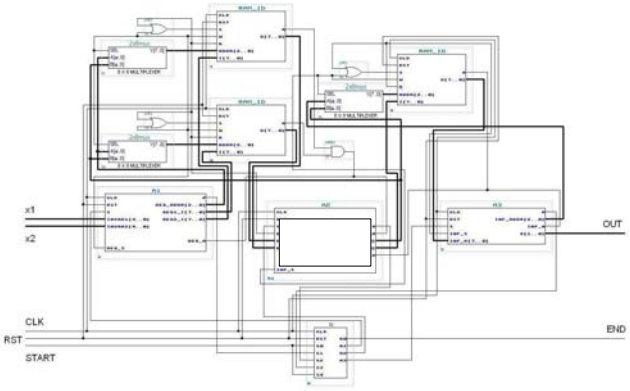
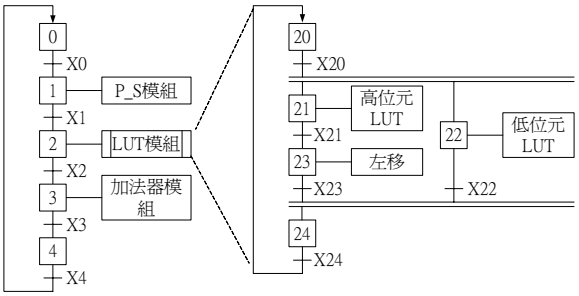
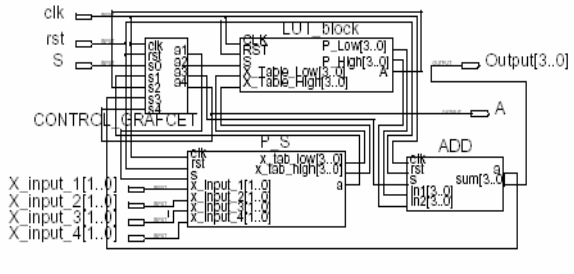


圖 11、合成的模糊控制器電路方塊圖

另一個 MIAT 方法論的應用實例是 FIR 濾波器的硬體電路實現。圖 12 是 FIR 濾波器的離散事件系統建模和電路合成方塊圖，該系統的主要目的是設計一個可覆用的(reusable)快速數位濾波器(DSP)核心元件。





Fuzzification Module

Infe
En

圖 12、FIR 濾波器的離散事件系統建模和合成電路方塊圖

6.結論

基於 Top-Down 設計範式(paradigm)和高階合成的目的，我們提出一個名為 MIAT 的系統設計方法論。此一方法論先採用階層式、模組化的功能架構設計方法—IDEF0。一個系統被由粗到細(coarse to fine)分解成一組獨立的功能模組，以便設計團隊得以進行協同設計；隨後針對每一個功能獨立的模組，我們使用 GRAFCET 圖形化工具建立其離散事件模型，也就是行為建模(behavior modeling)。第三個階段，根據一組合成的法則，所有 GRAFCET 模型得以轉譯為 VHDL 硬體描述語言的控制器硬體架構，最後結合一些通用的組合邏輯建構單元，便可實現完整的系統硬體高階合成。

此一方法論的優點在於將不同的方法、技術、工具和模型，整合成一個嚴謹的、完整的設計程序。它有利於研發團隊從事群體的、分工式的系統設計工作。此外，基於 GRAFCET 的行為建模，提供了傳統的演算法設計者一種(比 Petri Net)更簡潔、(比 Finite State Machine)更強大(powerful)的工具來表示系統的離散事件模型，除了可處理典型的順序邏輯(sequential logic)，更可處理平行發生的事件邏輯(concurrent event logic)，因此可獲致最高性能的硬體架構設計。

最後，根據我們所建立的 VHDL 合成規則，系統的所有的模組皆可快速的轉譯為 VHDL 形式的電路，藉由一些市售的工具(commercial utilities)，可將這些電路燒錄在適當的元件如 FPGA 再進行硬體驗證(hardware verification)，以此達到快速原型化的目標。

MIAT 方法論已經應用在許多系統設計的問題上，其中包括了工具機控制器、模糊控制器、FIR 濾波器等系統的設計與電路合成；同時也成功地實踐在指紋辨識系統單晶片、加密晶片等商業化產品開發案例。