行政院國家科學委員會專題研究計畫　成果報告

針對系統晶片（SoC）架構工作排程方法之探討及其模擬評估環境之研製

計畫類別：個別型計畫
計畫編號：NSC91-2213-E-009-063-
執行期間：91 年 08 月 01 日至 92 年 07 月 31 日
執行單位：國立交通大學資訊工程學系

計畫主持人：陳正

報告類型：精簡報告

處理方式：本計畫可公開查詢

中　華　民　國 92 年 10 月 31 日

行政院國家科學委員會補助專題研究計畫成果報告

※※※※※※※※※※※※※※※※※※※※※※※※※
※ 針對系統晶片 (SoC) 架構工作排程方法之探討及 ※
※ 其模擬評估環境之研製 ※
※ ※
※※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：☒個別型計畫　　☐整合型計畫

計畫編號：NSC91－2213－E－009－063

執行期間：　91 年 8 月 1 日 至 92 年 7 月 31 日

計畫主持人：陳正

共同主持人：

本成果報告包括以下應繳交之附件：

☐赴國外出差或研習心得報告一份

☐赴大陸地區出差或研習心得報告一份

☐出席國際學術會議心得報告及發表之論文各一份

☐國際合作研究計畫國外研究報告書一份

執行單位：國立交通大學資訊工程學系

中　華　民　國　92 年　10 月 25 日

行政院國家科學委員會專題研究計畫成果報告

針對系統晶片 (SoC) 架構工作排程方法之探討及其模擬評估環境之研製

A Study of Task Scheduling Techniques for SoC Architecture and Implementation of Its Simulation and Evaluation Environment

計畫編號：NSC 91-2213-E-009-063
執行期限：91 年 8 月 1 日至 92 年 7 月 31 日

主持人：陳正　　國立交通大學資訊工程學系
計畫參與人員：李宜軒、何碩展、李翰青、張明鈿、陳明志、陳建維、許順閔
國立交通大學資訊工程學系

一、中英文摘要

本年度的計畫主要是在系統晶片的架構下，同時考慮執行期限和功率消耗二個排程目標，分別以基因演算法和序列排程法為核心，設計數個工作排程方法。我們同時實作數個對應的模擬評估環境，評估新方法的執行效能。根據我們的模擬和分析，新方法不僅在效能上符合預期，在執行效率上相較於傳統方式，也有相當程度的改進。

關鍵詞：*系統晶片*、*工作排程*、*即時*、*功率消耗*

Abstract

In this project we design several task scheduling methods used in SoC system, which consider both timing and power constraints. We design these methods based on traditional genetic and list algorithms, and try to improve some existed shortcomings. Additionally, we implement corresponding simulation environments to evaluate our methods. According to our simulation results, all proposed methods can not only achieve effective performance, but also more efficient compared with original methods.

Keywords: *System-on-Chip* (*SoC*), *Task Scheduling, Real-time, Power Consumption*

二、計畫緣由與目的

往昔受限於積體電路的製程技術，單顆晶片的功能通常只有運算、控制或儲存，要實作整個系統就必須組合多顆不同功能的晶片。目前隨著深次微米 (deep-submicron) 及超大型積體電路 (VLSI) 製程的進步，同一顆晶片上可容納更多的電晶體 (transistor)，使得整個系統可以實作於單顆晶片之上，以節省功率消耗、空間需求及資料傳輸所需的時間。這程把整個系統實作在同一顆晶片上的技術，就是目前受到廣泛討論的系統晶片 (System-on-Chip, SoC) 架構[1-2]。

一般來說，SoC 系統中通常包含多個運算單元，而這些運算單元的計算能力 (computation power) 及功率消耗 (power consumption) 也不盡相同，若是根據多處理機的架構分類，它也可以看成是一個小型的異質性多處理機 (heterogeneous multiprocessor) 架構[3]。SoC 系統通常用於無線通訊或是資訊家電，這些應用程式對於是否能即時 (real-time) 執行以及功率消耗的多寡，相較於傳統的微處理機 (microprocessor)，都有較明顯且嚴格的限制[4]。因此，如何將欲執行的工作作適當的分配排程，以符合資源限制、執行時間及功率消耗的多重條件，便成為提升整體系統效能的一個切入點。

根據我們收集的資料，目前針對 SoC 系統架構的研究，大部分仍以架構設計及系統測試驗證為主，至於工作排程方面則較少有人涉獵。本實驗室多年來持續研究不同架構上的排程技術，既然 SoC 系統類似小型異質性多處理機架構，在本年度的計畫中，我們便將原本用於叢集式架構的工作排程概念及方法，做適度改進後引入 SoC 系統，提出新的排程方法，在符合系統資源及應用需求的條件限制下，提升整體系統效能。

除了工作排程方法的設計分析，針對各個方法我們也建立相關的模擬評估環境，使用數個實際的應用程式以及隨機產生的圖形來做測試評估。有關我們提出的方法及測試結果，將分別敘述如下。

三、結果與討論

在介紹我們提出的方法之前，先對其他相關的研究做簡單的分類整理。根據我們的觀察，目前用於 SoC 架構的工作排程方法，大部分使用序列排程法 (list scheduling) 或是基因演算法 (genetic algorithm) 做為核心排程技術。至於排程目標，由於最短執行時間或是最低功率消耗二者常無法兼顧，要同時考慮可有以下二種方式：首先是在資源數目 (特別指運算單元數目) 及功率消耗的範圍內，盡量提高平行度，縮短整體執行時間；其次則是在執行時間限制之下，降低功率消耗量。目前的做法以後者為主，因此本年度計畫中我們也以此為排程目標，具體成果說明如下。

第一個方法名為 Constrained Genetic Algorithm (CGA)，以基因演算法為核心，在模擬 SoC 系統的異質性多處理機架構下對即時工作 (real-time task) 做排程，同時考慮各工作的執行期限 (deadline) 及功率消耗。要同時考慮多個排程目標，使用以隨機搜尋 (random search) 為基礎的基因演算法通常可以得到不錯的排程結果。但也因為這種隨機搜尋的方式，當需要排程的工作個數較多時，不容易找到好的排程結果，或是需要很長的排程時間才能使排程結果收斂 (converge)，間接限制了基因演算法的適用範圍。為了改進這個本質上的缺點，在本計畫中我們提出 CGA，改變初始世代 (initial generation) 中染色體 (chromosome) 的產生方式，令其有較高的品質 (quality)，藉以加快排程結果收斂的速度。傳統基因演算法在初始世代通常使用隨機方式產生染色體 (在此代表排程結果)，意即每個工作 (用基因表示) 分配 (allocate) 到每個處理器的機率是相同的 (必須先確認該處理器可以在執行期限前將工作執行完畢)；但在 CGA 則是採用 roulette wheel 的方式，令每個工作在可以即時完成它的所有處理器中，有較高的機率分配給「消耗功率較低」的處理器。由此可知，用 CGA 產生的初始世代染色體可以對應較佳的適應函數值 (fitness function value)，經過染色體選擇 (selection)、互換 (crossover) 和基因突變 (mutation) 等基因運算 (genetic operation) 及數個世代 (generation) 的演化之後，可以在較短的排程時間內令排程結果收斂。根據我們的測試評估 (如圖 1 所示)，CGA 確實可以達到預期的效能，此部分相關研究已發表於會議論文[5-6]。

第二個方法名為 Partitioned Genetic Algorithm (PGA)，它結合 CGA 與演算法中的 divide-and-conquer 機制，解決相同的工作排程問題，進一步降低排程所需的時間。以基因演算法做工作排程，所需的時間與每次排程的工作個數呈正相關；意即只要降低同時排程的工作個數，即可有效縮短排程時間。根據這個特性，在本計畫中我們提出 PGA，先將所有待排程工作分割成數個子群集 (subgroup)，分別用 CGA 產生數個區域排程 (local schedule) 之後，再將它們整合成全域排程 (global schedule)。工作的分割可以有多種方式，在 PGA 中我們採用執行期限當成分割的基準，先將所有工作依執行期限遞增排序，再從頭依序平均分割成數個子群集來做區域排程。在研究中我們已証明這樣的分割方式，子群集之間的資料相依 (dependence constraints) 關係不會形成迴圈 (cycle)，意

即各個區域排程可以「循序」執行；因此我們可以直接將區域排程串接 (cascade) 產生全域排程 (global schedule)，不需設計額外的演算法。根據初步的測試評估 (如圖 2 所示，其中子群集個數為 1 的排程結果相當於用 CGA 排程)，PGA 需要的執行時間確實比 CGA 要短，而且與分割的子群集數目有直接的關係。雖然 PGA 可以達到預期的設計目標，但是它的排程效能卻比 CGA 差，容易產生消耗功率較高的全域排程結果，排程的成功率也較低。關於這個副作用 (side effect) 產生的原因，我們分析是因為各個區域排程均為當時的區域最佳解 (local optimum)，而各個區域最佳解的線性組合 (linear combination) 通常不是全域最佳解 (global optimum)，因此 PGA 的排程結果往往不如 CGA。為了改進這個問題，我們另外設計 Power Minimization Algorithm (PMA) 接在 PGA 之後執行，對之前產生的全域排程做細部調整。根據測試評估 (如圖 3 所示)，PMA 可以確實改進 PGA 的問題，此部分相關研究同樣發表於會議論文[5-6]。

第三個方法名為 Power-Aware List Scheduling (PALS)，以序列排程法為核心，排程問題的設定和目標均與 CGA 和 PGA 相同。大部分的序列排程法主要包含二個步驟：首先將所有工作依排程目標計算對應的優先權值 (priority) 並排序，再依序將工作分配給適合的處理器。由此執行步驟可知，不同的排程目標會產生不同的優先順序，適合的處理器也不相同；若是同時考慮二個以上性質互斥的排程目標，序列排程法會有設計上的困難。一般的做法是在排工作優先順序時使用加權函式 (weight function)，依照不同排程目標彼此之間的重要性，將數個優先權值整合成單一數值來做排序；而在選擇處理器方面也用類似的方式，將工作分配給同時適合多個排程目標的處理器。設計這種使用加權函式的排程法，最大的困難點莫過於各排程目標的加權值不易選擇；而且當處理器之間異質性 (heterogeneity) 較大時，容易產生效能不佳的排程結果。因此我們的 PALS 將不採用加權函式的方法設計，而是將執行時間和功率消耗二個排程目標分先後順序，先找到能令所有工作及時完成的初始排程結果，接著在不違反執行期限的條件下調整各工作分配的處理器，降低所需的功率消耗。PALS 分為以下四個步驟循序執行。首先是工作排序，在 PALS 中我們將工作分成有無執行期限的二類，分別依據各工作的 blevel 值 (在異質性多處理機架構中以平均值計算) 遞減排序；再將二個已排序的工作序列串接，令有執行期限的工作有較高的優先權。其次是工作分配，在此我們使用 ALAP 的機制，將工作依序分配給可以最慢將它執行完畢的處理器。完成這二個步驟之後，我們可以得到一個初始排程結果，在不考慮功率消耗的情形之下，令所有工作能夠及時完成。第三個步驟可以說是 PALS 的設計重點，包含數個規則 (rules)，定義如何在不違反執行期限的先決條件下，依序調整各工作分配的處理器，達到降低功率消耗的目的。經過調整之後的排程結果，已可符合最初設定的排程目標；但由於我們是以 ALAP 的機制來做排程，因此目前這個排程結果可能不是「從時間點為零處」開始執行。為了修正這種情形，PALS 的最後一個步驟是將排程結果做適當的平移 (shift)，使得其中至少有一個工作，可以從時間點為零處開始執行。根據模擬評估的結果，PALS 可以達到與 CGA 類似甚至更佳的排程效能；而且屬於序列排程法，排程的速度明顯比 CGA 和 PGA 快很多，可算是一個效能與效率俱佳的方法。此部分相關研究可參考[7]。

除了以上三個方法之外，我們也將 PGA 的設計理念，用來解決傳統同質性多處理機 (homogeneous multiprocessor) 架構上的工作排程問題 (沒有功率消耗的考量，排程目標為縮短執行時間)。這個方法同樣命名為 Partitioned Genetic Algorithm (PGA)，根據評估也有不錯的效能，相關研究已發表於[8]。

四、計畫結果自評

在本年度的計畫中，我們用異質性多

處理機架構模擬 SoC 系統，分別以基因演算法和序列排程法為核心，提出三個工作排程法，並分別實作對應的模擬評估環境加以測試評估。總體而言，本計畫大致達成以下幾點目標：

1. 提出 CGA，改進傳統基因演算法本質上的缺點，使其能在更短的時間內得到最終的排程結果。
2. 提出 PGA，結合 CGA 及演算法中的 divide-and-conquer 機制，進一步縮短排程所需的時間。
3. 提出 PALS, 以序列排程的方式同時考慮二個排程目標. PALS 的設計方式與傳統方法不同，不但在效能及效率上均有理想結果，亦可避免傳統方法容易產生的缺點。
4. 以 PGA 的設計概念解決傳統同質性多處理機架構上的工作排程問題，藉由 PGA 迅速排程的特性，使基因演算法能有更廣的應用範圍。
5. 設計及實作以上各方法的模擬評估環境，使用實際的應用程式及隨機產生的圖形來做測試評估，証明各方法均能達到預期的效能。

由以上幾點可知，本計畫確實能將 SoC 架構上的某些排程義做詳細深入的探討，改善基因演算法本質上的缺點，嘗試以不同的角度設計序列排程法，並設計實作模擬評估環境加以測試評估。整體看來，我們提出的方法無論在效能或效率上大致都能達到預期的效果。這些研究成果，有部分已發表於會議論文，其餘也將陸續整理成期刊論文投出。

五、參考文獻

[1] Hugo de Man, “System-on-Chip Design: Impact on Education and Research”, *IEEE Design & Test of Computers*, Vol. 16, Issue 3, pp. 11-19, July-Sep. 1999.

[2] Gary Silcott, Janet Wilson, Neil Peterson, William Peisel, and Kirk L. Krodker, “SoCs Drive New Product Development”, *IEEE Computer*, Vol. 32, Issue 6, pp. 61-66, June 1999.

[3] Peng Yang, Chun Wong, Paul Marchal, Francky Catthoor, Dirk Desmet, Diederik Verkest, and Rudy Lauwereins, “Energy-aware Runtime Scheduling for Embedded-multiprocessor SoCs”, *IEEE Design & Test of Computers*, Vol. 18, Issue 5, pp. 46-58, Sep.-Oct. 2001.

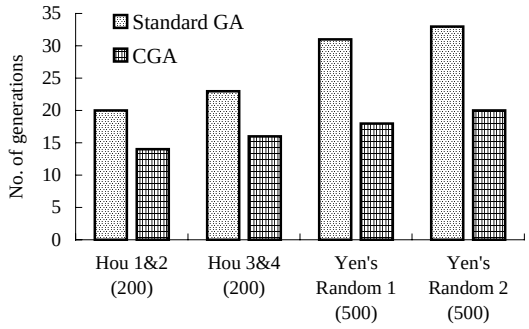
[4] Christian Piguet, Marc Renaudin, and Thierry J-F. Omnes, “Special Session on Low-power Systems on Chips (SoCs)”, *Proc. of Conference and Exhibition on Design, Automation, and Test in Europe*, pp. 488-494, 2001.

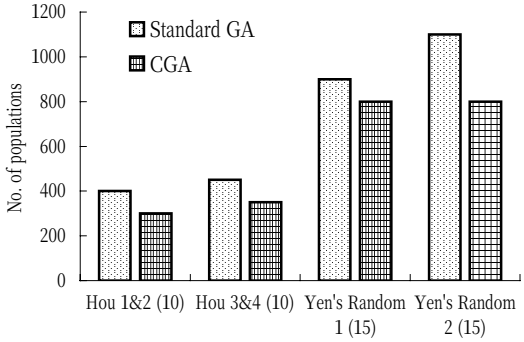
[5] Yen-Hsiang Chang, **An Effective Task Scheduling Genetic Method of Power Aware Consideration for Real-time Embedded Multiprocessor SoC Design**, Master Thesis, National Chiao Tung University, June 2002.

[6] Yen-Hsiang Chang, Yi-Hsuan Lee, and Cheng Chen, “Enhanced Genetic Algorithms for Task Scheduling in an Embedded Multiprocessor System-on-Chip System”, *Proc. of International Computer Symposium*, Vol. 1, pp. 191-199, Dec. 2002.

[7] Han-Ching Lee, **A List-based Task Scheduling Method for Power-aware Heterogeneous Distributed Real-time Embedded Systems**, Master Thesis, National Chiao Tung University, June 2003.

[8] Yi-Hsuan Lee and Cheng Chen, “A Modified Genetic Algorithm for Task Scheduling in Multiprocessor Systems”, *Proc. of the 9th Workshop on Compiler Techniques for High-Performance Computing*, pp. 129-137, March 2003.





* 2 computation components
* The number in the parenthesis on x-axis is the number of populations

* 2 computation components
* The number in the parenthesis on x-axis is the number of generations

圖 1. CGA 排程結果.

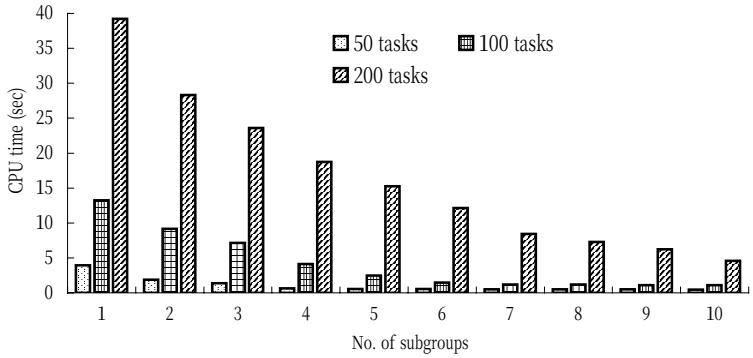
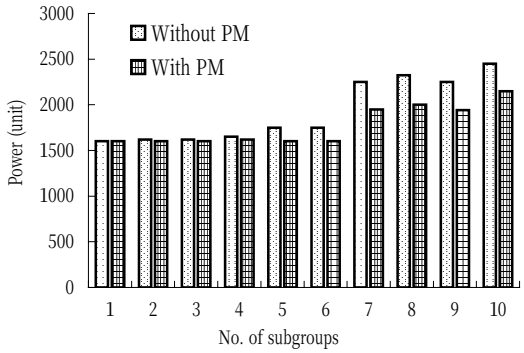


圖 2. PGA 排程結果.



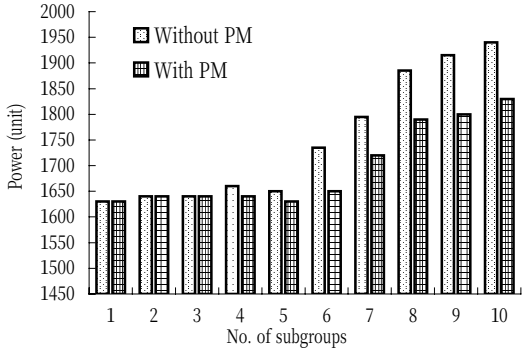


圖 3. PGA 結合 PM 排程結果.