行政院國家科學委員會補助專題研究計畫 □ 成 果 報 告
■期中進度報告

多執行緒程式之漏電耗能管理方法

(Leakage Energy Management for Multithreaded Programs)

計畫類別：■ 個別型計畫 □ 整合型計畫
計畫編號：NSC 97－2218－E－009－043－MY3
執行期間： 97 年 11 月 1 日至 98 年 7 月 31 日

計畫主持人：游逸平
共同主持人：
計畫參與人員： 羅世融、何柏瑲

成果報告類型(依經費核定清單規定繳交)：■精簡報告 □完整報告

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

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢
■涉及專利或其他智慧財產權，□一年■二年後可公開查詢

執行單位：國立交通大學資訊工程學系

中 華 民 國 98 年 5 月 25 日

一、 中文摘要

本計畫為多執行緒程式之漏電耗能管理方法計畫之第一年計畫，主要目的在於設計用於多執行緒程式中的漏電耗能管理方法。在計畫執行過程中，我們提出一套新穎的條件式電源閘控(Predicated Power Gating)硬體機制，並使用編譯器技術，利用已知的 MHP (May-Happen-in-Parallel)技術分析程式可能的並行區間，並提出一套適用於條件式電源閘控機制的管理方法，稱為多執行緒電源閘控分析(Multithreaded power-gating analysis, MTPGA)方法。在本年度計畫中，我們為多執行緒程式的漏電耗能管理提出了一套完整的解決方案，我們在下列各主要環節上取得豐碩的成果：(1) 對於多執行緒的可能並行運算分析做了深入的研究，了解現有分析技術能力，並將其應用至本計畫中。(2) 提出條件式電源閘控硬體機制的概念，以解決舊有機制無法應用至多執行緒程式漏電耗能管理的問題，且此概念未曾在文獻中提出。(3) 利用 MHP 及過去研究的分析資訊，提出一套多執行緒電源閘控分析方法，用以管理及佈置條件式電源閘控指令(Predicated-power-gating instructions)。我們將繼續此研究議題，並將此研究擴展至多核心系統架構。

關鍵詞：編譯器，多執行緒程式，漏電耗能管理

二、 Abstract

This is the first year of our three-year project, called Leakage Energy Management for Multithreaded Programs. The objective of this project is to design a methodology for leakage energy management for multithreaded programs. In this project, we proposed a novel hardware mechanism, called predicated power gating. We also used compiler techniques with the previous proposed MHP (May-Happen-in-Parallel) analysis to propose a scheme, called multithreaded power-gating analysis (MTPGA), for managing predicated-power-gating instructions. In the first year of the project, we developed a solution for employing leakage energy management for multithreaded programs. We achieved some significant results: (1) we have deeply investigated into multithreading analyses on the literature and applied the techniques to the proposed approach. (2) We proposed the concept of predicated-power-gating mechanism in order to solve the problem that normal power-gating mechanism cannot be applied to multithreaded programs. This is a brand new concept and was not appeared in the literature. (3) We reused the information obtained from MHP and previously proposed analyses and proposed a multithread power gating analysis to manage predicated power-gating instructions. We will keep doing research on this subject and extend our research to multi-core architectures.

Keywords: Compilers, Multithreaded programs, Leakage energy management

三、 研究目的

在現代半導體技術中，漏電功率在的整體功率中所占的比例愈來愈高，原因是電晶體的大小不斷縮小而速度卻不斷提升。最近，已有研究提出硬體搭配編譯器技術來降低漏電的消耗，其作法是透過編譯器分析程式並在程式中適當地安插指令以開啟或關閉特定元件，然而，在過去研究中所提出的方法與架構僅適用於單一執行緒的程式，無法針對多執行緒程式進行漏電管理。因此我們需要新的硬體及軟體架構來解決這樣的問題。我們之所以會對多執行緒程式在耗能管理上的議題感到興趣，主因是多執行緒程式設計在現代的程式設計中所扮演的角色愈來愈重要，因為多執行緒可以讓程式充分地使用 CPU 時間，也因此讓程式可以被寫得很有效率，並且，目前正興起的多核心架構也需要多執行程式設計來提高它們的效能；況且，多執行緒是處理以事件驅動軟體的最佳選擇，例如現今許多的嵌入式環境、分散式環境、網路環境等。

四、 文獻探討

為降低處理器或系統晶片的漏電功率消耗，許多以電源閘控(power gating)方法的研究報告及發明被相繼提出，例如 2002 年 LCPC (Workshop on Languages and Compilers for Parallel Computing) 會議中所發表名稱為”Compiler analysis and supports for leakage power reduction on microprocessors”，或 2002 年在 CC(Conference on Compiler Construction)會議中所發表名稱為” Optimizing static power dissipation by functional units in superscalar processors”，或 2003 年在 DATE(Design Automation and Test in Europe Conference)會議中所發表名稱為”Compiler support for reducing leakage energy consumption”或 2005 年在 EMSOFT(International Conference On Embedded Software)會議中所發表名稱為” A Sink-N-Hoist Framework for Leakage Power Reduction”或 2006 年在 ACM TODAES (ACM Transactions on Design Automation of Electronic Systems) 期刊中所發表名稱為” Compilers for Leakage Power Reduction”或 2007 年在 ACM TODAES (ACM Transactions on Design Automation of Electronic Systems) 期刊中所發表名稱為”Compilation for Compact Power-Gating Controls”等論文中，提供了以電源閘控指令的方式對給定的程式進行電源閘控指令佈置來降低各處理元件的漏電能量消耗。其中，電源閘控指令(power-gating instruction)係包含電源啟動指令(power-on instruction)與電源關閉指令(power-off instruction)。電源啟動指令係告知處理器針對某處理元件進行啟動的動作，電源關閉指令係告知處理器針對某處理元件進行關閉的動作。然而，前述這些漏電耗能管理方法僅適用於單一執行緒程式(single-thread program)，對於多執行緒程式(multithreaded programs)則因為編譯器無法掌握執行緒間的互動關係、無法精確得知各處理元件的使用狀況而束手無策。

五、 研究方法

■ Predicated-Power-Gating Mechanism

We import the idea of conditional execution into power-gating devices for solving the improper power-gating control among a set of concurrent threads. Suppose that there are *N* power-gating candidate units: *Candidate*1, *Candidate*2, …, *Candidate*N. Figure 1 and Figure 2 show the block diagram of a normal power-gating mechanism and a predicated-power-gating mechanism. In Figure 1, the block diagram with normal power-gating devices, there is a *N*-bit power-gating control register (*pgc*1, *pgc*2, …, *pgc*N), which controls the power state of the *N* power-gating candidates, and a power-gating controller, which takes power-gating instructions and revises the power-gating control register accordingly. The proposed predicated-power-gating mechanism (Figure 2) is similar to regular power gating, but a *N*-bit power-gating predicate register (*pgp*1, *pgp*2, …, *pgp*N) and *N* switches are added for predication support. In addition, we define the predicated-power-gating operations as follows for the implementation details. Note that these operations must be atomic with respect to each other in order to prevent multiple threads from accessing control at precisely the same time.

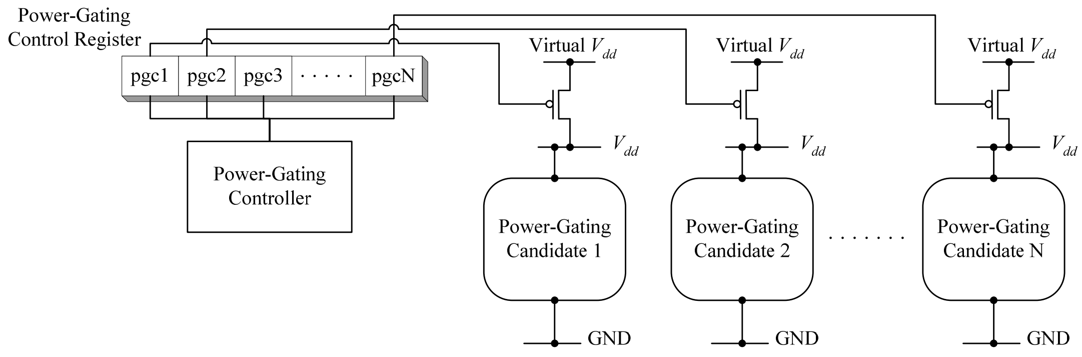


Figure 1: Power-gating mechanism

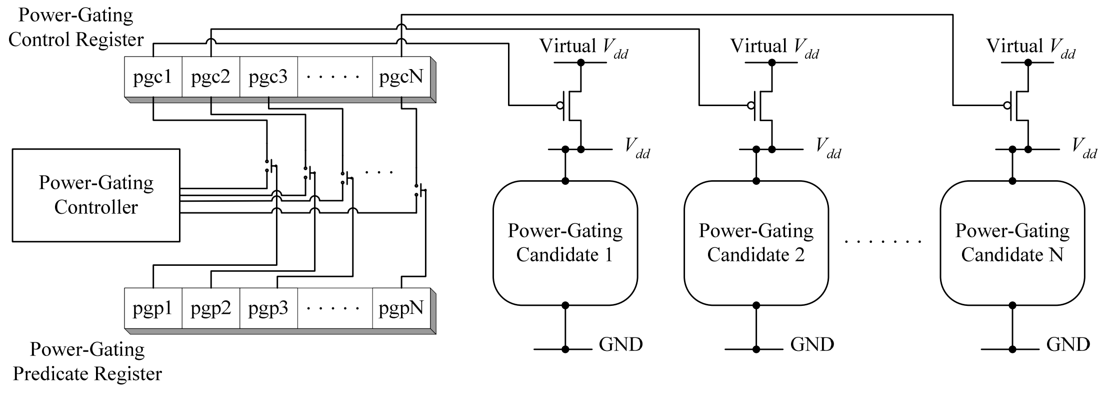


Figure 2: Power-gating mechanism with predication support

- Predicated-power-on operation:
  - ➢ Power on *Candidate*i if predicate bit *pgp*i is set.
  - ➢ Increase the reference counter of *Candidate*i (*rc*i), which keeps track of the number of threads that reference the power-gating candidate at this time, by one.
  - ➢ Unset predicate bit *pgp*i.

- Predicated-power-off operation:
    - Decrease the reference counter of *Candidate*i (*rc*i) by one.
    - Set predicate bit *pgp*i if reference counter *rc*i is zero.
    - Power off *Candidate*i if predicate bit *pgp*i is set.

Moreover, there should be an initialization process for unsetting all the predicate bits (*pgp*1, *pgp*2, …, *pgpN*) and emptying all the reference counters (*rc*1, *rc*2, …, *rcN*) when the processor is activated.

■ Multithreaded Power-Gating Analysis (MTPGA)

The leakage-power-reduction framework presented in "Compilation for Compact Power-Gating Controls" is inapplicable to multithreaded programs since the interleaved execution of threads most likely ruins the concept developed in the framework. However, it does reveal valuable information about the potential control concepts of each single thread within a program. It provides the information of the component activities of each thread and tells whether power-gating controls should be applied within each thread. Once the problem of incorrect power-gating controls among threads becomes solved, power-gating management for multithreaded programs will be achievable.

The proposed MTPGA is proceeded on top of the results of the CADFA with Sink-N-Hoist (proposed in "Compilation for Compact Power-Gating Controls") and MHP (May-Happen-in-Parallel) analyses. The basic idea is to combine both the CADFA with Sink-N-Hoist and MHP information with the predicated-power-gating mechanism. More specifically, MTPGA generally inserts a pair of predicated-power-on and -off operations at the positions that a power-gating candidate is first activated and last deactivated within a thread upon a proposed cost model.

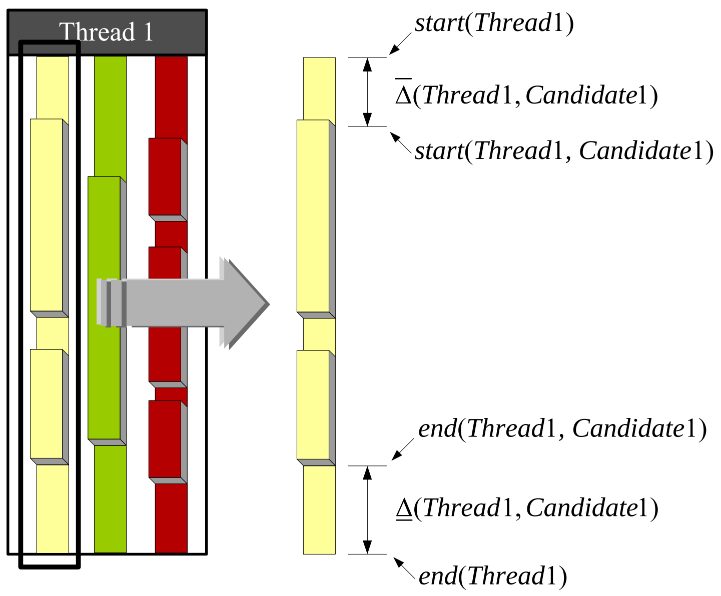
However, predicated-power gating is not cost-free and has more side effect than normal power gating. Sometimes applying predicated-power-gating control might result in less energy reduction than applying normal power-gating control. We take this into consideration and build a model for determining whether predicated-power gating should be employed. The model is based on the comparison of the energy cost between normal power gating and predicated-power gating in a MHP region. Suppose that there are power-gating candidate units: *Candidate*1, *Candidate*2, …, *Candidate*N and *K* threads in a MHP region: *Thread*1, *Thread*2, *: : :*, *Thread*K. We define two functions, named $\overline{\Delta}$() and $\underline{\Delta}$(), which take a thread and a power-gating candidate as their parameters, as follows for computing the inactive period of the power-gating candidate before/after the candidate operates for the first/last time within the thread.

$$\overline{\Delta}(Thread\mathrm{i}, Candidate\mathrm{j}) = start(Thread\mathrm{i}, Candidate\mathrm{j}) - start(Thread\mathrm{i})$$

and

$$\underline{\Delta}(Thread\mathrm{i}, Candidate\mathrm{j}) = end(Thread\mathrm{i}) - end(Thread\mathrm{i}, Candidate\mathrm{j}),$$

where *start*(*Thread*i,*Candidate*j) returns the time *Candidate*j is first used in *Thread*i and *start*(*Thread*i) returns the start time of *Thread*i, and on the contrary, *end*(*Thread*i,*Candidate*j) returns the time *Candidate*j is last used in *Thread*i and *end*(*Thread*i) returns the end time of *Thread*i.



Thread 1
start(Thread1)
$\overline{\Delta}(Thread1, Candidate1)$
start(Thread1, Candidate1)
end(Thread1, Candidate1)
$\underline{\Delta}(Thread1, Candidate1)$
end(Thread1)

Figure 3: Illustration of $\overline{\Delta}$ and $\underline{\Delta}$ computation

Figure 3 portrays the implications of the above functions with *Thread*1 and *Candidate*1, the candidate in yellow, as parameters. Furthermore, we define that $\overline{\mathbb{M}}$(*Candidate*j) and $\underline{\mathbb{M}}$(*Candidate*j) return the minimum of $\overline{\Delta}$(*Thread*i, *Candidate*j) and $\underline{\Delta}$ (*Thread*i, *Candidate*j) in terms of all *Thread*i:

$$\overline{\mathbb{M}}(Candidate\mathrm{j}) = \mathrm{MIN}_{\forall i}\overline{\Delta}(Thread\mathrm{i}, Candidate\mathrm{j})$$

and

$$\underline{\mathbb{M}}(Candidate\mathrm{j}) = \mathrm{MIN}_{\forall i}\underline{\Delta}(Thread\mathrm{i}, Candidate\mathrm{j}).$$

$\overline{\mathbb{M}}$(*Candidate*j) represents the earliest time that *Candidate*j might be used after the MHP region starts and $\underline{\mathbb{M}}$(*Candidate*j) the latest time that *Candidate*j might be operated prior to the MHP region ends. Accordingly, the energy is

$$\mathrm{E}_{on}(Candidate\mathrm{j}) + \mathrm{E}_{off}(Candidate\mathrm{j}) + K_j \times (\mathrm{E}_{pred\text{-}on} + \mathrm{E}_{pred\text{-}off}) + (\overline{\mathbb{M}}(Candidate\mathrm{j}) + \underline{\mathbb{M}}(Candidate\mathrm{j})) \times \mathrm{P}_{rleak}(Candidate\mathrm{j}),$$

where functions E_{on}(*Candidate*j) and E_{off} (*Candidate*j) return the energy consumption of issuing a power-on and a power-off instruction for component *Candidate*j respectively; K_j represent the number of threads in the MHP region that require *Candidate*j to operate; $\mathrm{E}_{pred\text{-}on}$ and $\mathrm{E}_{pred\text{-}off}$ stand for the energy consumption of operating a set of predicated-power-on and predicated-power-off manipulation operations, excluding the power-on and power-off operations, respectively; $\mathrm{P}_{rleak}(C)$ represents the leakage-power consumption of component *Candidate*j in a cycle when the power supply is gated. On the contrary, when we employ normal power-gating control at the beginning and end of the MHP region, rather than applying the predicated-power-gating management, the energy consumption of such operations and the potential leakage dissipation is

$$E_{on}(\textit{Candidate}\mathrm{j}) + E_{off}(\textit{Candidate}\mathrm{j}) + (\overline{\mathbb{M}}(\textit{Candidate}\mathrm{j}) + \underline{\mathbb{M}}(\textit{Candidate}\mathrm{j})) \times P_{leak}(\textit{Candidate}\mathrm{j});$$

where $P_{leak}(C)$ represents the leakage-power consumption of unit *Candidate*j in a cycle.

Accordingly, we can derive the following inequality for ensuring the worthiness of predicated-power gating:

$$E_{on}(\textit{Candidate}\mathrm{j}) + E_{off}(\textit{Candidate}\mathrm{j}) + K_j \times (E_{pred¡on} + E_{pred¡off}) + (\overline{\mathbb{M}}(\textit{Candidate}\mathrm{j}) + \underline{\mathbb{M}}(\textit{Candidate}\mathrm{j})) \times P_{rleak}(\textit{Candidate}\mathrm{j}) < E_{on}(\textit{Candidate}\mathrm{j}) + E_{off}(\textit{Candidate}\mathrm{j}) + (\overline{\mathbb{M}}(\textit{Candidate}\mathrm{j}) + \underline{\mathbb{M}}(\textit{Candidate}\mathrm{j})) \times P_{leak}(\textit{Candidate}\mathrm{j}),$$

and thus we have

$$\overline{\mathbb{M}}(Candidate\mathrm{j}) + \underline{\mathbb{M}}(Candidate\mathrm{j}) > \frac{K_j \times (\mathbb{E}_{pred-on} + \mathbb{E}_{pred-off})}{\mathbb{P}_{leak}(Candidate\mathrm{j}) - \mathbb{P}_{rleak}(Candidate\mathrm{j})}$$

as the criterion for deterring whether predicated-power gating should be applied. Figure 4 sketches the algorithm of the proposed multithreaded-power-gating analysis. Basically, it determines whether predicated-power gating should be employed for each MHP region.

```
Multithreaded-Power-Gating Analysis

  Input:  A multithreaded program and its MHP and CADFA with
          Sink-N-Hoist information.
 Output:  The program with power-gating controls.


  for each MHP region
    for each power-gating candidate $C$
      if $\overline{\mathbb{M}}(C) + \underline{\mathbb{M}}(C) \le THRESHOLD^{\dagger}$ then
        Place a power-on and a power-off instruction for
        $C$ at the beginning and the end of the MHP
        region, respectively.
      else
        for each thread
          Place a predicated-power-on and a predicated-
          power-off operation before/after the candidate
          operates for the first/last time within the
          thread.
        end for
      end if
    end for
  end for

  $^{\dagger}THRESHOLD = \frac{K_j \times (\mathbb{E}_{pred-on} + \mathbb{E}_{pred-off})}{\mathbb{P}_{leak}(C) - \mathbb{P}_{rleak}(C)}$
```

Figure 4: Algorithm of the multithreaded-power-gating analysis

六、 結果與討論

本年度計畫中，我們的研究重點包括：

（一） May-Happen-in-Parallel (MHP) Analysis

使用文獻中的技術分析多執行緒程式中可能並行的程式區塊，我們在文獻中找到在MHP 分析技術領域中重要的研究論文，並加以研讀，作為本項研究的 MHP 分析基礎：

◇ R. N. Taylor, “Complexity of analyzing the synchronization structure of concurrent programs,” Acta Informatica, vol. 19, pp. 57 – 84, 1983.

◇ D. Callahan and J. Sublok, “Static analysis of low level synchronization,” in Proceedings of the 1988 ACM SIGPLAN and SIGOPS Workshop on Parallel and Distributed Debugging, Madison, Wisconsin, January 1989, pp. 100 – 111.

◇ E. Duesterwald and M. L. Soffa, “Concurrency analysis in the presence of procedures using a data-flow framework,” in Proceedings of the Symposium on Testing, Analysis, and Verification (TAV’91), British Columbia, Canada, October 1991, pp. 36 – 48.

◇ S. P. Masticola and B. G. Ryder, “Non-concurrency analysis,” in Proceedings of the fourth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP’93), San Diego, California, May 1993, pp. 129 – 138.

◇ G. Naumovich and G. S. Avrunin, “A conservative data flow algorithm for detecting all pairs of statements that may happen in parallel for rendezvous-based concurrent programs,” in Proceedings of the 6th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE’98), Lake Buena Vista, Florida, November 1998, pp. 24 – 34.

◇ G. Naumovich, G. S. Avrunin, and L. A. Clarke, “An efficient algorithm for computing MHP information for concurrent Java programs,” in 7th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE’99), ser. Lecture Notes in Computer Science. Springer, 1999, vol. 1687, pp. 338 – 354.

（二） Predicated-Power-Gating Mechanism

在本計畫中，我們提出使用具有 Predicated Execution 機制的電源閘控模組，並在同一MHP 群組的每一執行緒前後安插 Predicated Power-Gating Instruction 及適當的Predication 管理指令，以達到漏電耗能管理的目的。在現有文獻中尚無發現與本技術相同的研究。

（三） Multithreaded Power-Gating Analysis (MTPGA)

在本計畫中，我們提出一套搭配上述硬體機制的編譯器技術，用來進行多執行緒的電源閘控指令管理，以決解舊有機制因多執行緒程式對於編譯器而言具有執行緒順序不定的特性而無法有效管理漏電的問題，此技術的發展將有助於多執行緒程式的漏電耗能管理。

七、 計畫成果自評

根據以上這些成果，我們將可以對往後的研究進行更進一步地瞭解與改良。並藉由一些初

期實驗的測試，得知我們研究的方法及結果確實對多執行緒程式的漏電耗能管理有所助益，符合計畫達成的目標及進度。未來本實驗室將持續針對多執行緒程式之漏電耗能管理相關設計進行廣泛與深入之研究與實作。

八、 參考文獻

1. Steven Dropsho, Volkan Kursun, David H. Albonesi, Sandhya Dwarkadas, and Eby G. Friedman. ``Managing static leakage energy in microprocessor functional units," In *Proceedings of the 35th International Symposium on Microarchitecture (MICRO'02)*, pages 321-332, Istanbul, Turkey, November 2002.
2. Yen-Hsiang Fan, Yuan-Shin Hwang, Yi-Ping You, and Jenq-Kuen Lee, ``Compiler-based vs. Hardware-based Power Gating Techniques for Functional Units," in *Proceedings of the 6th Workshop on Optimizations for DSP and Embedded Systems (ODES-6)*, pp. 26-35, Boston, MA, April 6, 2008.
3. Siddharth Rele, Santosh Pande, Soner Onder, and Rajiv Gupta. ``Optimizing static power dissipation by functional units in superscalar processors," In *Proceedings of the 11th International Conference on Compiler Construction (CC'02)*, pages 261-275, Grenoble, France, April 2002.
4. Yi-Ping You, Chingren Lee, and Jenq-Kuen Lee, ``Compiler Analysis and Supports for Leakage Power Reduction on Microprocessors," in *Proceedings of the 15th Workshop on Languages and Compilers for Parallel Computing (LCPC'02)*, College Park, MD, July 25-27, 2002. (also in *Lecture Notes in Computer Science*, Vol. 2481, Springer-Verlag, Germany, pp. 45-60, 2005.)
5. Yi-Ping You, Chingren Lee, and Jenq Kuen Lee, ``Compilers for Leakage Power Reduction," *ACM Transactions on Design Automation of Electronic Systems*, Vol. 11, Issue 1, ACM, New York, pp. 147-164, January 2006.
6. Yi-Ping You, Chung-Wen Huang, and Jenq Kuen Lee, ``A Sink-N-Hoist Framework for Leakage Power Reduction," in *Proceedings of the ACM International Conference on Embedded Software (EMSOFT'05)*, pp. 124-133, Jersey City, NJ, September 18-22, 2005.
7. Yi-Ping You, Chung-Wen Huang, and Jenq Kuen Lee, ``Compilation for Compact Power-Gating Controls," *ACM Transactions on Design Automation of Electronic Systems*, Vol. 12, Issue 4, Article 51, ACM, New York, September 2007.
8. Yi-Ping You and Jenq Kuen Lee, ``Compiler Frameworks for Leakage Power Reduction," in Student Poster Session of *ACM SIGPLAN/SIGBED 2005 Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES'05)*, Chicago, IL, June 15-17, 2005.
9. W. Zhang, Mahmut T. Kandemir, Narayanan Vijaykrishnan, Mary Jane Irwin, and V. De. ``Compiler support for reducing leakage energy consumption," In *Proceedings of the 6th Design Automation and Test in Europe Conference (DATE'03)*, pages 1146-1147, Messe Munich, Germany, March 2003.