使用電源閘控技術節省漏電流之控制策略研究

葉長青[1] 葉長茂[2]

[1] 中州科技大學電子工程系 eye2001@dragon.ccut.edu.tw

[2] 中州科技大學機動工程系 yeh120@dragon.ccut.edu.tw

摘 要

隨著深次微米製程技術快速發展，漏電流所造成的靜態功率消耗已成為系統晶片設計中極重要的問題。故在低功率晶片設計領域中，使用電源閘控技術插入適當大小及數目的睡眠電晶體（sleep transistors）於電路中，關閉正在閒置的功能單元是減少漏電流功率消耗最有效的方法。但是因切換電源需先經過睡眠電晶體對目標電路完成充放電，而且電晶體切換速度不可太快，以免瞬間電流產生過大的電源電壓突波，影響附近正在使用的電路，造成整體晶片功能錯誤；另一方面若改以延長切換時間來減小瞬間電流則往往造成系統效能損失，使得電源閘控設計在實際的應用遭遇到很大的困難；故本論文探討動態電源閘控控制策略，深入分析使用電源閘控所能減少的漏電流能量消耗以及對系統效能的影響。

關鍵詞：電源閘控、漏電流、低功率電路、睡眠電晶體

壹、緒 論

漏電流是深次微米製程晶片中功率消耗的主要部份，也是電路設計的關鍵考量因素。電源閘控（power gating）是現今大多數系統晶片採用的低功率設計技術，在電源與目標電路之間插入睡眠電晶體（sleep transistors），關閉電源即可減少漏電流，晶圓廠配合的標準元件庫（Standard Cell Library）及 IC 設計軟體也都支援此技術。基本上電源閘控很簡單，電路不用時只要關閉電源；然而，在實際電路設計中需要面對很多複雜的問題，比如睡眠電晶體的設計與尺寸的調整、目標電路與正在使用的電路之間信號的隔離、目標電路內部狀態的保存、電源網路壓降、喚醒期間的電源電壓突波、喚醒時間的減少及閘控電路增加的面積與額外功

通訊作者

姓名：葉長青

E-mail: eye2001@dragon.ccut.edu.tw

率消耗等等[1]。現今的控制技術藉由偵測閒置的功能單元，關閉電源使其進入低漏電流狀態；然而實際設計需考慮的限制條件是喚醒功能單元一定要打開睡眠電晶體，針對虛擬接地的電容電壓放電而需要不小的時間[2][3]。同時，電源閘控電路中切換電源產生的電壓突波[4]將會影響周圍電路運算的可靠性，因此減少喚醒時間（wakeup time）將需要進一步考量電壓突波的影響使得系統電路設計變的更複雜[5][6]。除了製程技術，大部份的技術集中在電路階層最佳化，目標電路在減少漏電流的同時也導致系統效能損失。若系統中同時有多個電路模組使用電源閘控減少漏電流，則其所造成的系統效能損失將十分可觀。因此在沒有損失效能的情況下，能有效的應用電源閘控於功能單元中每一個閒置時間間隔需要一個可行的預測控制技術[7]。

控制器設計方式的探討，IBM Watson 研究中心 Hu 提出 time-based power gating [8]，當功能單元的閒置時間超過一個預先設定的臨界時間，則預測將有足夠長的時間使用電源閘控。閒置的功能單元平時會進入睡眠狀態,僅當有指令需要執行時才喚醒；一旦功能單元使用電源閘控而無法立即執行時，處理器將需要延長程式的執行時間。如圖 1 所示，功能單元在沒有啓用電源閘控時，處於 working 狀態；若閒置時間超過計時器所設定的臨界偵測時間（也就是 idle detection threshold 或 Tidledetect）時，認為未來將有足夠長的閒置時間，故關閉電源並進入 break_even 狀態；若關閉電源時間超過 Tbreakeven 時間時，切換電源產生的額外能量消耗已被減少漏電流所節省的能量抵消，故進入 sleep（睡眠）狀態，開始達到節省能量的效果。也就是說，如果功能單元有連續 Tidledetect 沒有執行時，將啓動電源閘控來減少漏電流，在開始 Tbreakeven 的時間所節省漏電流的能量是用於抵消切換電源開關所消耗的額外能量，接下來在睡眠模式才有節省能量的效果，當有指令需要使用功能單元時，停止電源閘控回到正常狀態。

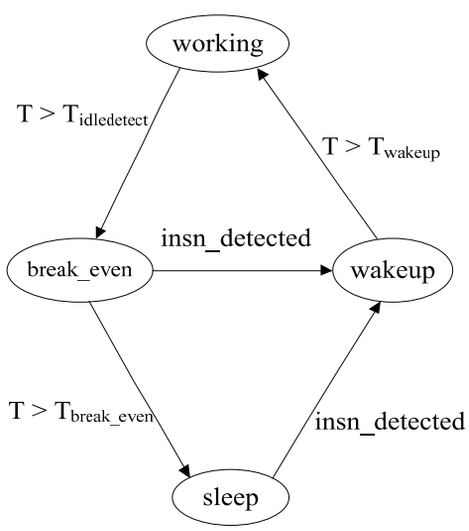


圖 1　time-based power gating 之狀態機 （Adapted from [8]）.

因此 Time-based power gating 針對在程式執行的過程中，功能單元的閒置時間超過事先設定的臨界偵測時間，預測未來有足夠的閒置時間使用電源閘控。這個方法基本上可以達到

節省能量的需求，但因為控制方法具有預測性，頻繁的預測失誤而切換電源將導致大量的額外能量消耗。而且，即使預測正確，使用電源閘控有足夠長的時間 Tbreakeven 可以抵消切換電源所消耗的額外能量，但對處理器所造成的執行時間的延長將使其它單元增加額外的漏電流消耗。這種節省能量效果的不穩定性有可能出現在造成重覆預測錯誤的應用程式中，甚至於被惡意軟體（如 power virus）利用[9]，因此，現今主要的研究方向是能有效節省漏電流並避免大量的不正確電源閘控。

因為各個程式在功能單元執行時皆有不相同的閒置時間，Youssef [10]動態調整臨界偵測時間的長度來避免預測重複失誤，為了提供閒置時間長度的預測訊息，新增一些計數器來計算過去使用電源閘控的時間是否達到可抵消切換電源所消耗的能量；當應用程式頻繁地重複使用功能單元，使得功能單元使用電源閘控的時間沒有超過 breakeven 時間，這段時間不適合使用電源閘控，就會提高臨界偵測時間來避免使用電源閘控；另一方面，若功能單元不常被使用，則降低臨界偵測時間來提早進入睡眠模式，可有效節省漏電流。

Duke 大學的 Lungu 及 IBM Watson 研究中心的 Bose [9]認為電源閘控控制對於各種應用程式可能會有無法偵測到的行為，包括 Hu 的 time-based power gating[8]及 Youssef [10]的動態調整臨界偵測時間，而導致額外能量損失，故提出一個防衛機制來避免頻繁切換電源產生的大量功率。當使用電源閘控的結果大部份都不正確時，則停止使用。

以上兩種方法可避免錯誤的預測產生額外能量的代價，然而面對今日複雜的應用程式環境，往往存在各種長度的閒置時間交錯，單純的計算預測錯誤的次數很難完全預測各種應用程式不同的閒置時間分佈。例如較長的閒置時間和較短的閒置時間呈現某種規律性的交錯分佈時，則使用 Youssef [10]的方法無法決定應該降低或提高臨界偵測時間；若使用 Lungu [8]的方法停止使用電源閘控，則將無法利用其中間斷出現的較長閒置時間來節省漏電流。基本上，雖然一個預測方法很難處理所有狀況，總是會有所取捨，但交錯的閒置時間分佈在各類應用程式中也是很普遍的情況，故還是應有一個適當的解決機制。除此之外，這兩種方法只針對電源閘控是否可成功減少功能單元的漏電流，沒有考慮頻繁喚醒造成效能損失的問題；然而，總是會有指令需要使用功能單元，所以喚醒也是必需考慮加以處理。因此現今的控制方式主要使用低功率編譯器技術，在指令執行之前插入合適的喚醒指令，避免喚醒所造成的效能損失。同樣的，一個有效的架構技術也應該需要提供一個預先喚醒機制，以便提供指令即時執行。

貳、電源閘控之研究與探討

目標電路有一段時間沒有在使用時，表示其閒置時間可以關閉電源。因使用電源閘控技術僅有在目標電路有足夠的閒置時間才能達到減少能量的效果，故在討論電源閘控技術之前，應先分析電源閘控能成功減少能量應具備的條件。其次在下一章實驗結果中，我們使用一個迴圈應用程式（取自一個人員資源管理應用，主要是用來計算一年中總共雇用的人口數目，[11]使用此應用程式說明快取記憶體如何利用閒置間隔時間來減少漏電流）來觀察功能

單元在不同長度的閒置時間使用電源閘控所能節省的能量及對系統效能所造成的影響。

一、能量分析模型

因為處理器執行程式通常僅需要使用一部份的功能單元，其它的閒置功能單元可進入睡眠模式以便節省漏電流。但是有些應用程式時常使用功能單元，導致頻繁的切換正常模式與睡眠模式，對可節省的能量及系統效能造成很大的不利影響。因此就系統階層的觀點，使用電源閘控設計一定要考慮不同模式頻繁切換所造成的負面影響。

這節針對使用電源閘控提出一個簡單的系統能量分析，首先提出分析方程式來估計可節省的能量，方程式中也將切換電源所增加的執行時間納入考量。如式（1）所示，功能單元所消耗的能量為動態能量及靜態能量之和。

$$E^{fu} = E^{fu}_{dyn} + E^{fu}_{leak}$$

$$= \mu T * P^{fu}_{dyn} + T * P^{fu}_{leak} \qquad (1)$$

其中、P^{fu}_{dyn} 是功能單元中的動態功率消耗、P^{fu}_{leak} 是靜態功率消耗、E^{fu} 是功能單元的總能量消耗、μ 是功能單元使用（執行）時間的比例、T 是執行時間。式中第一項動態能量消耗是平均動態功率消耗乘以功能單元的使用時間。同樣的，第二項靜態能量消耗是平均靜態功率消耗乘以功能單元的使用時間。因為電晶體變得更小而且更快，漏電能量消耗已經日益重要，一旦閒置的功能單元使用電源閘控，將可減少漏電能量。而且只有需要使用的功能單元消耗能量，因此能量消耗如式 2 所示

$$E^{fu}_{gated} = \mu_{gated} T_{gated} * P^{fu}_{dyn} + (1-\alpha) T_{gated} * P^{fu}_{leak} \qquad (2)$$

其中 μ_{gated} 和 T_{gated} 是當功能單元使用電源閘控技術之後的功能單元使用時間的比例和整體程式執行時間，而 α 是使用電源閘控的時間比例。因相同的負載應有相同的動態能量消耗，我們假設 $\mu T = \mu_{gated} T_{gated}$。因電源閘控可能延長執行時間，故 T_{gated} 可能大於 T。

如式中第 2 項說明，有效的電源閘控跟 α 值大小有關聯，為了減少較多的漏電流，α 值應越大越好。然而增加 α 可能產生額外的切換次數，造成執行時間（T_{gated}）大幅增加，反而影響處理器效能。

因此式 3 中將由於電源閘控所增加執行時間造成系統能量的提高加入考量

$$E_{cost} = E_{gated}^{other} - E^{other}$$

$$= P_{leak}^{other} * (T_{gated} - T) \quad (3)$$

其中 E^{other} 是處理器的能量消耗（不包括功能單元），而 E_{gated}^{other} 是當功能單元使用電源閘控時的能量消耗。因為電源閘控能夠延長程式的執行時間，E_{gated}^{other} 應該比 E^{other} 有較多的漏電能量消耗。

根據式 1、式 2 和式 3，使用電源閘控技術所能減少的能量比例如下所示：

$$E_{saving} = E^{fu} - E_{gated}^{fu} - E_{cost}$$

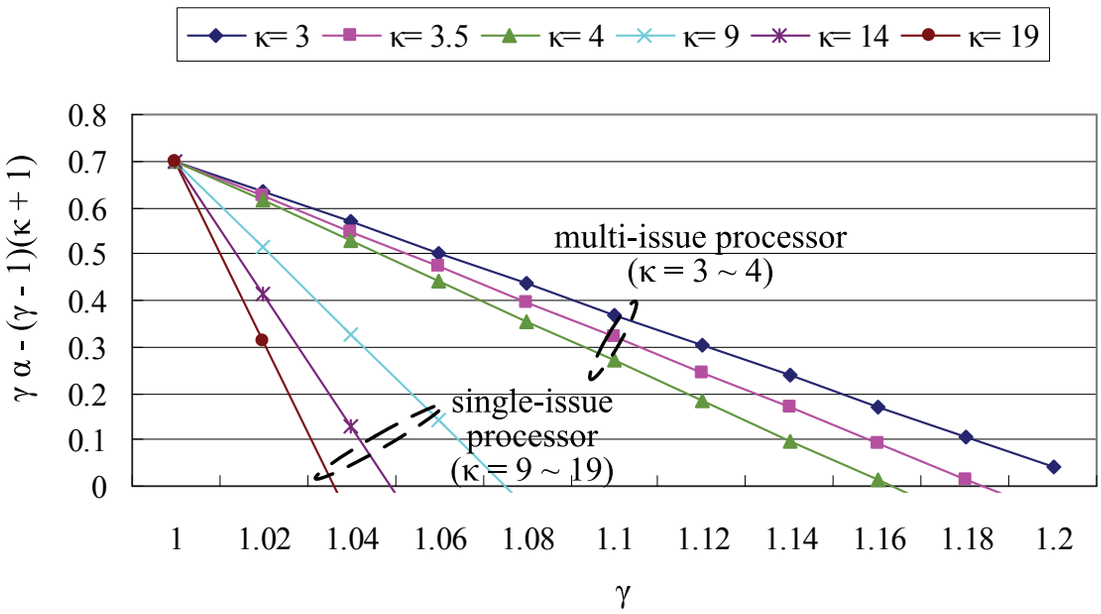
$$= T * P_{leak}^{fu} - (1-\alpha) T_{gated} * P_{leak}^{fu} - (T_{gated} - T) * P_{leak}^{other}$$

$$= T * P_{leak}^{fu} [1 - (1-\alpha)\gamma - (\gamma - 1)\kappa]$$

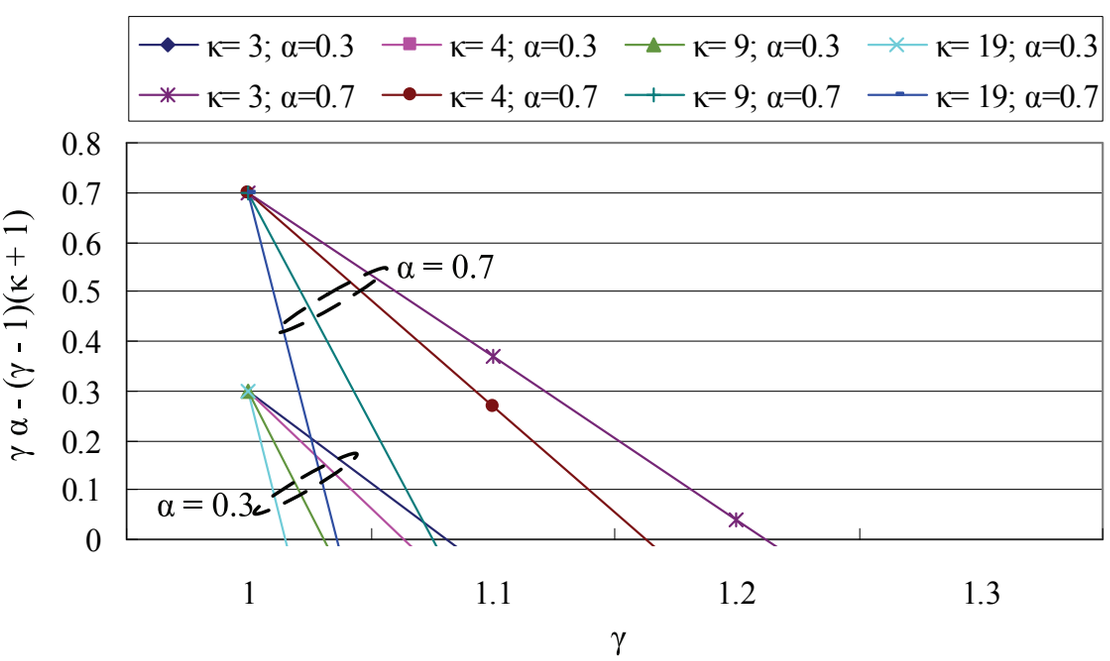
$$= T * P_{leak}^{fu} [\gamma\alpha - (\gamma - 1)(\kappa + 1)] \quad (4)$$

其中 Tgated= γ T 和 $P_{leak}^{other} = \kappa P_{leak}^{fu}$ 。因此僅有當 $\gamma\alpha - (\gamma - 1)(\kappa + 1) > 0$ 才能使用電源閘控。第 1 項擴充使用電源閘控的比例 γ；相對的，第 2 項是整體處理器系統由於使用電源閘控增加執行時間而導致額外增加的漏電流能量。參數 κ 是功能單元佔整體處理器系統的能量比例。

其次考慮式 4，Esaving 正比於 P_{leak}^{fu}，也就是說因為漏電流在功能單元中的比例將隨著製程縮小而增加,故在未來製程中使用電源閘控將能進一步節省更多的能量；而且為了滿足日益增加的運算需求，處理器將需要更多的功能單元進行運算。



(a) 根據不同的增加時間（γ）及功能單元在系統所佔的能量比例（κ）分析使用電源閘控可得到的效果。（其中電源閘控時間比例 α 值是 0.7）



（b）根據不同的電源閘控時間比例（α）分析使用電源閘控可得到的效果

圖 2 漏電流控制的敏感度分析

此外由式 4 可知如何提供有效的減少漏電流的方法，包括增加使用電源閘控時間（α）、減少因切換電源所增加的時間（γ-1）和增加功能單元佔整體處理器系統的能量比例（也就是增加參數（κ））。式 4 的主要目標就是將由於被延長的執行時間所導致處理器能量的增加納入考量。因此，隨著目前處理器晶片的發展趨勢朝向整合不同的功能，導致處理器面積的增加，也使得電源閘控對其所能節省的能量將有更多負面的影響。

二、電源閘控的能量探討

因此，γ 和 α 受到電源閘控技術的影響，也被視為可同時影響處理器效能及能量消耗的關鍵參數，更精確的說，γ 值應該越接近 1 越好。為了更了解這兩個參數所造成的影響，根據不同的 γ 及 κ 值，可得到圖 2；其中 α 值是 0.7，表示功能單元有 70%的執行時間可使用電源閘控來減少漏電流。因此，除非 γ 及 κ 值過大，否則漏電流控制都是有效的。

也就是說，為了避免電源閘控對系統效能及能量消耗產生不利的影響，控制機制必需要能夠避免增加過長的執行時間，而且必需有足夠多的功能單元才能使電源閘控電路有減少能量的效果。因為使用電源閘控在處理器的某些部份可能暫時停止整體程式執行而延長執行時間，造成系統其它部份的漏電流能量確時會增加。所以由於效能損失使處理器所增加的漏電流能量消耗不能太過嚴重，否則將導致使用電源閘控所節省的漏電流能量無法抵消整體系統增加的能量消耗。

漏電流能量的減少是在使用不同的電源閘控時間比例下，只有當 κ 值夠小時才比較有效。如圖 2（b）所示，很明顯的，隨著減少使用電源閘控的時間，所能節省的能量消耗比起增加使用電源閘控的時間來的少。此外，由於效能損失所增加的能量消耗很有可能抵消減少漏電流所節省的能量或甚至產生更多的能量消耗。因此漏電流控制機制僅有在有限執行時間增加（γ 值夠小）的條件下藉由提供足夠的使用電源閘控的時間（α 值夠大）和較多的功能單元（κ 值夠小）才能有效減少漏電流。

三、電源閘控控制方式之探討

有關漏電流問題，在電路設計階段，藉由分割目標電路（circuit clustering）至適當的大小，並使用喚醒排程技術（wakeup scheduling）決定睡眠電晶體的開啓順序，以便能同時保持系統運算正確性和最小喚醒時間。雖然這方面的研究可以降低喚醒時間對效能的影響，然而切換電源需同時考慮電源突波及喚醒時間，導致電路複雜度的增加（例如：喚醒排程演算法需要使用狀態機來控制睡眠電晶體打開的順序[5]）。

其次，針對超長指令字元（VLIW）處理器使用靜態排程，因此低功率編譯器技術可以在編譯程式時，加入一定數量的控制指令，執行時會根據這些控制指令決定適當的處理器工作模式；因此，利用編譯器可以針對功能單元有足夠長的閒置時間切斷電源，也可以在使用功能單元之前，預先做喚醒（pre-wakeup）的動作；然而，系統晶片包含很多的電路區塊，若多個功能單元同時使用電源閘控，將產生大量的控制指令，不僅增加程式的長度，也拉長程式執行的時間。因此 You[10]提出一種機制，合併多個控制指令至一個複合指令來壓縮控

制指令的數量；其次，因編譯器事先針對應用程式做靜態分析，找出適合使用睡眠模式的功能單元；然而，靜態分析無法完全準確預測分支指令執行方向，可能發生電源閘控或預先喚醒的預測失誤，增加不必要的切換能量及造成處理器效能損失。

參、研究結果與討論

為了了解電源閘控如何減少漏電流的能量及對效能的影響，我們使用 Wattch 工具模擬程式，模擬器內相關參數的設定如表 1。

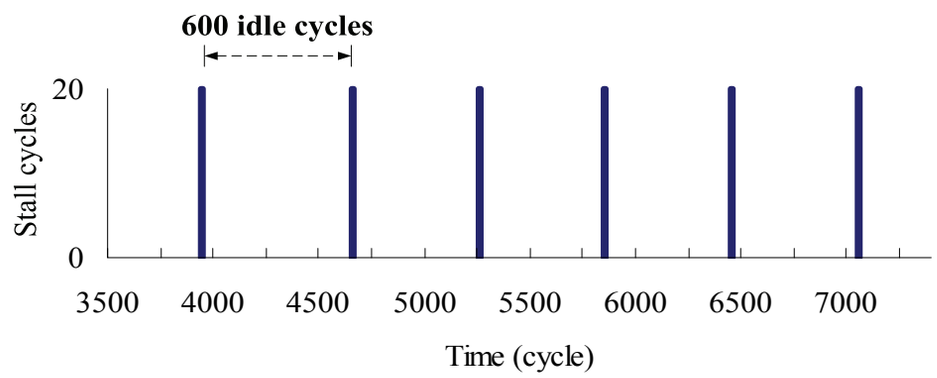
表 1 處理器基本架構

Fetch Queue	8 entries
Fetch/Decode	8 instructions per cycle
Issue/Commit	8 instructions per cycle
Functional units	4 integer alu, 1 integer mult 4 FP alu, 1 FP mult
Branch Predictor	gshare, 16K table, 14-bits history
L1 ICache	64K, 2-way, 32B
L1 DCache	64K, 2-way, 32B
L2 Cache	1MB, 4 way, 6 cycles hit latency
TLB size	ITLB: 16 set 4 way DTLB: 32 set 4 way 4KB page size, 30 cycles penalty
Memory	8 bytes/line, virtual memory 4 KB pages

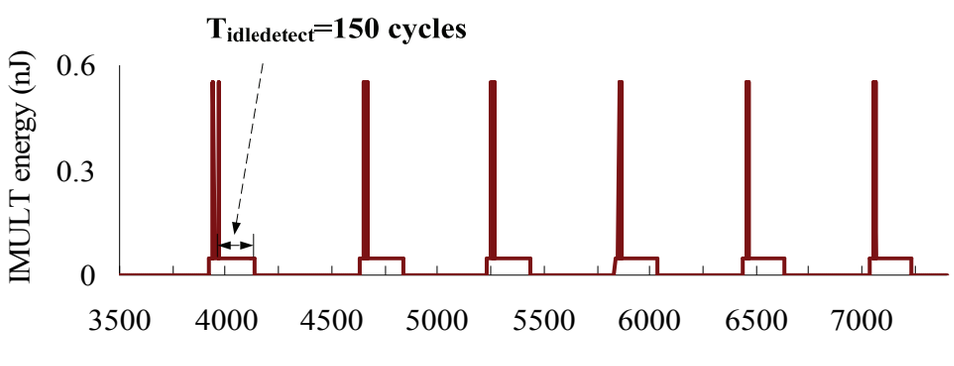
我們使用一個迴圈程式進行模擬，如圖 3 所示，兩個連續乘法指令的間隔跟內層迴圈中的時間（*high*（*i*）*-low*（*i*））有關。當這個間隔時間大於系統的閒置偵測時間（$T_{idledetect}$），整數乘法器將進入睡眠模式，並在下次執行指令之前，先喚醒需要的功能單元，使其進入正常模式後才可以使用。

```
int    i, j, sum, total;
int    low(int), high(int);
for (total=0, i=1; i <= 12; i++)
{
       for (sum = 0, j = low(i); j < high(i); j++)
       sum += a[j];
       mult: sum *= i;
       total+=sum;
}
```

圖 3　C 語言範例程式



(a) Recurrent wakeup activities.



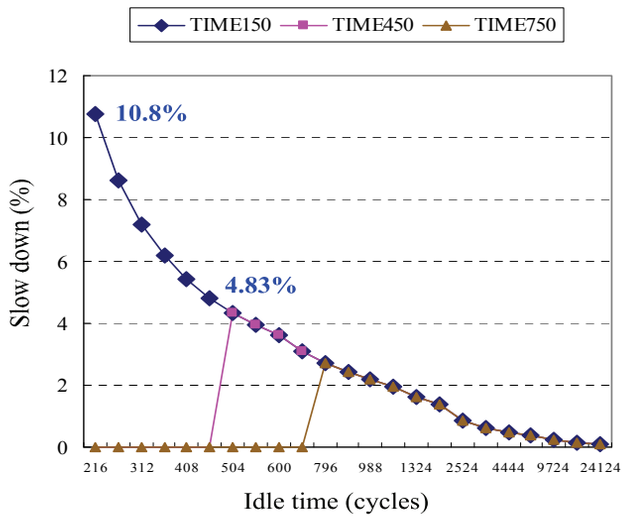
(b) Energy savings.

圖 4　動態電源閘控控制模擬波形，閒置偵測時間設定為 150 個時脈，可執行圖 3 之迴圈範例程式，每 600 個時脈執行一次乘法指令。

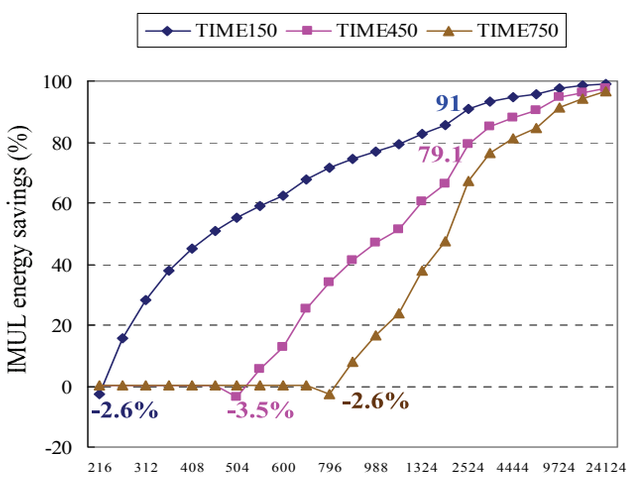
如圖 4 所示，我們假定每個間隔 600 個時脈執行乘法指令，time-based power gating 使用閒置偵測時間（$T_{idledetect}$）來檢查功能單元沒有執行指令的時間長度，超過時則開始使用電源閘控，此長度設定為 150 個時脈。隨著時間的增加，圖 4 顯示整數乘法器暫停的時脈數及節省的能量。因為閒置偵測時間比整數乘法器的連續沒有使用的時間（也就是閒置時間）短，導致整數乘法器不斷重覆使用電源閘控及喚醒，也就是說，整數乘法器每隔 600 個時脈即暫停喚醒時間（T_{wakeup}）。在這個例子中，執行時間每隔 600 個時脈被延長喚醒時間（T_{wakeup}），而且有閒置偵測時間（$T_{idledetect}$）不能減少漏電流。而且需注意的地方是若這些功能單元使用電源閘控比每 600 個時脈更頻繁時，將進一步減少系統效能。

為了顯示圖 3 迴圈程式執行時，有關乘法器的閒置時間、電源閘控閒置偵測時間（$T_{idledetect}$）、系統效能損失及可節省的能量。我們改變（$high(i)$ -$low(i)$）的範圍，圖 5 顯示不同的閒置時間由 216 個時脈到 24124 個時脈。整數乘法器的最小閒置時間是 216 個時脈，這表示每個 216 個時脈執行一次乘法指令。圖 5（a）表示執行時間增加的比例，圖 5（b）表示整數乘法器相對節省的能量。在 Time150，閒置偵測時間是 150 個時脈，比整數乘法器的最小閒置時間 216 個時脈還要小，因此，重覆喚醒造成效能損失 10.8%。

相對的，使用 450 個時脈及 750 個時脈等較高的閒置偵測時間能夠避免較短閒置時間的大部份的喚醒次數，然而，節省能量的效果就比較低。例如 TIME450 比起 TIME150 在閒置時間小於 9000 個時脈時就節省較少能量。在閒置時間為 504 個時脈時，閒置偵測時間為 450 個時脈得到負的節省能量，因為執行時間增加太多的關係，額外增加的漏電流反而使整體能量增加。最後當閒置時間大於 9724 個時脈時，效能損失可忽略，節省能量接近 100%，因此，動態控制技術僅能針對較長的閒置時間達到有效的減少漏電流及保持效能。



(a) Slow down due to wakeup latency overhead.



(b) IMUL energy savings.

圖 5　動態閘控技術在閒置偵測時間有 150、450 和 750 個時脈時的不同閒置時間

因此，圖 5（a）顯示動態控制技術的閒置偵測時間（Tidledetect）太低的話（如 TIME150 及 TIME450），則可能造成不小的效能損失，但圖 5(b)顯示太高的閒置偵測時間(如 TIME750）將不能有效減少漏電流。我們已經探討閒置偵測時間對能量及效能取捨的影響，如所期望的，不同的閒置偵測時間不能在功能單元有不同閒置時間時，同時滿足有效減少漏電流及避免效能損失。

肆、結 論

減少功能單元的靜態功率消耗已是低功率處理器的必要方式，雖然快取記憶體在處理器中佔有相當大比例的面積，Butts 和 Sohi[2]提出一個靜態功率模型來估計不同電路的功率消耗，根據觀察，組合邏輯（static logic）跟記憶體（D Flip-flop、6T RAM cell 及 CAL cell 等）比較起來約有 10 倍的漏電流,儘管整數和浮點數的功能單元跟快取記憶體比起來有較少的電晶體，但在處理器的靜態功率消耗中佔了不小比例。

為使用低功率技術應避免影響電路性能，電源閘控技術需要注意其對系統效能的影響及可減少的漏電能量。而且各種應用程式在各功能單元有不同的閒置時間，所以現今主要的控制研究方向朝向低功率編譯器技術，因為編譯器可以事先分析程式結構，預測功能單元的閒置時間，於程式中插入適當的控制指令，執行時會根據這些控制指令決定適當的處理器工作模式；因此，利用編譯器技術可以針對功能單元有足夠長的閒置時間切斷電源，也可以在使用功能單元之前，預先做喚醒（pre-wakeup）的動作。相對的，動態控制技術利用閒置偵測時間（$T_{idledetect}$）來避免電源閘控使用在較短的閒置時間，往往很難兼顧減少足夠的漏電流及保持原有效能。

因此面對急遽增加的省能需求，一個有效的使用方式應可成功預測適合使用電源閘控的時間，而且可以事先喚醒；即使在閒置時間變動較大的程式，應可根據程式過去的行為，分析程式找出相關指令特徵，而可適時預測使用電源閘控及喚醒的時間。

參考文獻

[1] Y. Shin, J. Seomun, K.-M. Choi, and T. Sakurai, “Power gating: Circuits, design methodologies, and best practice for standard-cell VLSI designs,” ***ACM Transactions on Design Automation of Electronic System***, vol. 15, no. 4, 2010.

[2] J. Kao, A. Chandrakasan, and D. Antoniadis, “Transistor sizing issues and tool for multi-threshold cmos technology,” in *Proceedings of the 34th annual Design Automation Conference*. New York, NY, USA: ACM, 1997, pp. 409–414.

[3] S. Bhunia, N. Banerjee, Q. Chen, H. Mahmoodi, and K. Roy, “A novel synthesis approach for active leakage power reduction using dynamic supply gating,” in *Proceedings of the 42nd annual Design Automation Conference*. New York, NY, USA: ACM, 2005, pp. 479–484.

[4] M. D. Pant, P. Pant, D. S. Wills, and V. Tiwari, “An architectural solution for the inductive noise problem due to clock-gating,” in *Proceedings of the 1999 international symposium on Low power electronics and design*. New York, NY, USA: ACM, 1999, pp.255–257.

[5] A. Abdollahi, F. Fallah, and M. Pedram, "A robust power gating structure and power mode transition strategy for mtcmos design," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 15, no. 1, pp. 80–89, 2007.

[6] Y. Lee, D.-K. Jeong, and T. Kim, "Simultaneous control of power/ground current, wakeup time and transistor overhead in power gated circuits," in *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design.* Piscataway, NJ, USA: IEEE Press, 2008, pp. 169–172.

[7] S. W. Chung and K. Skadron, "On-demand solution to minimize I-Cache leakage energy with maintaining performance," *IEEE Transactions on Computers*, vol. 57, no. 1, pp. 7–24, 2008.

[8] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural techniques for power gating of execution units," in *Proceedings of the International Symposium on Low Power Electronics and Design*. New York, NY: ACM, 2004, pp. 32–37.

[9] A. Lungu, P. Bose, A. Buyuktosunoglu, and D. J. Sorin, "Dynamic power gating with quality guarantees," in ***Proceedings of the ACM/IEEE international symposium on Low power electronics and design***, 2009, pp. 377-382.

[10] A. Youssef, M. Anis, and M. Elmasry, "A comparative study between static and dynamic sleep signal generation techniques for leakage tolerant designs," ***IEEE Trans. Very Large Scale Integr. Syst.*** vol. 16, no. 9, pp. 1114-1126, 2008.

[11] Y. Meng, T. Sherwood, and R. Kastner, "Exploring the limits of leakage power reduction in caches," *ACM Transactions on Architecture and Code Optimization*, vol. 2, no. 3, pp. 221–246, 2005.

[12] Y.-P. You, C. Lee, and J. K. Lee, "Compilers for leakage power reduction," ***ACM Transactions on Design Automation of Electronic System***, vol. 11,no. 1, pp. 147-164, 2006.

[13] J. Adam Butts and Gurindar S. Sohi, "A static power model for architects," in *Proceedings of the International Symposium on Microarchitecture*, New York, NY, USA, 2000. ACM, pp. 191–201.