應用在即時工作之節能排程方法

吳卓俊　　潘怡文　　柯凱朧
屏東商業技術學院　屏東商業技術學院　屏東商業技術學院
資訊工程系　　資訊工程系　　資訊工程系
junwu@npic.edu.tw　pew6991@gmail.com　leon.kidd@gmail.com

摘要

節能省電是當前軟硬體設計的重要課題，隨著即時系統與 DVS 處理器技術的成熟，可兼顧系統時間限制與節能要求的即時排程方法也愈受重視。本論文考慮非理想化的 DVS 處理器，針對會存取共享資源的相依性即時工作，提出一個可視工作使用共享資源的情形動態決定其執行速度的即時節能排程方法。過去的相關研究大多針對動態優先權工作，並假設工作使用共享資源時是不可搶先的，本研究則針對排程成本較低的固定式優先權工作進行探討，並假設共享資源的使用是可搶先的。具體來說，本論文將結合著名的 rate monotonic 排程與 priority ceiling protocol 同步方法，提出一個速度調整協定，確保即時工作之可排程性及節能要求。經過一系列的實驗驗證，我們所提出的方法將可有效降低系統的電源消耗。

關鍵詞：節能排程、即時系統、DVS、相依性工作。

1. Introduction

在全球暖化議題受到關注與環保意識抬頭下，節能省電已成為軟硬體研發設計的首要課題。近年來新穎的處理器已支援動態電壓調整(dynamic voltage scaling，DVS)技術，可使用不同的速度來執行工作，以期能節省工作所耗用的電源。常見的 DVS 處理器包含 Intel StrongARM SA-1100[18]、Intel XScale PXA 27X[19]、Transmeta Crusoe TM5800[38]等皆屬之。

DVS 處理器可於執行階段視需要調整其供給電壓，讓系統內的工作可以在不同的速度執行，其耗電量與處理器的供給電壓之關係為 $P=KV^3$[4](其中 P 為耗電量，K 為常數，V 為供給電壓)。我們若能在不影響系統效能的前題下，儘量讓工作採用較低的執行速度，就可以達成節能省電的目標。具備效能保證的即時系統近年來亦廣受重視，一個即時工作通常伴隨 deadline 等時間限制，其工作的執行不但要求邏輯上的正確性，同時也必須滿足相關的時間限制。若採用 DVS 技術，即時系統只要能滿足工作的時間限制，儘可能地以較低速度(意即較少的耗電量)執行，就可以為系統帶來節能省電的好處。有鑑於此，即時系統的 DVS 節能排程便成為近年來相當重要的研究課題，許多設計良好的節能排程方法先後被發表[4,7,11,12,13,31,37,39,42,43]，然而對於在執行時需存取共享資源(shared resource)的相依性即時工作(dependent real-time task)的節能排程與並行控制卻少有探討。

本論文將針對相依性即時工作的 DVS 排程問題進行研究，提出的即時工作排程方法與同步控制方法，將能在保證系統效能的前題之下，大幅度地降低系統的耗電量。具體來說，我們考慮會以學者提出著名的固定式優先權 rate monotonic scheduling (RMS)[5]的方法來給定優先權及並行排程，並且採用另一篇著名的研究 priority ceiling protocol (PCP)[25]為共享資源的存取方法(resource access protocol)。並以著名的 critical section maximum speed (CSMS)[32,33]為基礎，並提出改進的方法規範工作存取共享資源時的處理器速度，進而達到節能的目的。

本文組織架構如下，第二節為介紹相關研究成果，第三節則就本論文所探討的問題及系統加以定義，第四節說明我們所提出的解決方案，第五節為效能評量，最後第六節為結論與未來研究。

2. Related Work

即時系統及其相關學理最早發展於 1970 年代，基於不同的系統或工作特性的考量，許多設計良好的方法先後被提出。例如，著名的 rate monotonic (RM)[5]、earliest deadline first (EDF)[5]及 least slack first (LSF)[1,20]等排程方法，以及著名的 priority inheritance protocol (PIP)[25],priority ceiling protocol (PCP)[25],

stack resource policy (SRP)[40]等並行控制方法。近年來，學者開始針對具備 DVS 處理器的系統，進行即時工作的排程與並行控制的相關研究。Yao[8]等人率先針對非週期性即時工作(aperiodic real-time task) 進行探討，提出一個 off-line 最佳演算法[8]，可在理想[1](ideal)的 DVS 處理器上為每個工作指定其執行的速度並進行排程，其排程結果被證明可確保所有工作都能滿足其時間限制，並使用最少的電源消耗。除了 off-line 的排程問題，學者們[8]亦提出了一些非週期性工作之排程 on-line 演算法。

後續學者們亦針對非理想的DVS處理(non-ideal DVS processor)進行非週期性即時工作排程的探討，並提出許多設計良好的節能排程方法[11,12,13,21,26,41,42]。除此之外亦有學者針對frame-based的非週期性即時工作進行探討，例如[4,21,31,39,41]。除了這些非週期性工作的節能排程方法外，在週期性的即時工作之節能排程方面，相關的研究亦是先探討ideal DVS processor[14,15]，再拓展到 non-ideal 的 DVS processor[7,16,17,22,28,29,37,43]。

雖然已有許多即時工作的 DVS 節能排程方法已被提出，但大部份的研究成果都是針對 independent 即時工作進行排程，近年來學者們已開始探討 dependent 即時工作的節能排程問題。所謂的 dependent 工作是假設工作在執行時會透過進入 critical section 以存取 shared resource，相關研究針對 critical section 的特性做不同的假設，例如可搶先的與不可搶先的。Jejurikar 與 Gupta[32,33]針對可搶先的 critical section，採用 fixed priority 的 RMS 排程方法，並使用 priority ceiling protocol[25]做為並行控制方法，提出了兩個節能排程方法：critical section at maximum speed (CSMS)與 constant static slowdown (CSS)。其中 CSMS 方法是將工作的執行分成兩種速度，當工作進入 critical section 時以最高速度執行，反之當工作未進入 critical section 時則以較低速度加以執行；至於 CSS 方法則是不區分 critical section 與 non-critical section，為工作的執行計算單一固定的速度，並以該速度執行所有的工作。後續 Jejurikar 與 Gupta 又將 CSMS 與 CSS 拓展應用至動態優先權的即時工作[34,35]，分別採用 EDF[5]及 dynamic priority ceiling protocol (DPCP)[6]做為優先權給定及並行控制方法，並且提出一個 frequency inheritance[35,36]的概念以解決採用 PCP 或 DPCP 做為同步控制方法時，所可能造成的違反 deadline 時間的問題。當處理器切換速度的成本不可忽略時，Chen 等人[44,45]將 frequency inheritance 的概念亦被拓展到應用到 PCP 與 SRP，以減少處理器切換速度的次數。此外亦有學者針對具有不可搶先的 critical section 的工作，進行相關的排程與並行控制之探討[2,9,10,23]。

本研究的主要目的是針對硬即時系統(hard real-time system)的相依性工作(dependent task)，進行DVS節能排程與同步控制方法之研究。考慮到執行階段的排程成本，我們將採用固定式優先權做為工作的優先權給定及排程方法；並考慮到提高系統的平行度，因此我們假設工作的執行是可搶先的，包含 critical section 與 non-critical section。綜觀過去的研究，只有Jejurikar等學者[32,33,34,35,36]所提出的方法是基於相同的假設，其中又以CSMS[32,33]是最能兼顧排程成本與效益的方法，本論文將以此方法為基礎，提出一個改進的方法稱為critical section at high speed(CSHS)，以在確保工作滿足時間限制的前題之下，將耗電量降至最低。

3. System Model and Definitions

本研究假設系統中有一組相依性的週期性即時工作(dependent periodic real-time task) $T=\{\tau_1, \tau_2, \cdots, \tau_n\}$的工作集合，其集合中每一個 $\tau_i \in T$ 可以表達為 $\tau_i=\{A_i, P_i, D_i, C_i\}$，其中 A_i、P_i 與 D_i 分別代表工作 τ_i 的到達時間(arrival time)、週期(period)與截限時間(deadline)，而 C_i則代表 worst-case 的運算時間(worst-case computation time),其值是當工作 τ_i 在系統中以最高速度執行時所耗費的時間。一個週期性即時工作 τ_i 從時間 A_i 到達系統並執行工作，每次時間 P_i的間隔就會再次進入系統並等待工作的執行，每次都會產生一個工作實例(instance，或稱為 Job，記作 $J_{i,j}$)，而每個 instance 在到達系統後必須在 deadline 前完成其工作執行。明顯地，而參數的給定必須滿足 $C_i \le D_i \le P_i$，我們並假設 $D_i = P_i$。當工作執行時會存取系統中一個或一個以

[1] 所謂 ideal 處理器，可任意指定處理器的工作電壓，反之 non-ideal 處理器僅提供固定各數的電壓。

上的共享資源(shared resource) $R=\{r_1, r_2, \cdots, r_m\}$，其使用 shared resource 的期間則被定義為 critical section。我們假設 resource 的使用必須透過 semaphore 進行存取 $L(r_j)$與 $UL(r_j)$，分別代表對共享資源 r_j進行 lock (鎖定)與 unlock (解鎖)的操作。此外令 $Z_i=\{Z_{i,1}, Z_{i,2}, \cdots, Z_{i,n_i}\}$為工作 τ_i所使用的 critical section，其中 $Z_{i,j}$代表工作 τ_i的第 j 個 critical section。至於在 DVS 處理器方面，我們考慮 non-ideal 的處理器，提供了 k 種不同的速度執行，並將速度遞增排序後表達為集合 $S=\{S_1, S_2, \cdots, S_k\}$，並且令 $S_{min}=S_1$、$S_{max}=S_k$分別代表為系統中最低的速度和系統最高速。在不失一般性下，我們令 S_{max}為 1，並對其它速度依序做正規化(normalization)。

本論文所探討的排程問題如下：給定一組 n 個可搶先的、相依的週期性即時工作 $T=\{\tau_1, \tau_2, \cdots, \tau_n\}$，一組 m 個 shared resources $R=\{r_1, r_2, \cdots, r_m\}$，以及一個支援 k 種速度 $S=\{S_1, S_2, \cdots, S_k\}$的 DVS 處理器。工作 τ_i在執行時會存取一個或一個以上的 shared resources，同時其對應的 critical section 也是可搶先的。此問題主要是要進行這些工作的排程，處理 shared resource 的同步，並且動態地決定工作執行時的處理器速度(其中包含 critical section 與 non-critical section)，其所產生的排程必須使每個工作都能符合時間限制(也就是不允許任何工作在其 deadline 後才完成)，同時還必須使系統的電源消耗降到最低。

此一問題若不考慮最佳化耗電量的目標，單純就相依性的即時工作排程與同步問題就已被證明為 NP-hard[1, 24, 27]，因此本問題當然也是 NP-hard。本研究將針對即時工作的優先權給定方式、排程策略、共享資源的並行控制方法以及處理器速度的給定等研究議題分別加以探討，並提出可行的方法進行排程。

4. Our Approach

我們會針對工作先行計算出一個較低的處理器速度 S^{base}，待發生 priority inversion[2] 異常現象時，則動態地調整工作的執行速度，使其仍能滿足工作的時間限制。後續我們將就 S^{base} 的計算與動態調整處理器速度的方法及共享資源的並行控制等進行探討。

4.1 Base Speed Assignment

首先，我們將計算出工作的執行所採用的處理器速度，其計算是假設工作沒有發生任何阻擋的情況下，依據 RM 的可排程性分析方法，制訂出可確保所有工作都能排程的方法。Liu 與 Layland[5]指出，對一組 independent 週期性工作，若能滿足下列式子，則採用 RMS 一定可排程[5]：

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \le n(2^{\frac{1}{n}} - 1) \qquad (1)$$

參考公式(1)，我們可制定出一個可確保可排程性的最低處理器速度 S^{base}：

$$S^{base} = \min_{S_j \in S} \left\{ S_j \mid \sum_{i=1}^{n} \frac{C_i}{T_i} / n(2^{\frac{1}{n}} - 1) \le S_j \right\} \qquad (2)$$

4.2 Priority-Driven Scheduling and Concurrency Control

本研究將採用 priority-driven 的方式進行排程工作，在工作的優先權給定方面，是採用 Liu 與 Layland[5]所提出的 RMS 方法，依據工作週期來給定，週期越短者工作的優先權愈高，反之愈短優先權越短(若且為若當工作 τ_i 的週期小於 τ_j, 優先權 τ_i 大於 τ_j)。我們將工作 τ_i的優先權定義為 $P(\tau_i)$。並採用 Sha 等人[25]所提出的 PCP 方法做為並行控制方法。PCP 首先對每個 shared resource r_j，給定一個 priority ceiling $PL(r_j)$，其值為所有會存取到 r_j的工作當中優先權最高者之優先權。當一個工作 τ_i 要存取 r_j 時，必須先透過對應的 semaphore 進行 $L(r_j)$操作，待成功取得使用權後才能進入其 critical section，PCP 方法使用下列的條件來決定是否讓工作取得 shared resource 的使用權：

1. r_j必須是處於 unlock 的狀態。
2. τ_i 的優先權必須大於系統內被其它工作 lock 住的 resource 中，其 priority ceiling 最高者。

此外，PCP 也規範當 priority inversion

[2] 所謂的 priority inversion 現象是指低優先權的工作阻擋了高優先權的工作的異常現象。

現象發生時，阻擋者必須暫時繼承被阻擋者的優先權，直到 priority inversion 現象結束為止。

4.3 Dynamic Speed Adjustment Method – Critical Section at High Speed (CSHS)

Jejurikar 等人[32,33]所提出的 CSMS 方法，會強制使用 S_{max} 做為所有 critical section 的執行速度[3]，本論文參照此一做法，但為節省系統的耗電量，我們提出一個 critical section at high speed(CSHS)方法，將 CSMS 方法做了以下兩點改進：

1. 當工作進入 critical section 時，不再永遠使用最高速度執行，而是先檢查是否發生 priority inversion 的現象，再進行速度的改變。

2. 為確保工作的可排程性，因此當工作進入 critical section 後，若發生 priority inversion 現象時，指定採用 S^{high} 速度來執行阻擋者以及被阻擋者剩餘的工作。

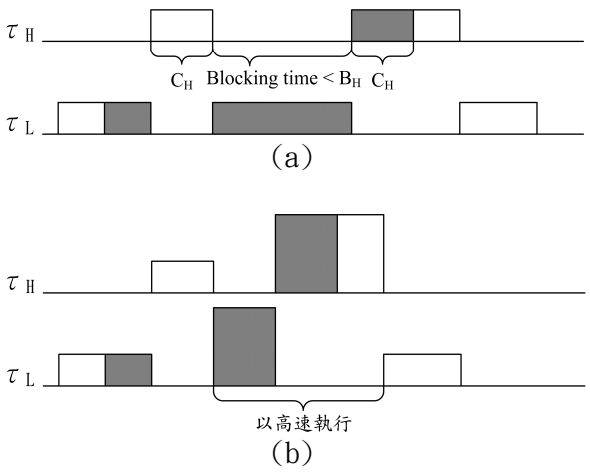


圖 1 CSHS 範例

請參考圖 1(a)，假設一個高優先權的工作 τ_H 被一個較低優先權的工作 τ_L 所阻擋，τ_H 已執行完成及尚未執行完成的剩餘工作分別定義為 C_H' 及 C_H''，τ_H 的 worst case 的被阻擋時間為 B_H。為使工作 τ_H 縱使發生了 priority inversion 並且其阻擋時間為 worst case 的阻擋時間，仍能確保其可排程性，我們必須找到下列的 S^{high}：

$$S^{high} = \min_{S_j \in S} \left\{ \frac{S^{base}(C_H'' + B_H)}{C_H''} \le S_j \right\} \quad (3)$$

在執行階段，當一個工作 τ_i 試圖 lock 資源 r_j 時，除使用 PCP 決定是否允許其進入 critical section 外，我們並以下列方法動態地改變工作的執行速度。

Algorithm 1: Critical Section at High Speed (CSHS)

當工作 τ_i 無法 lock 資源 r_j 時(依據 PCP):

```
1: if (P(τi) > P(τcurrent)) //發生 priority inversion
2:     S^high ← S_max
3:     for j = 1 to k
4:         if ( S^base(C''_H + B_H) / C''_H ≤ S_j )
5:             S^high ← S_j
6:             break;
7:         end if
8:     end for
9:     Speed(τi) ← S^high
10:    Speed(CS(τcurrent)) ← S^high
11:end if
```

其中 P(τ)為工作 τ 的優先權，Speed(τ)為指定給工作 τ 的速度，CS(τ)為工作的 critical section，$\tau_{current}$ 則是目前執行中的工作(也就是阻擋者)。Algorithm 1 的第 1 行即為判定是發生 priority inversion 現象，後續第 2-8 行則是計算出 S^{high}。第 9 行及第 10 行則分別指定被阻擋者與阻擋者的執行速度。其中被阻擋者是所有剩餘的執行時間都以 S^{high} 執行，而阻擋者只有其 critical section 所剩餘的執行時間以 S^{high} 執行。要特別注意的是，這些速度的改變僅限於目前的工作

[3] CSMS 在 non-critical section 所使用的速度與本論文所使用的 S^* 的計算方式並不相同，但本論文基於 CSMS 相同的概念，針對 critical section 與 non-critical section 設計兩種不同的執行速度。

instance，下一次工作再次到達時仍使用速度 S^{base} 執行；同時原先在 PCP 中的優先權繼承等方式仍照常使用。圖 1(b)為 CSHS 方法的一個排程範例，對照圖 1(a)在 Blocking time 區間中我們以最高速進行工作執行，並且保證工作的可排程性。

5. Performance Evaluation

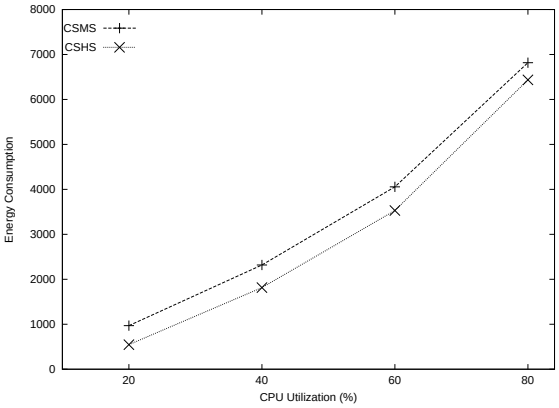
本節將探討本論文所提出之 CSHS 方法在效能方面的表現。我們首先建置一個模擬環境具有固定優先權與並行控制存取資源的機制，進行工作的模擬排程，此外也具備有 DVS 處理器速度的調整方式，對處理速度上我們假設有從 0.1 到 1 等不同速度，具體來說 $S_{min} = S_1 = 0.1$， $S_2 = 0.2$，…，$S_{10} = S_{max} = 1$。

在本研究中我們關注的重點在於工作在系統中執行的能源消耗，我們假設一個電壓和 $\{V_1, V_2, \cdots, V_k\}$ 和速度 $\{S_1, S_2, \cdots, S_k\}$ 的關係集合，其中電壓和耗電量的關係為 $P=KV^3$[4](其中 P 為耗電量，K 為常數，V 為供給電壓)。我們會以工作所使用的速度所對應的電壓，求得其耗電量，並將所有工作執行的耗電量加總。

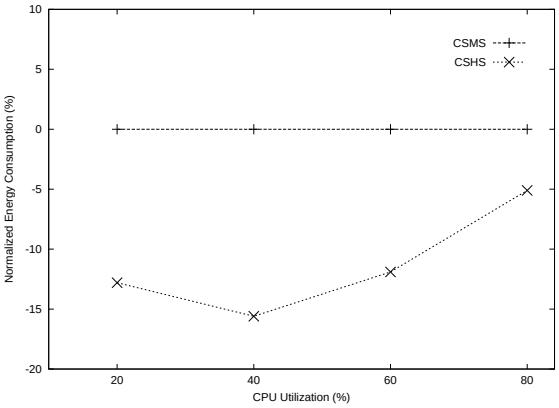
而在 workload 方面，我們主要針對兩類工作，一類為共享資源使用率高的工作集合(以下稱為高資源工作使用率集合)，另一類為低的工作集合(低資源工作使用率集合)，相關參數說明如下：CPU 的使用率從 20%到 80%，工作週期範圍從 500 到 2000，工作的計算時間從 10 到 200 之間，工作使用共享資源的數量 0 到 5，系統模擬時間 10000，共享資源使用率在低資源使用率集合中佔 15%以下，高資源使用率集合中佔 25%到 30%。

實驗並實作了 Jejurikar 與 Gupta[32, 33] 的 CSMS 方法，與我們所提出的 CSHS 進行效能的驗證。圖 2 顯示了高資源工作使用率集合的工作之耗電量，可觀察到不論在何種 CPU 使用率情況下(從 20%到 80%)，CSHS 在耗電量方面都優於 CSMS，其中圖 2(b)是以 CSMS 為基準，CSHS 的耗電量較 CSMS 減少約 5%到 15%之間。圖 2 則顯示了低資源工作使用率集合的工作之耗電量，可觀察到 CSHS 在耗電量方面仍優於 CSMS，CSHS 的耗電量較 CSMS 減少約 5%到 10%之間。

經由我們的實驗結果可得知 CSHS 在各種情況下都優於 CSMS，其主要原因就在於 CSMS 永遠將 critical section 加速，並且其加速都是使用系統的最高速度。相對的，CSHS 僅於需要時(意即發生 priority inversion 時)才進行增速，同時我們所採用的加速度並不是系統的最高速度，也就因為這些原因 CSHS 的效能較 CSMS 來得好。



(a) Energy Consumption



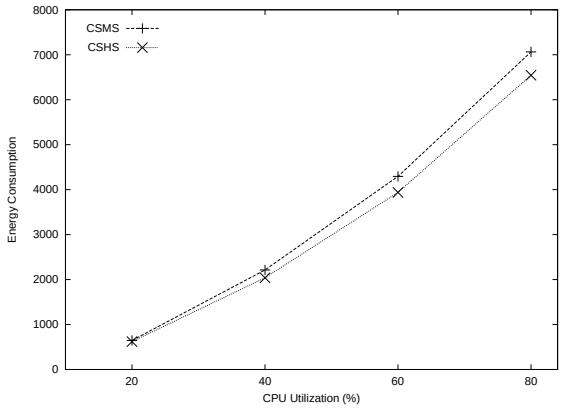
(b) Normalized Energy Consumption

圖 2 高資源使用率工作集合的耗電量

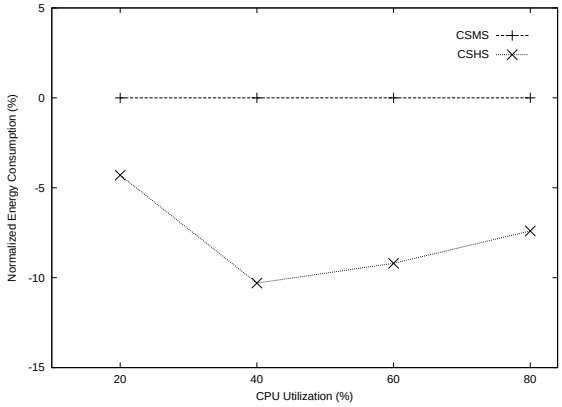
6. Conclusion and Future Work

在本研究中，我們以著名的 RMS 與 PCP 為基礎，提出一個即時系統的執行速度調整方法，可有效減少系統的能源耗損。由於當處理器以高速執行工作時會造成大量的功率耗損，而我們所採用的策略是減少系統在高速度的時間，並且妥善地決定系統在執行工作時的速度，以降低耗電量。我們並且就所提出的方法之特行進行分析，可保證工作必定可在其 deadline 前完成排程。在未來研究工作的拓展方面，我們將繼續此一研究主題，並將發展更精確的速度給定方法，以求工作能使用更低的處理器速度來執行，以獲取更大幅度的能源節

省。我們也將考慮更多不同的工作或系統特性，例如動態優先權的工作、有執行順序的工作、使用不同特性的共享資源等，並發展相關的節能排程方法。



(a) Energy Consumption



(c) Normalized Energy Consumption

圖 3 低資源使用工作率集合的耗電量

誌謝

本研究部份經費由國科會計畫 NSC 98-2815-C-251-003-E 輔助，在此誌謝。

參考文獻

[1] A. K. Mok. Fundamental Design Problems for the Hard Real-Time Environment. Ph.D. dissertation, MIT, Cambridge MA, 1983.

[2] Abdullah M. Elewi, Medhat H. A. Awadalla, and Mohamed I. Eladawy. Energy-efficient multi-speed algorithm for scheduling dependent real-time tasks. In ***Proceedings of the International Conference on Computer Engineering and Systems (ICCES 2008)***, pages 237 – 242, Cairo, Egypt, November 2008.

[3] Abdullah M. Elewi, Medhat H. A. Awadalla, and Mohamed I. Eladawy. Energy efficient real-time scheduling of dependent tasks sharing resources. In ***Proceedings of the 2008 High Performance Computing and Simulation Conference (HPCS)***, pages 107 – 116, Nicosia, Cyprus, June 3-6 2008.

[4] C.-M. Hung, Jian-Jia Chen, and Tei-Wei Kuo. Energy-efficient real-time task scheduling for a dvs system with a non-dvs processing element. In ***Proceedings of the 27th IEEE Real-Time Systems Symposium (RTSS)***, 2006.

[5] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. ***Journal of the Association for Computing Machinery (JACM),*** 20(1):46 – 61, January 1973.

[6] Chen and Lin. Dynamic priority ceilings: A concurrency control protocol for real-time systems. Technical Report UIUCDCS-R-89-1511, Department of Computer Science, University of Illinois at Urbana-Champaign, 1989.

[7] E. Bini, G. Buttazzo, and G. Lipari. Speed modulation in energy-aware real-time systems. In ***Proceedings of the 17th EuroMicro Conference on Real-Time Systems (ECRTS)***, pages 309 – 318, 2005.

[8] Frances Yao, Alan Demers, and Scott Shenker. A scheduling model for reduced cpu energy. In ***Proceedings of the 36th IEEE Annual Symposium on Foundations of Computer Science (FOCS'95)***, pages 374 – 382, 1995.

[9] Fan Zhang and Samuel T. Chanson. Processor voltage scheduling for real-time tasks with non-preemptible sections. In ***Proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS'02)***, pages 235 – 245, 2002.

[10] Fan Zhang and Samuel T. Chanson. Blocking-aware processor voltage scheduling for real-time tasks. ***ACM Transactions on Embedded Computing Systems***, 3(2):307 – 335, May 2004.

[11] G. Quan and X. Hu. Minimum energy fixed-priority scheduling for variable voltage processor. In ***Proceedings of the Design Automation and Test Europe Conference*** , pages 782 – 787, 2002.

[12] G. Quan and X. Hu. Energy efficient

fixed-priority scheduling for real-time systems on variable voltage processors. In ***Proceedings of the 38th Conference on Design Automation***, pages 828 – 833, 2001.

[13] H.-S. Yun and J. Kim. On energy-optimal voltage scheduling for fixed-priority hard real-time systems. ***ACM Transactions on Embedded Computing Systems***, 2(3):393 – 430, August 2003.

[14] H. Aydin, R. Melhem, D. Moss, and P. Mejia-Alvarez. Determining optimal processor speeds for periodic real-time tasks with different power characteristics. In ***Proceedings of the IEEE EuroMicro Conference on Real-Time Systems***, pages 225 – 232, 2001.

[15] H. Aydin, R. Melhem, D. Moss, and P. Mejia-Alvarez. Dynamic and aggressive scheduling techniques for power-aware real-time systems. In ***Proceedings of the 22nd IEEE Real-Time Systems Symposium (RTSS)***, pages 95 – 105, 2001.

[16] H. Aydin, R. Melhem, D. Moss, and P. Mejia-Alvarez. Power-aware scheduling for periodic real-time tasks. ***IEEE Transactions on Computers***, 53(5), 2004.

[17] H. Aydin and D. Zue. System-level energy management for periodic real-time tasks. In ***Proceedings of the IEEE Real-Time Systems Symposium (RTSS)***, 2006.

[18] Intel. Strong ARM SA-1100 Microprocessor Developer's Manual, 2003.

[19] Intel. Intel XScale Core Developer's Manual, 2004.

[20] J.Y.T. Leung and J. Whitehead. On the complexity of fixed-priority scheduling of periodic real-time tasks. ***Performance Evaluation***, 2(4):237 – 250, December 1982.

[21] Jian-Jia Chen and Tei-Wei Kuo. Voltage-scaling scheduling for periodic real-time tasks in reward maximization. In ***Proceedings of the 26th IEEE Real-Time Systems Symposium (RTSS)***, pages 345 – 355, 2005.

[22] Jian-Jia Chen, Tei-Wei Kuo, and C.-S. Shih. 1+ □ approximation clock rate assignment for periodic real-time tasks on a voltage-scaling processor. In ***Proceedings of the 2nd ACM Conference on Embedded Software (EMSOFT)***, pages 247 – 250, 2005.

[23] Jaewoo Lee, Kern Koh, and Chang-Gun Lee. Multi-speed dvs algorithms for periodic tasks with non-preemptible sections. In ***Proceedings of the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA2007)***, 2007.

[24] J. A. Stankovic, M. Spuri, M. D. Natale, and G. Buttazzo. Implications of classical scheduling results for real-time systems. I***EEE Transactions on Computers***, 28(6):16 – 25, June 1994.

[25] L. Sha, R. Rajkumar, and J. P. Lehoczky. Priority inheritance protocols: An approach to real-time synchronization. ***IEEE Transactions on Computer***s, 39(9):1175 – 1185, Sepember 1990.

[26] M. Li and F. Yao. An efficient algorithm for computing optimal discrete voltage schedules. ***SIAM Journal on Computing***, 35(3):658 – 671, 2005.

[27] M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, 1979.

[28] P. Mejia-Alvarez, E. Levner, and D. Moss . Adaptive scheduling server for power-aware real-time tasks. ***ACM Transactions on Embedded Computing Systems***, 3(2):284 – 306, 2004.

[29] P. Pillai and K. G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In ***Proceedings of the 18th Symposium on Operating Systems Principles***, 2001.

[30] R. Shah and J. Rabaey. Energy aware routing for low energy ad hoc sensor networks. In ***proceedings of the 2002 Wireless Communications and Networking Conference (WCNC2002)***,page 350-355,March 2002.

[31] R. Xu, D. Mosse, and R. G. Melhem. Minimizing expected energy in real-time embedded systems. In ***Proceedings of the International Conference on Embedded Software (EMSOFT)***, pages 251 – 254, 2005.

[32] R. Jejurikar and R. Gupta. Energy aware task scheduling with task synchronization for embedded real time systems. In ***Proceedings of the International Conference on Compilers, Architecture and Synthesis for Embedded Systems(CASES)***, pages 164 – 169, 2002.

[33] Ravindra Jejurikar and Rajesh K. Gupta.

Energy aware task scheduling with task synchronization for embedded real-time systems. Technical Report 02-12, Center for Embedded Computer Systems, Department of Information and Computer Science, University of California at Irvine, June 21 2002.

[34] Ravindra Jejurikar and Rajesh K. Gupta. Energy aware edf scheduling with task synchronization for embedded real-time operating systems. Technical Report 02-24, Department of Information and Computer Science, University of California at Irvine, August 2002.

[35] R. Jejurikar and R. Gupta. Energy aware task scheduling with task synchronization for embedded real time systems. ***IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems***, 25(6):1024 – 1037, 2006.

[36] Ravindra Jejurikar, Cristiano Pereira, and Rajesh K. Gupta. Dual-mode frequency inheritance algorithm for energy aware task scheduling with task synchronization. Technical Report 03-07, Center for Embedded Computer Systems, Department of Information and Computer Science, University of California at Irvine, Feburary 28 2003.

[37] S. Saewong and R. Rajkumar. Practical voltage-scaling for fixed-priority-systems. In ***Proceedings of the 9th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)***, pages 106 – 115, 2003.

[38] Transmate. The Technology Behind Crusoe Processors - Low-Power x86-compatible Processors Implemented with Morphing Software, 2000.

[39] T. Ishihara and H. Yasuura. Voltage scaling problem for dynamically variable voltage processors. In ***Proceedings of the International Symposium on Low Power Electronics and Design***, pages 197 – 202, 1998.

[40] T. P. Baker. A stack-based resource allocation policy for real-time processes. In ***Proceedings of the IEEE 11th Real-Time Systems Symposium (RTSS***), pages 191 – 200, Lake Buena Vista, Florida, USA, December 4-7 1990.

[41] W.-C. Kwon and T. Kim. Optimal voltage allocation techniques for dynamically variable voltage processors. In ***Proceedings of the 40th Design Automation Conference***, pages 125 – 130, 2003.

[42] Y. Shin and K. Choi. Power conscious fixed priority scheduling for hard real-time systems. In ***Proceedings of the 36th ACM/IEEE Conference on Design Automation Conference*** , pages 134 – 139, 1999.

[43] Y. Shin, K. Choi, and T. Sakurai. Power optimization of real-time embedded systems on variable speed processors. In ***Proceedings of the 2000 IEEE/ACM International Conference on Computer-Aided Design***, pages 365 – 368, 2000.

[44] Ya-Shu Chen, Chuan-Yue Yang, and Tei-Wei Kuo. Fl-pcp: Frequency locking for energy-efficient real-time task synchronization. In ***Proceedings of the 13th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA2007)***, 2007.

[45] Ya-Shu Chen and Tay-Jyi Lin. Voltage emergence prevention for energy-efficient real-time task synchronization. In ***Proceedings of the 8th IEEE International Conference on Computer and Information Technology Workshops***, pages 545 – 550, 2008.