

リークモニタを用いた細粒度パワーゲーティングの実行時制御を行う Linux プロセススケジューラ

小柴 篤史^{*1}, 坂本 龍一^{*1}, 佐藤 未来子^{*1},
小坂 翼^{*2}, 宇佐美 公良^{*2}, 天野 英晴^{*3}, 近藤 正章^{*4}, 中村 宏^{*4}, 並木 美太郎^{*1}
東京農工大学^{*1}, 芝浦工業大学^{*2}, 慶應義塾大学^{*3}, 東京大学^{*4}

1 緒言

プロセッサの処理性能向上に伴い, LSI の消費電力の増大, 特にリーク電力の増加が問題となっている [1]. 我々はリーク電力の削減によるプロセッサの省電力化を目標に研究を進めており, 削減手法としてパワーゲーティング (PG) 技術に注目している.

PG とは使わない回路への電源電圧を遮断 (スリープ) することで, リーク電力を削減する技術だが, 電源の切替え時にオーバヘッド電力が発生する. より高効率に電力を削減するためにはオーバヘッド電力を考慮した PG 制御を行う必要がある. また, リーク電力はチップ温度に依存して変化するため, チップ温度ごとに適した制御を行うことが重要となる.

我々はこれまでの研究で, オーバヘッドによる電力増加を抑えるように PG 制御を行う機構を Linux スケジューラ内に実装し, シミュレーションによる電力削減効果を得た. 本研究では, この PG 制御方式を実チップ環境で実現し, 電力削減効果を検証する. 本研究の特徴として, シミュレーション環境では存在しなかった温度デバイスにリークモニタを使用する. チップ温度を取得する機能を Linux スケジューラに実装し, PG 制御機構と連携して制御を行うことで, 実行時のチップ温度に適した PG 制御による電力削減を目指す.

2 アーキテクチャ

2.1 Geyser[2]

Geyser は, MIPS R3000 アーキテクチャを基に作成された低電力プロセッサであり, 細粒度パワーゲーティング機能を有している. Geyser はソフトウェアからの PG 制御を可能にするための制御用レジスタを備えており, 四つの演算ユニット (ALU・SHIFT・MULT・DIV) それぞれに対して, ユニット未使用時の挙動 (スリープポリシー) を設定できる. 設定可能なスリープポリシーは以下の 3 種類である.

- (1) 動的 PG: ユニット未使用時は常にスリープ
- (2) キャッシュミス時スリープ: キャッシュミスが発生した時のみスリープ
- (3) 常にアクティブ: スリープせず, 常に電源供給

2.2 リークモニタ

Geyser は, 回路内で発生するリーク電力量が一定値に達するまでの時間 (Detection Time, DT) を計測で

きるリークモニタを備えている. リーク電力の発生量はチップ温度に依存して変化するため, DT の値も温度によって異なる. 従って, チップ温度ごとの DT の値が既知であれば, 逆に DT の値からチップ温度を定められる. 本研究では, この DT の値に基づく温度情報を用いて PG 制御を行う.

3 細粒度 PG の制御方式

PG による電源切替えで発生するオーバヘッド電力がスリープで削減できるリーク電力より大きい場合は, PG を行うことで逆に電力が増加してしまう. リーク電力の削減量はスリープの長さ依存し, PG が有効に働くスリープの長さの指標を BEP と呼ぶ. リーク電力の特性上, BEP も温度によって変化する.

筆者らはこれまでに, BEP 未満のスリープが頻出する場合は PG を行わないようにスリープポリシーを切り替える制御方式を提案しており, 汎用 OS である Linux のスケジューラへ実装した PG 制御機構により, シミュレーションによる電力削減効果を得ている. そこで本研究では, 実チップ環境に本制御方式を適用し, 実チップでの PG 制御方式とチップ温度の追従による電力削減効果を検証する.

4 システム設計

4.1 リークモニタによるチップ温度の取得

リークモニタから得た DT の値を温度に変換し, 実行時のチップ温度を取得する. 手順を図 1 に示す. まず, DT とチップ温度の対応表をカーネルの内部情報として保持しておく. カーネルはリークモニタに DT の計測を指示し, 実行時の DT を取得する. 取得した DT と表の DT を比較し, DT に対応する温度を定める.

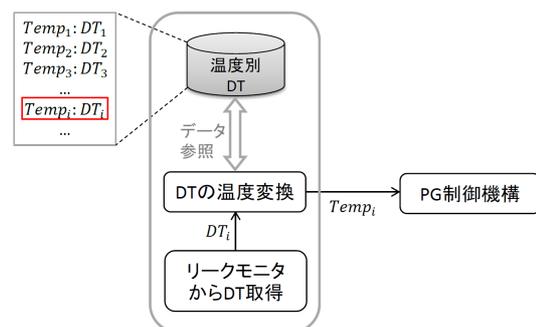


図 1: チップ温度取得の手順

4.2 スケジューラへの温度取得処理の組み込み

本研究では、Linux のプロセススケジューラにチップ温度取得処理を追加し、3 節で述べた PG 制御機構と連携して動作させる。

チップ温度取得処理を追加した PG 制御の流れを図 2 に示す。プロセススケジューラにおいて、まずリークモニタからチップ温度を取得し、その後 PG 制御機構を呼び出す。制御機構ではチップ温度に応じて BEP の値や閾値を変更しつつ、スリープポリシーを決定する。

スケジューラは、プロセスごとにスリープポリシーを算出し、プロセスコンテキストへ決定したスリープポリシーを保持しておく。その後は通常のスケジューリング処理を行った後、次プロセスのコンテキストを参照して格納しておいたスリープポリシーを各演算ユニットに適用し、コンテキストスイッチを行う。

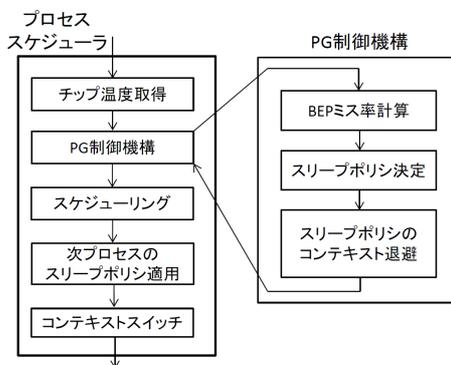


図 2: PG 制御の流れ

5 実装と評価

Geyser チップが接続された基板、および Geyser のスリープ頻度情報を取得するハードウェアを実装した ML605 ボードで構成される実行環境を用いて、提案方式の電力評価を行った。基板に取り付けた電流計を使い、評価対象のベンチマーク (adpcm, fft, lame) 実行中の平均消費電力を計測した。提案方式の効果を検証するため、次に示す 3 種類の PG 制御方式における消費電力を比較する。

- (1) OS による PG 制御なし
- (2) PG 制御機構による制御 (温度情報固定)
- (3) 提案方式: PG 制御機構+チップ温度取得処理

各制御方式を適用した場合の平均消費電力を図 3 に示す。OS による PG 制御を行わない方式 (方式 1) と比較して、提案方式では、チップ温度 25°C 環境において平均 12.8%、最大 17.5% の消費電力削減、チップ温度 55°C 環境において平均 7.1%、最大 10.3% の消費電力削減、チップ温度 85°C 環境において平均 2.2%、最大 5.2% の消費電力削減に成功した。これにより、実チップ環境でも提案方式による PG 制御が有効であることが確かめられた。チップ温度が高くなるほど電力削減効果が小さくなっているが、これはチップ温度の上昇

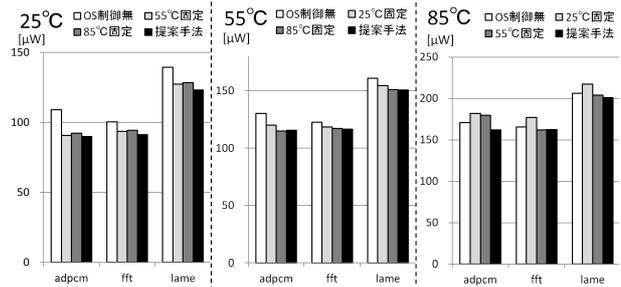


図 3: ベンチマーク別平均消費電力

によって BEP が短くなり、OS の制御によって抑制できるオーバヘッド電力が少なくなったためである。

また、実際のチップ温度と異なる温度情報を用いた制御方式 (方式 2) と比較した場合も、提案方式の電力削減量は、それぞれのチップ温度環境において最も大きくなった。特に、チップ温度 85°C 環境においては、25°C 向けの制御では平均 6.3% の電力増加、55°C 向けの制御では adpcm において 5.1% の電力増加が生じた。このように、誤った温度情報を用いた PG 制御では逆に電力が増加する可能性があることが分かり、チップ温度の追従による電力削減効果が確かめられた。

提案方式の適用による処理オーバヘッドにより、アプリケーションの実行時間は非適用時と比較して 0.18% 増加した。しかし、この実行時間の増加に伴う消費エネルギーの増加量は、PG 制御による削減量に対して 1.0 ~ 8.8% 程度と小さい。したがって、実行時間の増加を考慮しても提案方式は電力削減に有効であると言える。

6 結言

本研究では、OS による PG 制御方式を実チップ環境に適用し、提案方式が電力削減に有効であることを確認した。チップ温度によって変化するリーク電力の特性に対応するため、リークモニタを用いて OS がチップ温度を取得する方式を提案し、チップ温度の追従により更なる電力削減が可能であることを明らかにした。

今後の課題としては、チップの製造ばらつきによりチップ特性が異なることを考慮し、OS のスケジューラで保持する BEP や閾値をチップごとに最適化した場合の評価や、PG 制御を行う時間粒度を変更した場合の電力削減効果の検証などが挙げられる。

参考文献

- [1] F. Fallah and M. Pedram, "Standby and Active Leakage Current Control and Minimization in CMOS VLSI Circuits.", *IEICE Trans. on Electronics, Special Section on Low-Power LSI and Low-Power IP*, Vol. E88-C, No.4 Apr.2005, pp. 509-519.
- [2] 白井利明・香嶋俊裕・武田清大・中田光貴・宇佐美公良・長谷川揚平・関直臣・天野英晴著『ランタイムパワーゲーティングを適用した MIPS R3000 プロセッサの実装設計と評価 (低消費電力化技術)』芝浦工業大学・慶応義塾大学 / 情報処理学会研究報告. SLDM, [システム LSI 設計技術] 2008(2), 43-48, 2008-01-16