社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.
ICD2001-32 (2001-05)

低電力実時間組込システムのための OS，アプリケーション，ハードウェア協調による CVS(Cooperative Voltage Scaling)と電圧ホッピング

川口 博，張 綱，*李 誠洙，辛 英洙，桜井 貴康
kawapy@iis.u-tokyo.ac.jp
東京大学国際・産学共同研究センター 東京都目黒区駒場 4-6-1
*梨花女子大学情報通信学科 韓国ソウル市西大門区大峴洞 11-1

あらまし ソフトウェアによってシステムが実現されるにしたがい，実時間組込システムの低電力設計はさらに重要になってきている．この論文では OS，アプリケーション，ハードウェアの協調による低電力化方法について述べる．この方法で従来の定電源電圧による rate-monotonic スケジューリングに比べ 1/4 以下にまで電力を低減できる．電力低減は実時間性を損なうことなく実現される．

キーワード 低電力，組込システム，リアルタイム OS，アプリケーションスライシング，電圧ホッピング

Cooperative Voltage Scaling (CVS) and V_{DD}-Hopping among OS, Applications and Hardware for Low-Power Real-Time Embedded Systems

Hiroshi Kawaguchi, Gang Zhang, *Seongsoo Lee, Youngsoo Shin, and Takayasu Sakurai
kawapy@iis.u-tokyo.ac.jp
Center for Collaborative Research, University of Tokyo
4-6-1, Komaba, Meguro-ku, Tokyo 153-8505, JAPAN
*Department of Information Electronics, Ewha University
11-1 Daehyun-dong, Seodaemun-gu, Seoul, KOREA

Abstract Power-efficient design of real-time embedded systems becomes more important as system functionality is increasingly realized by software. This paper presents a cooperative power-optimization scheme among an operation system, application programs and hardware. This scheme is shown to reduce the power to less than 1/4 compared to the conventional rate-monotonic scheduling with fixed supply voltage. The power saving is achieved without degrading the real-time features.

key words Low Power, Embedded System, Real-Time OS, Application-Slicing, V_{DD}-Hopping

1. はじめに

携帯電話や PDA などの携帯システムのために LSI には高性能と低電力の両立が求められている．負荷が最大定格より小さい場合にプロセサの周波数 f と電源電圧 V_{DD} を動的に制御することにより低電力化を達成する方式がある[1]-[6]．所望の f を与えるとハードウェア帰還によりアナログ的多段階の V_{DD} がプロセサに供給される．これらはプロセスばらつきには強いが，クリティカルパスを模したハードウェアの追加が必要であり，市販プロセサに適応する場合に再設計が必要となる．またソフトウェアの観点からはシステムに課せられた時間制約の下で電力が最小となるようにアプリケーションと OS を設計することが問題となっている．

この論文では市販プロセサを用いた組込システム上でのアプリケーションと OS の協調による低電力設計法について述べる．これを CVS (cooperative voltage scaling) と呼ぶ．図 1に示すとおり，CVS は OS，アプリケーション，市販プロセサと ASIC で構成される．

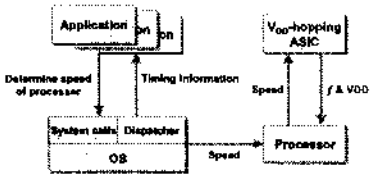

図1. CVS 構成図.

OS はアプリケーションに対して時間情報を与え，アプリケーションはその時間情報を f と V_{DD} を変化させるために利用する．それぞれのアプリケーションは自身の実行時間に関する知識をもっている一方で，OS だけがアプリケーション間の動的な時間情報を持っている．すなわちアプリケーションと OS が協調して時間情報を処理すべきである．CVS においてアプリケーションは起動周期と WCET (worst-case execution time)を持ったタスクの集合とする．タスクは RMS(rate-monotonic scheduling)のように優先度固定でスケジュールされるが，他のアルゴリズムも用いることができる[7].

f と V_{DD} を変化させるために電圧ホッピング(V_{DD}-hopping)技術を用いる[8]-[9]．負荷が低下すると f と V_{DD} も低下させ，これにより電力を低減させる．電圧ホッピングは新たなハードウェアを外部に追加するだけで市販プロセサに適応可能である．これはハードウェア帰還を持たず，V_{DD} を 2 段階まで減らしているためである．さらに言えば V_{DD} の段階が少ないとそれだけ少ない製造テストで済む．コストの上昇を抑えるためにも V_{DD} の段階を減らすことは重要である．

2. 電圧ホッピングとアプリケーションスライシング

図2は 50%負荷における 3 とおりのプロセサ制御方式を示している．(A)と(B)は従来の方式であり，V_{DD} は固定され，タスクを実行している時間のみが制御される．一方(C)の電圧ホッピングでは f と V_{DD} が動的に制御される．消費電力は f と V_{DD}^2 に比例するので電圧ホッピングが最も効果的に消費電力を低減できる．

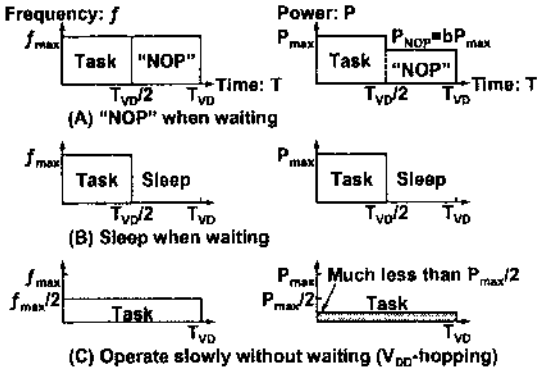

図2. 50%負荷におけるプロセサ制御方式.

(A) 暇なときは"NOP"する：実行すべきタスクがない場合，次のタスクまでプロセサは"NOP"する．それでも PLL/DLL，キャッシュメモリ，アドレス演算器は動作し，一定電力 bP_{max} を消費する．ここで b は 1 以下である．規格化負荷を NW とすると，規格化電力 NP は以下のとおり与えられる．

$$NP(NW) = (1-b)NW + b.$$

(B) 暇なときはスリープする：スリープ可能なプロセサの場合はタスク終了から次のタスクまでスリープを利用できる．スリープにおいて電力は消費されないとすると，NP は以下のとおり与えられる．

$$NP(NW) = NW.$$

(C) できる限りゆっくりとする：これが電圧ホッピングである．NW と NP は V_{DD} の関数として以下のとおり与えられる[10].

$$NW(V_{DD}) = \frac{V_{DD\,max}}{V_{DD}}\left(\frac{V_{DD}-V_{TH}}{V_{DD\,max}-V_{TH}}\right)^{\alpha},$$

$$NP(V_{DD}) = \left(\frac{V_{DD}}{V_{DD\,max}}\right)^{2} NW(V_{DD}),$$

$$NP(NW) = NW^{\frac{\alpha+1}{\alpha-1}} \quad if \quad V_{TH} = 0.$$

V_{TH} はしきい値電圧を示す．αは速度飽和指数を示し，Shockley の長チャネルモデルでは 2.0 であるが，最近の短チャネル MOSFET では約 1.2 である．MOSFET がスケールされればαが低下し，電圧ホッピングの有効性が増すことは好ましいことである．図 3に各制御方式における NP の NW 依存を示す．

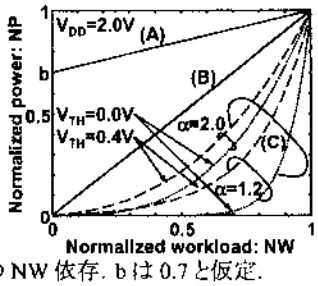

図3. NP の NW 依存．b は 0.7 と仮定.

そこで電圧ホッピングにおいては負荷に応じて f と V_{DD} を柔軟に変化させるアルゴリズムが重要となる．負荷はデータ依存性が強く，ときには実行時間が WCET より大

幅に短いので制御はランタイムに動的に行い，コンパイルタイムに静的に行うべきではない[11]．だがタスクが終了してしまえばもう何もできないので，負荷が WCET よりずっと軽かったとあとで明らかになったところですでに遅い．そうとはいえ将来の負荷を予想することは不可能である．これらの問題解決のためにアプリケーションスライシングとソフトウェア帰還アルゴリズムを導入する．

タスクは N 個のスライスに分割されている．現在の時刻と次のスライスを実行すべき時刻までの余裕を調べ，最適な f を決定する．図 4にアルゴリズムを示す．

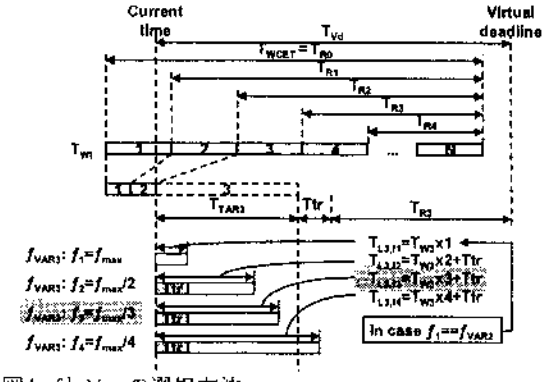

図4. f と V_{DD} の選択方法．

(A) 以下のパラメタはタスクの解析や測定を通して得られる[12]．なお T_{Wi} が T_{Ri-1} - T_{Ri} となるとは限らないので注意を要する．

- T_{WCET}: タスク全体の WCET．
- T_{Wi}: i 番スライスの WCET．
- T_{Ri}: i + 1 から N 番スライスまでの WCET

(B) i 番スライスにおける目標実行終了時間は T_{TARi} = T_{VD} - T_{Ri} - T_{TD} として計算される．T_{TD} は f と V_{DD} を変化させるための遷移時間である．T_{VD} は OS からシステムコールによって得られる仮想締切(virtual deadline)までの時間である．通常 T_{VD} は次のタスクの開始までの時間を意味するが，詳細は次章で述べる．

(C) 候補周波数を $f_j = f_{max} / j$ (j = 1, 2, 3...)とする．j を整数とすることで外部デバイスとのインターフェースにおける同期の問題を回避できる．見積最大実行時間は $T_{Li,fj} = T_{wi} \times j + T_{TD}$ として計算される．もし f_j が i - 1 番スライスにおける周波数と等しければ T_{TD} は必要なく，$T_{Li,fj} = T_{wi} \times j$ となる．

(D) i 番スライスにおける周波数である f_{VARi} として，$T_{Li,fj}$ が T_{TARi} を超えない最小の f_j が選択される．

このように f と V_{DD} はソフトウェアによりスライス単位で制御される．またこのアルゴリズムはタスクの実時間性を保証する．なお f と V_{DD} の関係はプロセサの測定から得られる．

図 5は電圧ホッピングを適用した MPEG4 codec の 1 フレームにおける電力，f，V_{DD} のシミュレーション遷移曲線である．これらは 42%負荷の場合を示しているが，2 段階の f と V_{DD} を持つ場合は電力が 8.6%にまで低減できる．2 段階以上の f と V_{DD} が与えられれば，さらなる低減が可能であるが，その程度は無限段階の場合でも 8%の改善に過ぎない．2 段階電圧ホッピングの場合は f_{max} は全体の 6%しか利用されないが，$f_{max}/2$ は 70%も利用される．残りの時間においてプロセサはスリープする．最悪データが来ることはまれであるが，それに備えて f_{max} はやはり必要である．

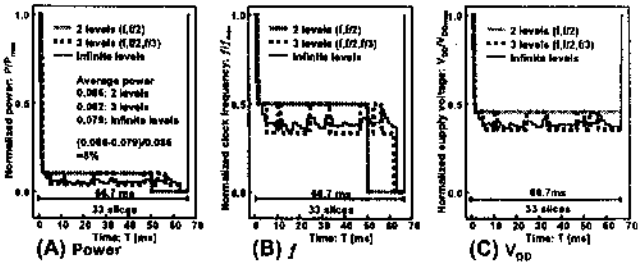

図5. 電圧ホッピングを適用した MPEG4 の(A)電力，(B)f，(C)V_{DD} のシミュレーション遷移曲線．T_{TD} = 66.7ms と設定．

一般的に MPEG4 codec は平均 50%程度の負荷である．他の MPEG2 decode や音声処理の VSELP codec のシミュレーションにおいてもこの傾向は同様であり，いずれも電圧ホッピングを利用した場合，1 桁程度の電力低減を達成できる．

3. CVS (Cooperative Voltage Scaling)

3.1 Power-Conscious OS

プロセサの低電力化に関する OS レベルでの研究は広範囲になされているにもかかわらず，負荷のばらつき，すなわちタスク実行時間の WCET からのばらつきを十分に活かしていない[11], [13]-[16]．また前章で述べた電圧ホッピングによる効果は単一のアプリケーションに制限されている．そこでこの OS レベルへの拡張を考える．具体的には仮想締切を得る機構を有する power-conscious OS と電圧ホッピングを適用したアプリケーションを協調させる．

タスク状態は図 6に示すとおり power-conscious OS によって遷移される．もしタスクが RUN 状態にあるならば，現在それがプロセサを占有している．もしタスクが READY 状態にいるならば，それは実行を待っており，より高い優先度を持った他のタスクが RUN 状態にあることを意味する．READY 状態にあるタスクは READY キュー[1]で優先度順に保持される．もしタスクが DORMANT 状態にあるならば，実行がすでに終了し，次の起動を待っている．DORMANT 状態にあるタスクは DORMANT キューで次の起動が短い順に保持される．RUN タスクが終了した場合にスケジューラは READY キューの先頭のタスクをプロセサへ割付ける．またスケジューラは単位時間毎に DORMANT キューを調べ，タスクを READY キューに起動させるかどうかも判断する．DORMANT キューにあるタスクが READY キューに起動された場合にスケジューラはそのときの RUN タスクと

[1]RUN タスクは READY キューの先頭のタスクであり，プロセサへ割付けされたとはいえ，なお READY キューに留まっていることに注意．

READY キューの先頭のタスクを比較する．それで起動されたばかりのタスクの優先度が RUN タスクより高い場合は RUN タスクの入替えが行われ，入替えられたタスクは再びプロセサへの割付を待つことになる．

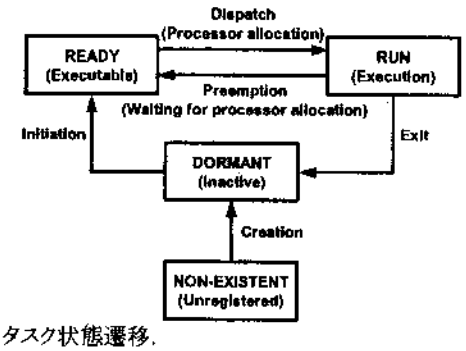

図6. タスク状態遷移．

上記で述べた基本動作以外の power-conscious OS 特有の機能は前述とおり RUN タスクに仮想締切を与えることと RUN すべきタスクがない場合にプロセサをスリープモードにすることである．これらは READY と DORMANT キューの状態に基づいている．もし READY キューに RUN タスク以外のタスクがない場合は RUN タスクの仮想締切は DORMANT キューの先頭にあるタスクの起動時間に設定される．なお RUN タスク以外に READY キューにタスクがある場合は RUN タスクは自身の WCET 以内に実行を終えるようにする．もし全てのタスクが DORMANT キューにあり，RUN すべきタスクがない場合は DORMANT キューの先頭にあるタスクの起動時間までプロセサをスリープさせる．

3.2 CVS

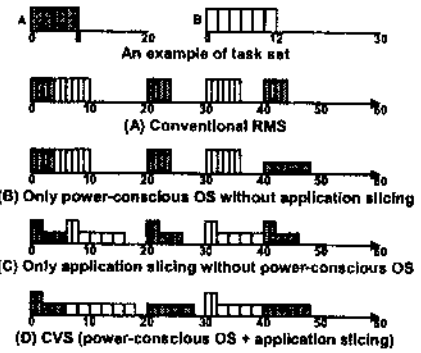

図7. スケジューリングの例．スライスの高さは f を表す．(A)従来の RMS．(B)power-conscious OS のみでアプリケーションスライシングなし．(C)アプリケーションスライシングのみで power-conscious OS なし．(D)CVS．

図 7にスケジューリングの例[1]を示す．タスク集合としてタスク A と B を考える．1 スライスは 2 単位時間を持ち，WCET としてそれぞれ 4 と 6 スライスで構成され，WCET の半分で実行されると仮定する．RMS に基づきは A を高優先度とする．CVS の場合は図 7(D)のようになる．時刻 0 で，タスク B が READY キューにあるためにタスク A は自身の WCET である時刻 8 までに実行を終えなければならない．しかし時刻 6 で，タスク A は終了する．その後タスク B がプロセサに割付けられ，その仮想締切はタスク A の次起動時刻 20 となり，自身の WCET よりも時間余裕が存在する．このような条件下では電力低減の効果がさらに増す．以降も同様に理解できる．

3.3 CVS の実装

μITRON に準拠した日立製作所製 H17750 を改造し，CVS を実装した[17]．具体的には優先度や開始アドレスなどのタスク固有の情報を持ったデータ構造である TCB (task control block)に周期，次起動時間，仮想締切などを追加した．また仮想締切を得たり，f や V_{DD} を変化させたりするためにいくつかのシステムコールを追加した．スケジューラも単位時間毎に次起動時間と仮想締を計算するように改造され，RUN すべきタスクがない場合はプロセサをスリープさせる機構も追加した．なおこれらの実装の詳細はここでは省略する．

4. ハードウェア設計

4.1 ブレッドボード設計

CVS のターゲットとして図 8に示すとおり日立製作所製 SH-4 組込システム基板を選び，電圧ホッピング基板(電圧ホッピングブレッドボード)を接続した．アプリケーションとしては MPEG4 codec と FFT (fast Fourier transform) をマルチタスクとして実行した．SH-4 組込システム基板のブロック図を図 9に示す．MPEG4 の入力生データとして H.263 標準画像の“carphone”をフラッシュ ROM に保存している．FFT は 4096 点を計算する．

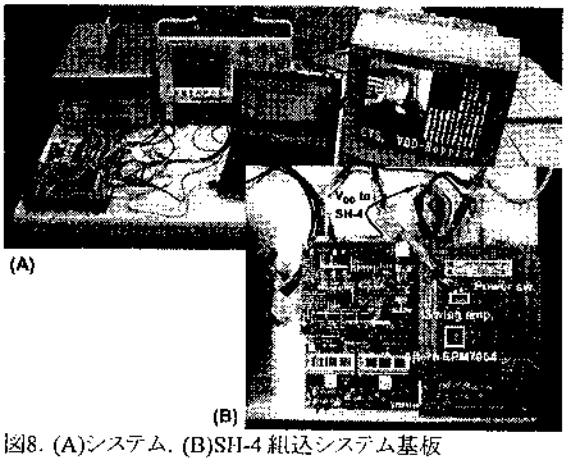

図8. (A)システム．(B)SH-4 組込システム基板

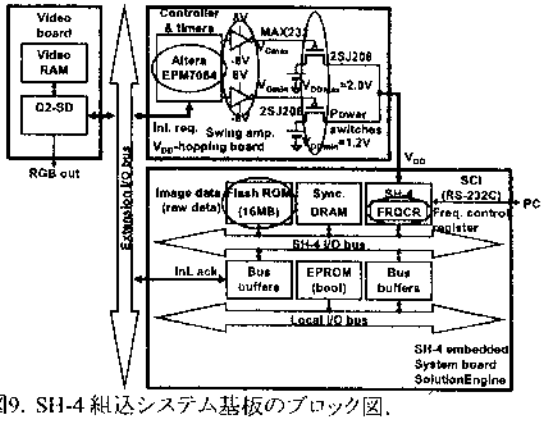

図9. SH-4 組込システム基板のブロック図．

[1] ここでは理解しやすいように時刻は絶対値として表記されているが，実際の power-conscious OS では相対時間が採用されている．

前章のとおり，プロセサによって最適なfとV_{DD}が計算されるが，電圧情報はバスバッファ経由で拡張バスに送られ，FPGA (Altera EPM7064)によって実装された電圧ホッピング基板を制御する．つまり I/O 命令のみで制御可能なため追加命令は必要とならない．このことも電圧ホッピングがプロセサの再設計なしに実装可能であることに貢献している．

この FPGA は内部に 2 つのタイマを持つ．1 つは単位時時間毎に割込みを要求し，時間管理のために使われる．もう 1 つのは V_{DD}を遷移させる際に，遷移時間を設定し，その間はプロセサをスリープさせる．その後設定時刻が来たときに割込みでプロセサを再起動させる．これはプロセサがスリープすると，タイマによって再起動させない限り，何もプロセサを再起動させることができなくなるからである．プロセサ固有の特別な機能が利用されていないので電圧ホッピングはどんなプロセサにも適応できる．

次節に電圧ホッピングの実装について注意すべき点をいくつか挙げる．

4.2 周波数

図 9のとおり，SH-4 は FRQCR と呼ばれる周波数制御レジスタを内蔵しており，即時に内部周波数を変更できる．この内部周波数は外部周波数 33MHz と同期している．このシステムでは 33MHz の整数倍である 200MHz と 100MHz を内部周波数fとして使っており，外部システムとのインターフェースにおける同期の問題は発生しない．しかしながら一般にはこのような周波数制御レジスタは実装されていないこともあるので，そのときは電圧ホッピング基板から 2 段階以上の周波数を与えなければならない．次章に述べる ASIC はそれ自身でこれらの周波数を出力することができる．

4.3 スイッチ

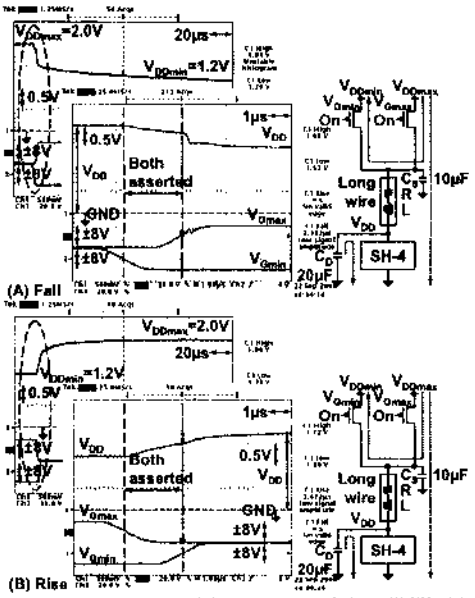

図10. V_{Gmax}とV_{Gmin}の両方がアサートされる期間がある場合のV_{DD}の波形．

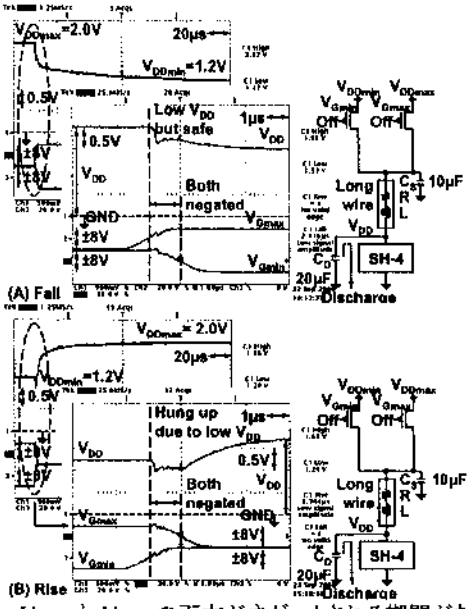

図11. V_{Gmax}とV_{Gmin}の両方がネゲートされる期間がある場合のV_{DD}の波形．

図 9に示すとおり電圧ホッピング基板上で V_{DD} は f = 200MHz のとき $V_{DD} = V_{DDmax}$, f = 100MHz のとき $V_{DD} = V_{DDmin}$となるように 2SJ208 (スイッチ MOSFET)で変更される．このスイッチは市販品で最も低いしきい値を持つが，それでも 2.8V もあり，プロセサの定格 V_{DD} である 2.0V より高い．そのためスイッチはこのままでは決してオンしない．そのためゲート信号を±8V まで増幅する MAX232 (RS-232C ドライバ)を信号振幅増幅器として使用する．なおプロセサの測定により V_{DDmax}=2.0V，V_{DDmin}=1.2V が最適値として得られている．

図 10と図 11は V_{DD}の測定波形である．V_{DD}の立上がりと立下がり時間はデカップリング容量が 30μF 場合，それぞれ 200μs と 100μs 以下である．

ここでスイッチのゲートのタイミングについて述べる．V_{DDmax}を有効にする信号 V_{Gmax} と V_{DDmin} を有効にする信号 V_{Gmin} のタイミングに注意が必要である．タイミングには 2 つの場合がある．1 つは信号の重なり時間がある場合，もう 1 つは重なりがない場合である．1 つの MOSFET をオンすると同時にもう 1 つをオフすることは事実上不可能である．図 10のとおり重なりがある場合は大電流が V_{DDmax} から V_{DDmin} へ流れ，問題となるかもしれない．しかしながら信号の重なりを 2μs に設定すればデカップリング容量 C_D のおかげでスパイクノイズや急速な電圧変動は観察されない．一方図 11に示すとおり重なりがなければ，V_{DDmax} と V_{DDmin} の両方とも完全にオフしている期間があり，このとき本当に問題となる．V_{DD} が V_{DDmin} 以下に低下し，システムがハングアップする．結論としてスイッチのタイミングは V_{DDmax} と V_{DDmin} の両方が V_{DD} に短い時間つながるように実行されるべきである．

他に重要な点としてプロセサリセットにおいては V_{Gmax} を有効としなければならない．なぜならばプロセサの安定したブートアップのために最も高い V_{DD} である V_{DDmax} が供給されなければならないからである．

4.4 電力

図 12にシステムで測定された電力特性を示す. プロセサの消費電力は f = 200MHz において 0.8W, f = 100MHz において 0.16W, スリープにおいて 0.07W である. f = 200MHz を利用する時間は 10.5%, f = 100MHz は 54.0%, スリープは 35.5%であったので平均電力は 0.20W (10.5% x 0.8+54.0% x 0.16+35.5% x 0.07)である. なお SH-4 の I/O バッファは定格以下の電源電圧で最適化されていない. もし注意深く設計されていたなら V_{DDmin}は 1.2V より低い 0.9V でよいはずである. その場合は 100MHz 時の消費電力はさらに半分程度まで低減できる.

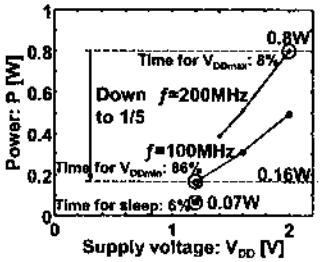

図12. CVS システムの測定消費電力特性.

5. LSI 設計

電圧ホッピングブレッドボードと同様の機能を持つ LSI を設計, 試作した. 具体的には電圧ホッピング基板の FPGA 部分をスタンダートセルベースで設計し, スイッチを追加した.

まず LSI 設計においてスイッチのゲート幅が重要になる. 図 13にシミュレーション V_{DS} 曲線を示す. なおプロセスしきい値は 0.6V であり, V_{DDmin} = 1.2V より小さく, 電圧ホッピング基板では必要となった信号振幅増幅器は必要ない. ゲートバイアス V_{GS} = V_{DDmin}で負荷電流 0.13A の時, 最も大きなゲート幅を必要とする. このときスイッチによる電圧降下が 0.05V 以下になるように 270,000μm のゲート幅を選択する. この大きさでもちろん V_{GS} = V_{DDmax}の場合の負荷電流 0.4A を流すこともできる.

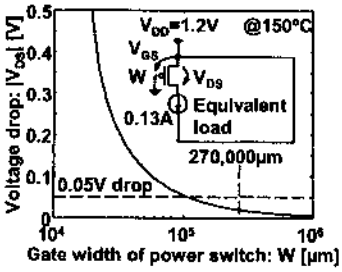

図13. スイッチによる電圧降下のゲート幅依存. これは最低ゲートバイアス V_{GS}=1.2V なので最悪の場合を示している.

図 14に電圧ホッピング LSI の回路図を示す. スイッチの重なり時間を調整するためにプログラマブルタイマがスイッチのゲートを駆動する. またスイッチ用汎用デコーダと, 周波数制御レジスタを実装していないプロセサのために f_{max}と $f_{max}/2$ のどちらかを任意に選択できる周波数セレクタも備える. あとここでは示されていないが, 現在時刻の監視と電圧遷移のための割込み付タイマも持っている. なお一般的な注意として, fと V_{DD}が変更されている間は誤動作を防ぐためにプロセサはスリープさせるべきである.

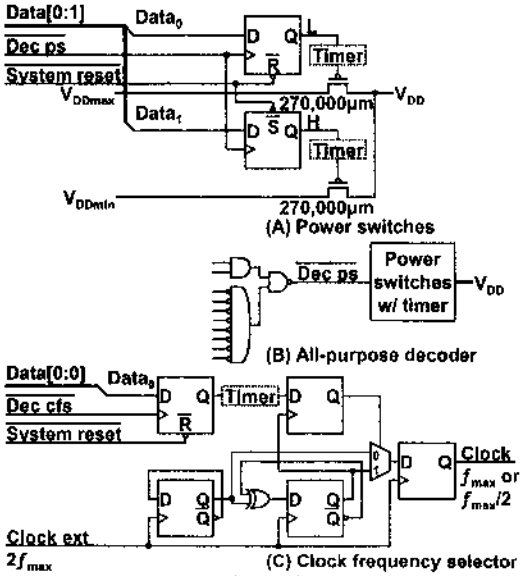

図14. (A)スイッチ, プログラマブルタイマが重なり時間を調整する. (B)汎用デコーダ, (C)周波数セレクタ. プログラマブルタイマがプログラム実行中の周波数の変更を妨げる.

なお電圧ホッピング LSI は 0.6μm, 3 層メタル CMOS プロセスで試作され, 消費電力は 0.01W である. にチップ写真を示す. 2 つのスイッチを含め, 大きさは約 4.6mm x 2.3mm である.

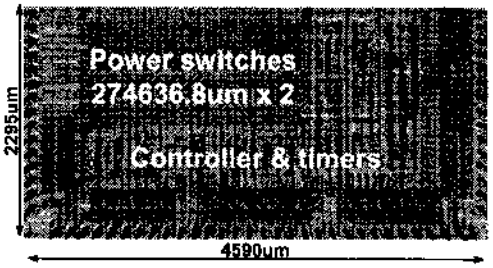

図15. 電圧ホッピング LSI.

6. 結果

図 16はプロセサの V_{DD}とスリープ信号の測定波形を示す. 図 5のシミュレーション波形と似ていることが理解できる. 各電源電圧の時間的割合はホッピング基板同様, V_{DDmax}は時間の 10.5%にのみ用いられている. そのため平均負荷は 37.5% (10.5% x 1+54% x 0.5+35.5% x 0) である.

CVS と従来の RMS との消費電力比較を図 17に示す. RMS ではタスクによってプロセサが占有されていないときは NOP を実行する. CVS は 0.20W の電力を消費すると測定され, 図 16の場合は RMS の 1/4 以下の電力にまで低減できる. もし I/O バッファを低電圧において最適化し, V_{DDmin}を 0.9V まで低下させることができる場合はさらに 0.16W にまで低減できる.

実は CVS の電力低減はタスク周期の組み合わせに依存し, つまりは仮想締切をどれだけ利用できるかで効果

が異なる．さらにはタスクの実行時間のばらつきにも依存している．しかしながらマルチメディアアプリケーションが持つ実効時間のばらつきを最大限に利用しており，CVS はこのようなアプリケーションに対して十分に電力低減を期待できる．

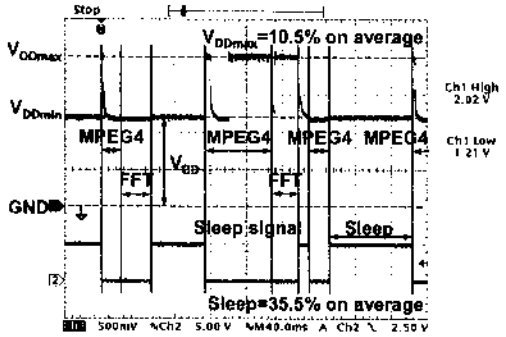

図16．プロセサの V_{DD} とスリープ信号の測定波形．MPEG4 と FFT は OS 上でマルチタスクとして実行されている．MPEG4 の周期は 114ms でその WCET は 79ms である．FFT のそれらはおのおの 171ms と 35ms である．MPEG4 と FFT は 22 と 2 スライスに分割されている．V_{DD} の波形は前述の CVS の例に似ている．

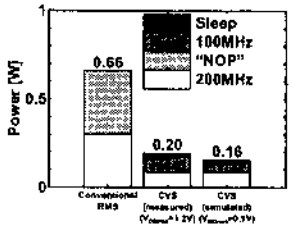

図17．CVS と RMS の電力比較．

7. 結論

この論文では CVS のための OS，アプリケーションスライシング，電圧ホッピングハードウェアについて述べた．CVS の本質はタスク間の時間余裕とタスク内部における実行時間の WCET からのばらつきを利用することである．測定結果から CVS は MPEG4 と FFT のマルチタスク環境において電力を従来の 1/4 以下にまで低減できることを示した．

謝辞

日立製作所中央研究所の佐々木勝郎氏，石橋孝一郎氏，日立米沢電子の八巻一志氏に感謝いたします．

この研究は日立製作所と日本学術振興会から受託されたものである．LSI 試作は東京大学大規模集積システム設計教育研究センターをとおしロームと凸版印刷の協力で行われたものである．

参考文献

[1] A. Chandrakasan, V. Gutnik, and T. Xanthopoulos, "Data Driven Signal Processing: An Approach for Energy Efficient Computing," Proceedings of International Symposium on Low Power Electronics and Design, pp. 347-352, 1996.

[2] T. Kuroda, K. Suzuki, S. Mita, T. Fujita, F. Yamane, F. Sano, A. Chiba, Y. Watanabe, K. Matsuda, T. Maeda, T. Sakurai, and T. Furuyama, "Variable Supply-Voltage Scheme for Low-Power High-Speed CMOS Digital Design," IEEE Journal of Solid-State Circuits, vol. 33, no. 3, pp. 454-462, 1998.

[3] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. B. Srivastava, "Power Optimization of Variable Voltage Core-Based Systems," Proceedings of Proceedings of Design Automation Conference, pp. 176-181, 1998.

[4] T. Pering, T. Burd, and R. Brodersen, "The Simulation and Evaluation of Dynamic Voltage Scaling Algorithms," Proceedings of International Symposium on Low Power Electronics and Design, pp. 76-81, 1998.

[5] T. Ishihara, and H. Yasuura, "Voltage Scheduling Problem for Dynamically Variable Voltage Processors," Proceedings of International Symposium on Low Power Electronics and Design, pp. 197-202, 1998.

[6] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A Dynamic Voltage Scaled Microprocessor System," IEEE International Solid-State Circuits Conference Digest of Technical Papers, pp. 294-295, 2000.

[7] C. L. Liu, and James W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real Time Environment," Journal of the ACM, vol. 20, no. 1, pp. 46-61, 1973.

[8] S. Lee, and T. Sakurai, "Run-time Power Control Scheme Using Software Feedback Loop for Low-Power Real-time Applications," Proceedings of Asia and South Pacific Design Automation Conference, pp. 381-386, 2000.

[9] S. Lee, and T. Sakurai, "Run-time Voltage Hopping for Low-power Real-time Systems," Proceedings of Design Automation Conference, pp. 806-809, June 2000.

[10] T. Sakurai, and A. R. Newton, "Alpha-Power Law MOSFET Model and its Applications to CMOS Inverter Delay and Other Formulas," IEEE Journal of Solid-State Circuits, pp. 584-594, vol. 25, no. 2, 1990.

[11] Y. Shin, and K. Choi, "Power Conscious Fixed Priority Scheduling for Hard Real-Time Systems," Proceedings of Design Automation Conference, pp. 134-139, 1999.

[12] S. Lim, Y. Bae, G. Jang, B. Rhee, S. Min, C. Park, H. Shin, K. Park, and C. Kim, "An Accurate Worst Case Timing Analysis for RISC Processors," Proceedings of IEEE Real-Time Systems Symposium, pp. 97-108, 1994.

[13] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for Reduced CPU Energy," Proceedings of USENIX Symposium on Operating Systems Design and Implementation, pp. 13-23, 1994.

[14] F. Yao, A. Demers, and S. Shenker, "A Scheduling Model for Reduced CPU Energy," Proceedings of IEEE Annual Foundations of Computer Science, pp. 374-382, 1995.

[15] C. Hwang and A. Wu, "A Predictive System Shutdown Method for Energy Saving of Event-Driven Computation," Proceedings of IEEE/ACM International Conference on Computer Aided Design, pp. 28-32, 1997.

[16] Y. Lee and C. Krishna, "Voltage-Clock Scaling for Low Energy Consumption in Real-time Embedded Systems," Proceedings of International Workshop on Real-Time Computing Systems and Applications, 1999.

[17] Hitachi Semiconductor and Integrated Circuits Web Site: http://www.hitachi.co.jp/Sicd/English/Products/micome.htm.

[18] Y. Shin, H. Kawaguchi, and T. Sakurai, "Cooperative Voltage Scaling (CVS) between OS and Applications for Low-Power Real-Time Systems," Proceedings of IEEE Custom Integrated Circuits Conference, May 2001 (in press).

[19] H. Kawaguchi, G. Zhang, S. Lee, and T. Sakurai, "An LSI for V_{DD}-Hopping and MPEG4 System Based on the Chip," Proceedings of IEEE International Symposium on Circuits and Systems May 2001 (in press).