情報処理学会短期集中セミナー「組込みシステム技術の展望」

組込みハードウェア技術の現状と展望

冨山宏之
名古屋大学　大学院情報科学研究科
http://www.ertl.jp/~tomiyama/

機械～電子～ソフトウェア制御

- 炊飯器
  - 釜
  - 1955年：電気炊飯器
  - 1980年代：マイコン制御炊飯器
- カメラ
  - 感光フィルム
  - 1988年：デジタルカメラ
- テレビ
  - アナログ映像＋機械制御
  - アナログ映像＋電子制御
  - デジタル映像＋電子制御（ソフトウェア）
- ビデオ
  - 1970～80年代：VHS 対 ベータ
  - 現在：Blu-ray Disc 対 HD DVD

組込みハードウェアの歴史

- 1980年～
  - 4～16ビットマイコン
  - 家電製品（炊飯器、電子レンジ、ビデオ等）の制御
  - 要求：コード効率
- 1990年～
  - 16～32ビットRISC型組込みプロセッサ　＋　ASIC/DSP
  - 機器の制御＋データ処理
    - 情報機器（FAX、コピー機等）
    - 携帯機器（PHS、携帯電話、PDA等）
  - 要求：コード効率＋高性能＋低消費電力
  - キーテクノロジ
    - 1990年代前半
      - ハードウェア記述言語＋自動論理合成
    - 1990年代後半
      - ハードウェア/ソフトウェア・コデザイン
      - 低消費電力設計技術

組込みハードウェアの歴史

- 2000年～現在
  - SoC（Systems-on-Chip）～MPSoC（Multiprocessor SoC）
    - システムLSIは和製英語
    - 1999年：設計自動化（DA）研究会が、システムLSI設計技術（SLDM）研究会へと名称変更
  - 機器の制御＋マルチメディア処理
    - デジタル家電（DVDプレーヤ/デコーダ、デジタルTV、第3世代携帯等）
  - キーテクノロジ
    - システムレベル設計、プラットフォームベース設計
      - ハードウェア／ソフトウェアの統合的ソリューション
    - マルチプロセッサによる並列処理
    - 低消費電力設計技術

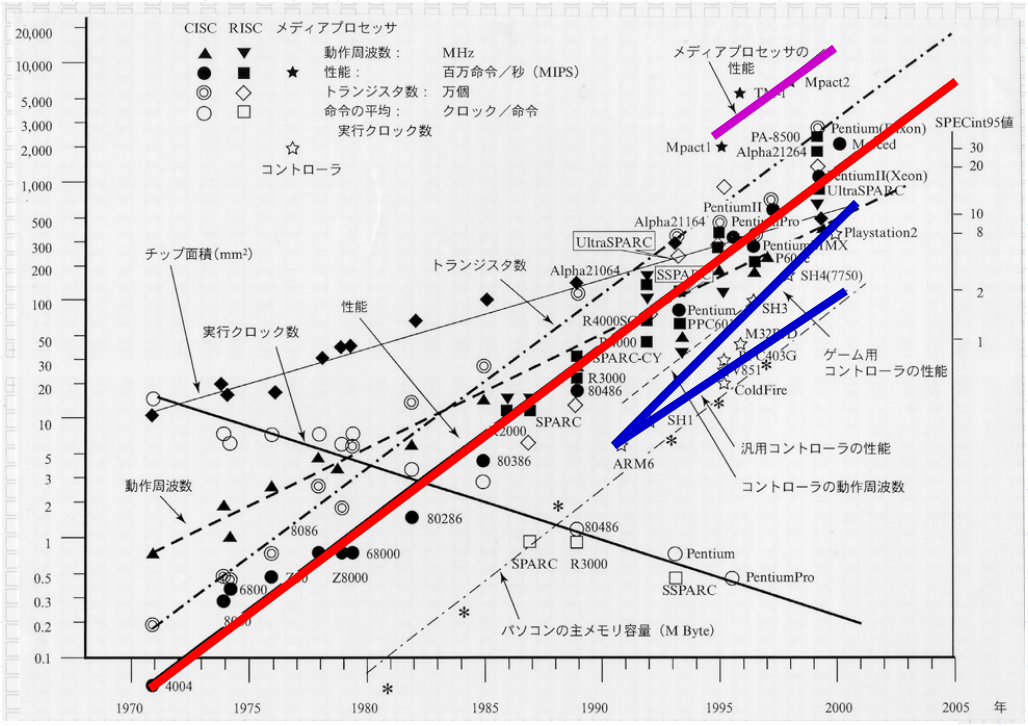
汎用プロセッサと組込みプロセッサ

- サーバ機向け
  - Dual-Core Itanium 2 Processor 9000
  - 電源電圧：1.08～1.25V
  - 周波数：1.6GHz
  - 消費電力（TDP）：104W
- デスクトップPC向け
  - Core 2 Duo Desktop Processor
  - 電源電圧：0.85～1.6V
  - 周波数：1.86～2.66GHz
  - 消費電力（TDP）：65W
- ノートPC向け
  - Core 2 Duo Mobile Processor
  - 電源電圧：0.75～1.3V
  - 周波数：1～2.33GHz
  - 消費電力（TDP）：20～34W
- 組込みシステム向け
  - XScale PXA 270 Processor
  - 電源電圧：0.85～1.55V
  - 周波数（可変）：13～624MHz
  - 消費電力：44～925mW

その他の組込みプロセッサ/DSP

- ルネサス SH-4A SH7780
  - 電源電圧：　1.0～1.25V
  - 周波数：　200～400MHz
  - 性能：　360～720MIPS
  - 消費電力：　80～250mW
- TI　C64x Fixed-Point DSPs
  - 電源電圧：　1.1～1.4V
  - 周波数：　300MHz～1GHz
  - 性能：　2400～8000MIPS
  - 消費電力：　260mW～1.06W

汎用プロセッサと組込みプロセッサ



出典：嶋正利、「マイクロプロセッサの25年」、電子情報通信学会誌、82巻10号、1999年

本講演の内容

1. はじめに
2. マルチプロセッサ技術
3. 低消費電力設計技術
4. 今後の展望

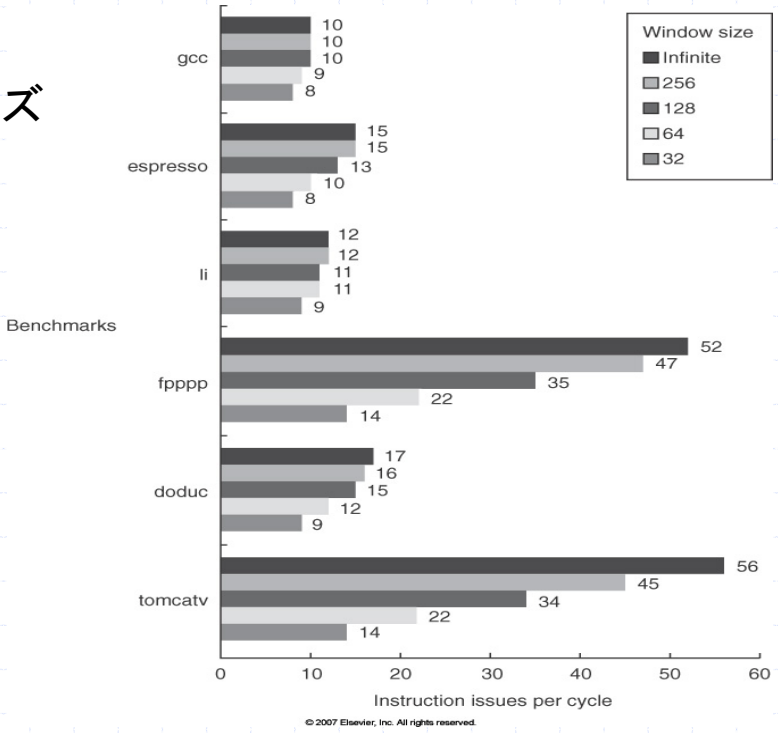
マルチプロセッサ技術

何故マルチプロセッサか？

- 汎用プロセッサ
  - 性能の向上
    - ILP（Instruction Level Parallelism）の限界
      - 命令間のデータ／制御依存関係
      - Memory Wall
  - 消費電力の抑制
    - 高性能プロセッサの消費電力は100W以上
    - 冷却装置や電源装置のコストが大
  - 信頼性（耐故障性）の向上
- 組込みプロセッサ
  - 性能向上と消費電力削減の両立
    - 低性能CPU複数個の消費電力 ＜ 高性能CPU 1個の消費電力
  - コスト（価格）
    - 低性能CPU複数個の価格 ＜ 高性能CPU 1個の価格

ILP限界の主な要因

- 最大命令発行数
- 命令ウィンドウのサイズ
- リネームレジスタ数
- 分岐予測
- キャッシュ
- メモリエイリアス



Source: J. L.. Hennessy and D. A. Patterson, Computer Architecture, Elsevier, 2006.

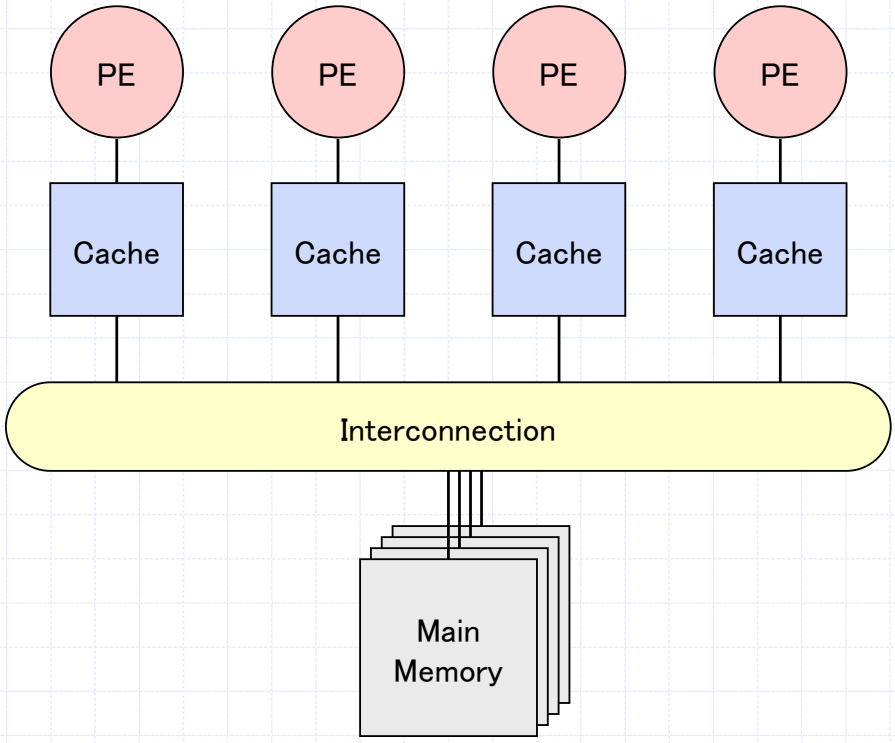
マルチプロセッサの種類に関する用語

- マルチプロセッサ
- マルチコア・プロセッサ
- マルチコンピュータ
- CMP（Chip Multiprocessors）
- SMP、AMP/ASMP、FDMP
- MPSoC（Multiprocessor Systems-on-Chip）
- MCSoC（Multi-Core Systems-on-Chip）
- NoC（Networks-on-Chip）
- UMA、NUMA、CC-NUMA、COMA、NORA/NORMA
- 共有メモリ、集中共有メモリ、分散共有メモリ、メッセージパッシング
- 密結合、疎結合
- ホモジニアス、ヘテロジニアス
- タスク並列、データ並列
- スレッドレベル並列（TLP）、データレベル並列（DLP）

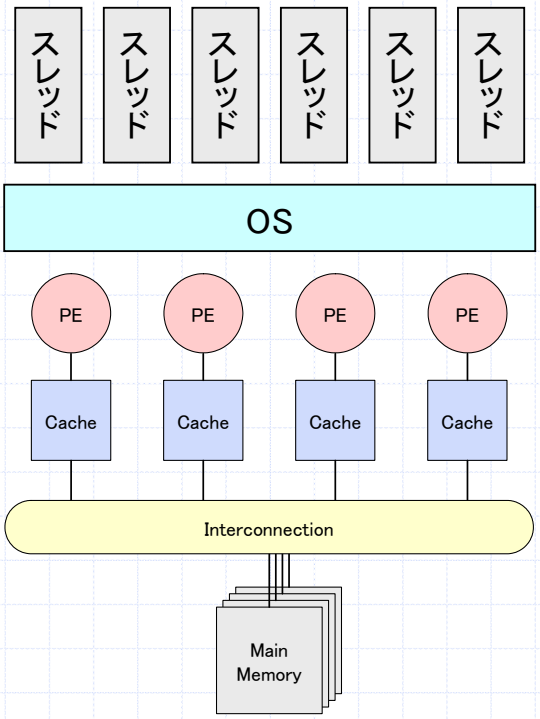
マルチプロセッサの分類

- プロセッサの種類
  - 均質(Homogeneous)
  - 非均質(Heterogeneous)
- メモリの構成
  - 共有メモリ(Shared Memory)
    - 対称共有メモリ(Symmetric Shared-Memory)
      - 単に、対称型マルチプロセッサ(SMP)とも呼ばれる
      - 集中共有メモリ(Centralized Shared-Memory)と同義
      - UMA(Uniform Memory Access)とも同義
    - 分散共有メモリ(Distributed Shared-Memory)
      - NUMA(Non-Uniform Memory Access)と同義
  - メッセージ交換(Message Passing)
    - NORA/NORMA(No Remote Memory Access)と同義
    - 厳密には、マルチプロセッサではなく、マルチコンピュータ
- 相互接続
  - バス、リング、完全網、クロスバ、メッシュ、トーラス、ハイパーキューブ、Fat Tree、スター、オメガ、その他

Symmetric Shared-Memory (UMA)

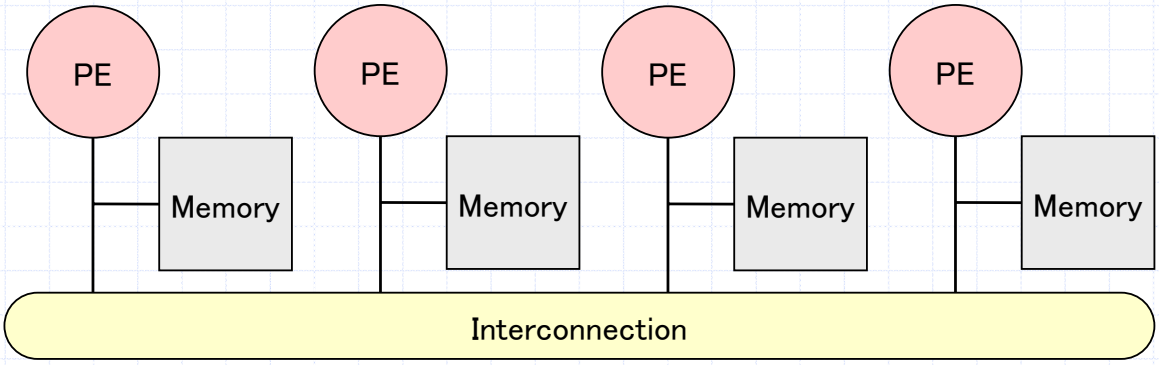


Symmetric Shared-Memory (UMA)



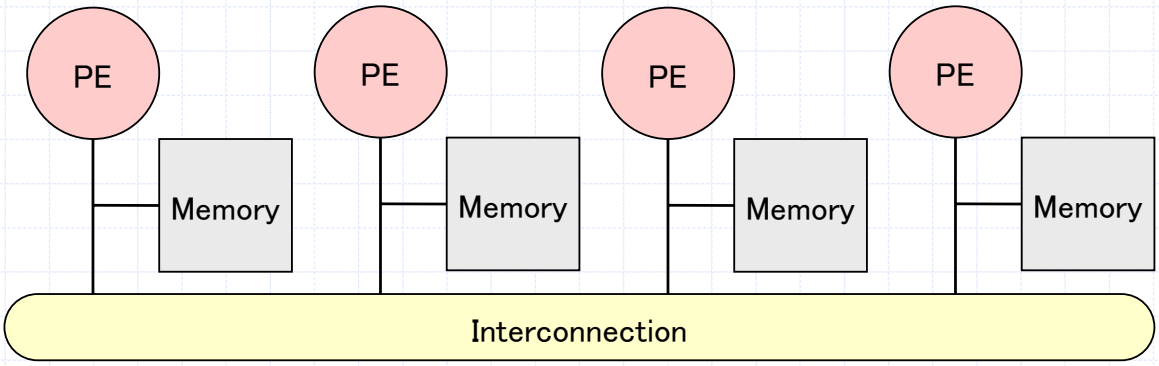
- 単一の主記憶
  - 主記憶が複数のバンクに分かれていることもある
  - 主記憶以外に、各PEがローカルメモリを持つこともある
- メモリアクセス・レイテンシが全PEで同一
- 多くの場合、コヒーレントキャッシュを持つ
- 多くの場合、一つのOSが、動的にスレッド(タスク)の割当てを行う
  - 動的な負荷分散

Distributed Shared-Memory (NUMA)



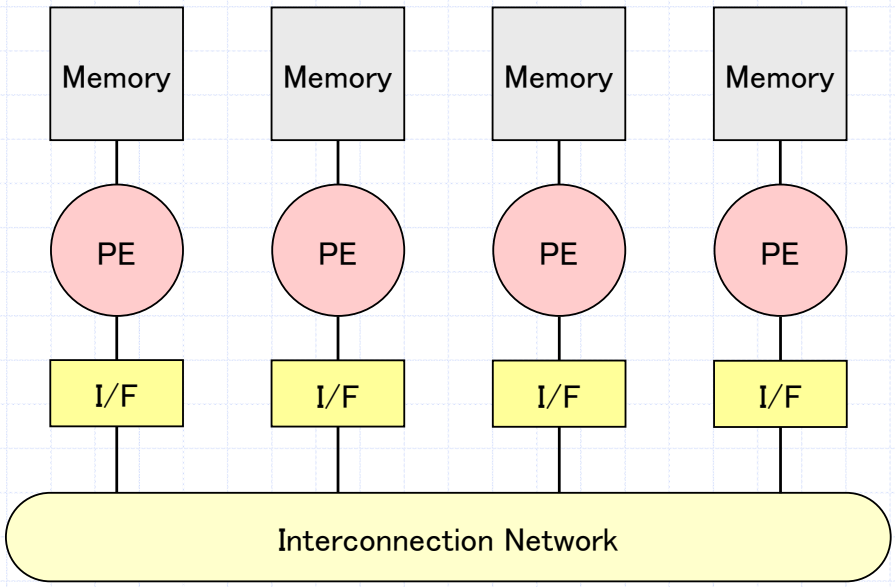
- 各PEがメモリを持つ
- 自分以外のPEのメモリにアクセスすることはできるが、レイテンシが長い
- 多くの場合、
  - 各PE上でOSが動作し、
  - 動的な負荷分散はアプリケーション(またはミドルウェア)レベルで行われる、あるいは、
  - 動的な負荷分散は行われない(各PE上で実行されるスレッド／タスクは固定)

Distributed Shared-Memory (NUMA)

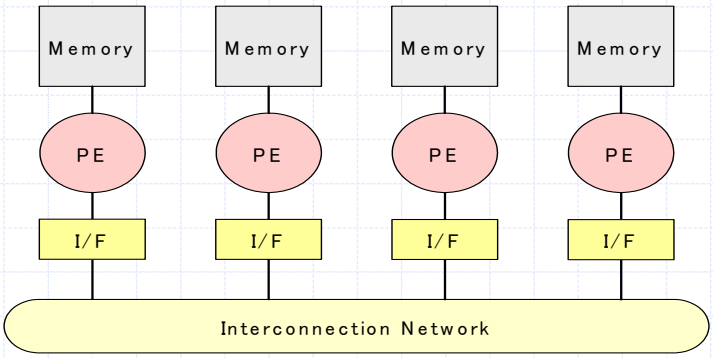


- 多くの場合、各PEはメモリだけでなくキャッシュも持つ
  - （単純な）NUMA
    - コヒーレントキャッシュを持たない
  - CC-NUMA（Cache Coherent NUMA）
    - コヒーレントキャッシュを持たない
  - COMA（Cache Only Memory Access）
    - キャッシュしか持たない（主記憶を持たない）

Message Passing (NORA)



NORA (Message Passing)



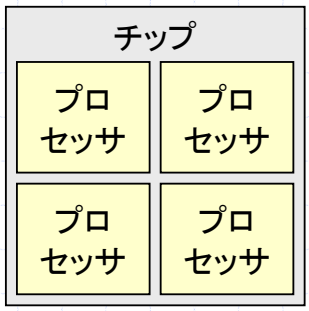
- 各PEはプライベートなメモリを持ち、共有メモリは持たない
- 自分以外のPEのメモリにアクセスすることはできない
- メッセージ（パケット）を送受信することにより、PE間で通信を行う

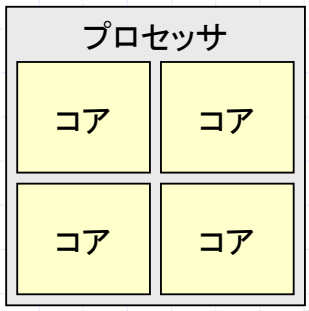
用語の補足説明

- 密結合
  - Tightly Coupled
  - 一般的に、UMA型（SMP）を指す
- 疎結合
  - Loosely Coupled
  - 一般的に、メッセージ交換型（NORA）を指す
  - 粗結合は誤変換
    - 粗粒度（Coarse Grained）と混同しないように
- 最近は密結合／疎結合と呼ぶことは稀

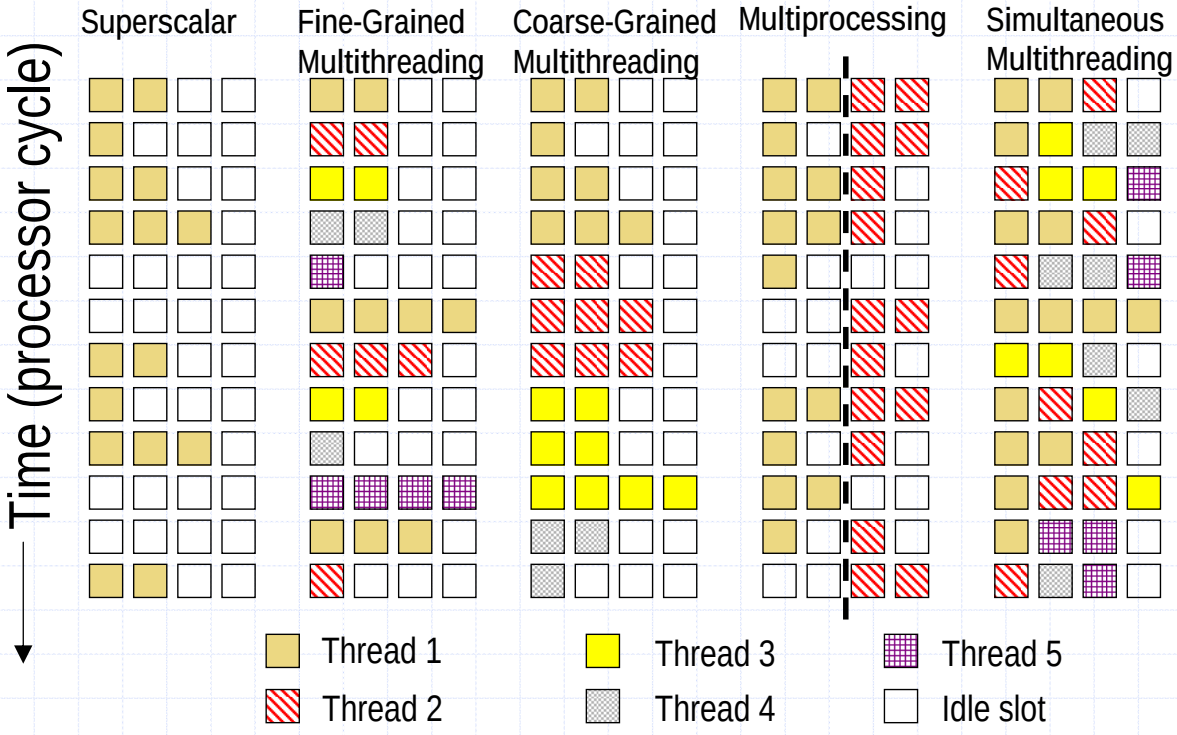
用語の補足説明

- CMP（Chip Multiprocessors）
  - 1チップ上に集積されたマルチプロセッサシステム
  - アーキテクチャ屋が好んで使用する
    - 1チップ上のUMA（SMP）型を指すことが多い
- マルチコア・プロセッサ
  - CMPと同義
  - 単に、呼び方の違い





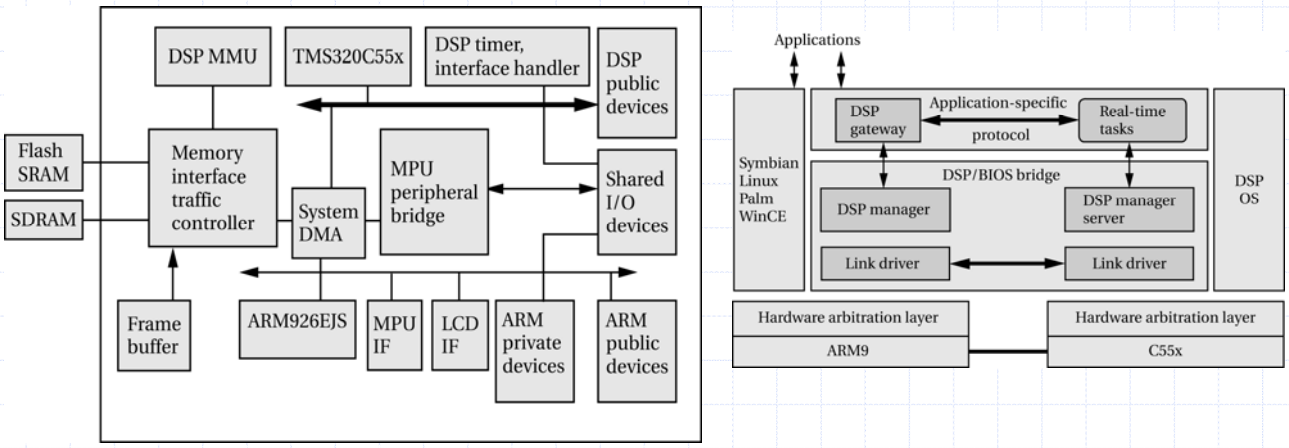
マルチスレディングとマルチプロセッサ



Source: P. A. Patterson, Computer Architecture, Elsevier, 2006.

TI OMAP

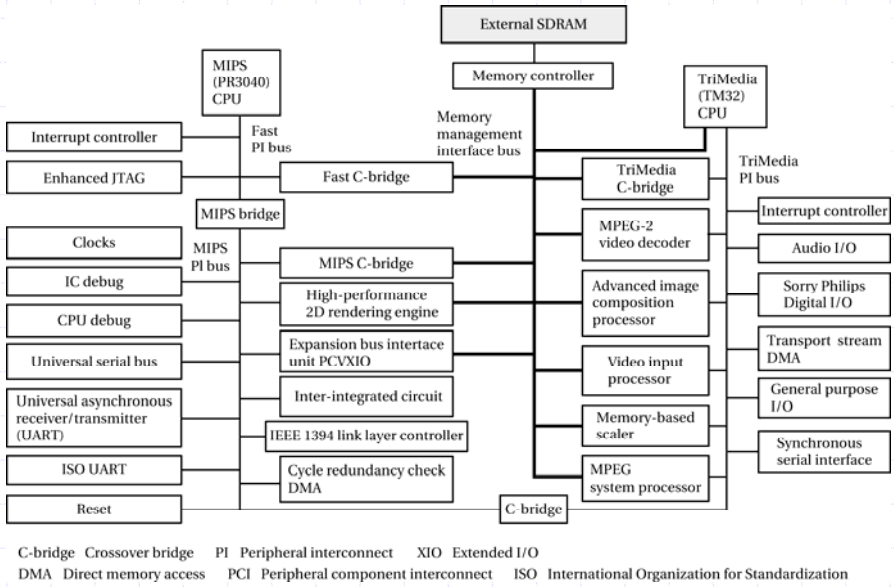
- 携帯機器向け
- ヘテロジニアスなデュアルプロセッサ
- ARM9　＋　TI C55x



Source: W. Wolf, High-Performance Embedded Computing, Elsevier, 2006.

NXP Viper Nexperia

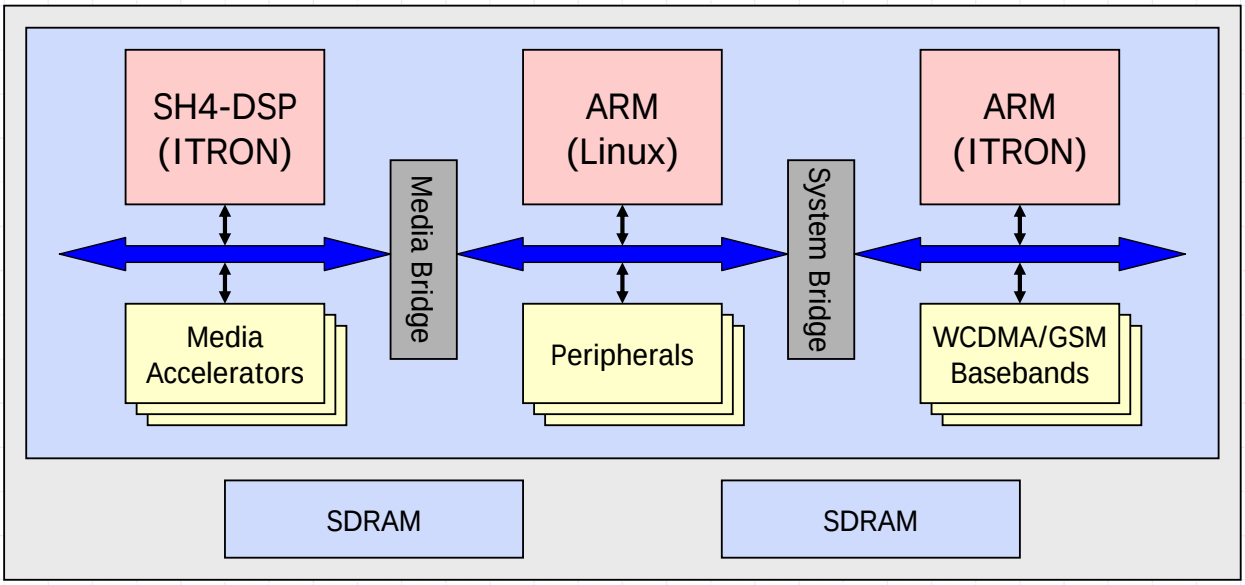
- HDTV（1920x1080）向けSOC
- MIPS＋TriMedia（128ビットSIMD、5命令発行VLIW）



Source: W. Wolf, High-Performance Embedded Computing, Elsevier, 2006.

Renesas SH-Mobile G1

- 携帯電話向け
- ヘテロジニアスAMP



組込み用マルチプロセッサの現状

- SMP（UMA）方式は稀
  - 他方式と比較して、ソフトウェア開発は楽
  - しかし、リアルタイム性の保証が困難
    - 共有資源が多い　→　衝突が生じやすい
  - コストパフォーマンスも良くない
    - ヘテロジニアスの方が、アプリケーションの特徴を活用できる
- 現状では、以下の方式が多い
  - ヘテロジニアスなデュアル・プロセッサ
    - 1つのアプリケーション・プロセッサ
    - 1つのメディア・プロセッサ（DSP/VLIW/SIMD）
  - ヘテロジニアスなNUMA型マルチプロセッサ
    - プロセッサ:　　ヘテロジニアス
    - メモリ:　　ヘテロジニアス
    - 相互接続:　　ヘテロジニアス
  - ホモジニアスとヘテロジニアスの折衷方式
    - 1つのアプリケーション・プロセッサ
    - ホモジニアスな複数のメディア・プロセッサ（DSP/VLSI/SIMD）

低消費電力設計技術

背景

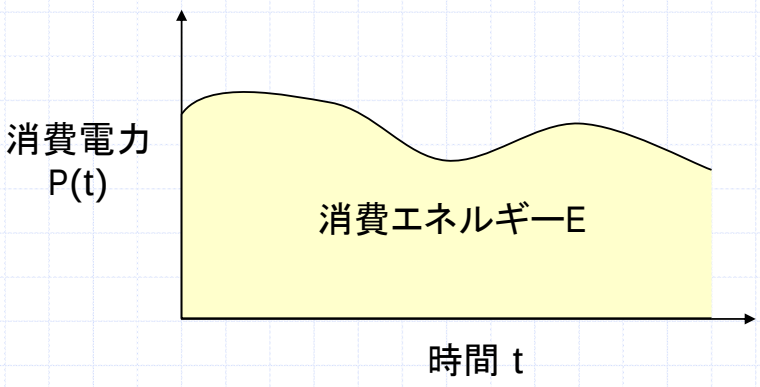
- 携帯型組込みシステムの普及
  - 携帯電話、デジタルカメラ、ビデオカメラ、音楽プレーヤ、ゲーム、PDA、他
  - バッテリ寿命は商品価値を大きく左右する
- 情報家電の普及
  - デジタルTV、HDレコーダ・プレーヤ、ゲーム機、プリンタ／複合機、他
  - 使用時だけでなく、待機時も電力を消費
- ICタグ、センサネットワークの普及

低消費電力化の効果

- 製造コストの低減
  - パッケージ、ヒートパイプ、ヒートシンク、ファン、電源
- 運用コストの低減
  - 電気料金、電池代
- 信頼性の確保
  - 温度が10℃上昇すると故障率がほぼ倍増
- 環境の保全
  - 生活：停電（家庭～建物～地域一帯）
  - 環境：地球温暖化
    - 京都議定書
      - 先進国等に対し、温室効果ガスを1990年比で、2008年～2012年に一定数値（日本6%、米7%、EU8%）を削減することを義務づけ
  - 資源：原油、天然ガス
  - 安全：原子力発電所
    - 日本人1人当たり10ワットの電力を節約すると、日本全体で130万キロワットの節約
      - 美浜原発1号機×4機

消費電力と消費エネルギーの違い

- 消費電力
  - 仕事率（瞬間発熱量）
  - Watt [W]、VA
  - P=V・I
  - パッケージ、冷却装置、電源、信頼性に影響
- 消費エネルギー
  - 仕事量
  - Joule [J]
  - 消費電力の時間積分
    - $E = \int P(t)\,dt \fallingdotseq Pave \cdot T$
  - バッテリ寿命に影響



消費電力

- 動的な消費電力
  - ゲートの出力が0から1に、あるいは、1から0に変化するときに消費する電力
  - 動的な消費電力の例
    - スイッチングによる消費電力（80%程度）
    - 貫通電流による消費電力（微小）
- 静的な消費電力
  - 電源が入っているときに定常的に消費する電力
  - 静的な消費電力の例
    - リーク電流による消費電力（20%程度）

リーク電流による消費電力

- 回路がスイッチングしていなくても、定常的に漏れて流れる電流（リーク電流）により消費される電力
- リーク電流の種類
  - サブスレッショルドリーク ⇐ 支配的
  - ゲートトンネリングリーク
  - ジャンクションリーク
  - GIDL（Gate-Induced Drain Leakage）
- オフのときでも、電流が僅かに漏れ流れてしまう
  - 現実のトランジスタ（MOS FET）は完全なスイッチではない
- 微細化に伴い、問題が顕在化

消費電力／エネルギーの近似式

$$P = C \cdot A \cdot f \cdot Vdd^2 + Vdd \cdot Ileak$$

$$Ileak \propto \exp\left(\frac{-Vth}{\alpha \cdot Tk}\right)$$

$$E = P \cdot T = C \cdot A \cdot Vdd^2 \cdot Cycles + Ileak \cdot Vdd \cdot T$$

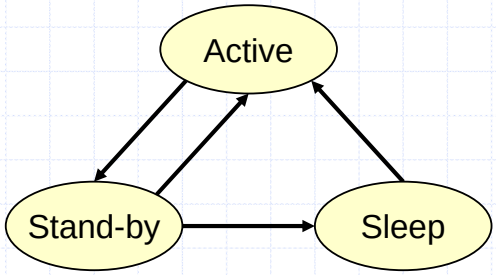
- C　負荷容量
- A　活性化率（スイッチング確率）
- f　クロック周波数
- T　実行時間（=Cycles/f）
- Cycles　実行サイクル数
- Vdd　電源電圧
- Vth　閾値電圧
- Tk　絶対温度

低消費電力／エネルギー化技術

- 現在のハードウェアは低消費電力化のための様々な機構を有している
  - これらの機構を有効活用することがソフトウェア技術者の任務
- 代表的な低消費電力化技術
  - 動的電力管理（DPM: Dynamic Power Management）
  - 動的電圧制御（DVS: Dynamic Voltage Scaling）
  - スクラッチパッドメモリ（SPM: Scratch-Pad Memory）

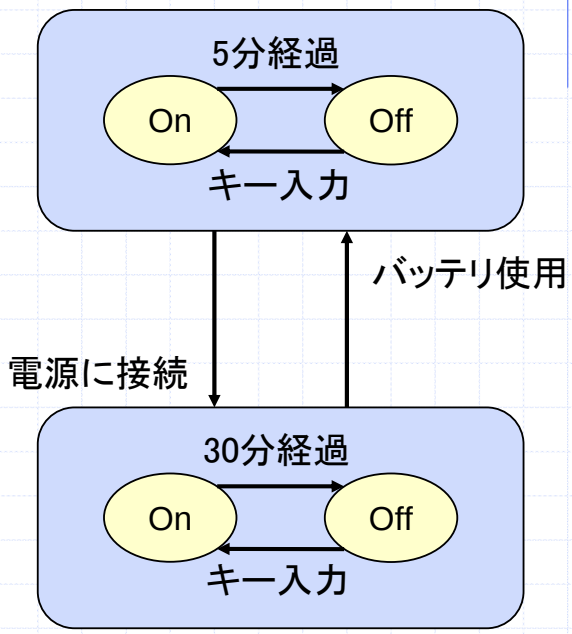
動的電力管理

- DPM（Dynamic Power Management）
- システムの運用中、使用していないコンポーネント（CPU、メモリ、周辺回路、周辺デバイスなど）を停止させる
  - コンポーネントが複数の電力モードを備えていることが前提
    - 通常モード（アクティブモード）
    - 1つまたは複数の停止モード
- 電力モードの例
  - アクティブモード
  - スタンバイモード
    - クロックの供給停止
    - 動的電力の削減
  - スリープモード
    - 電源の供給停止
    - 動的電力とリーク電力の削減

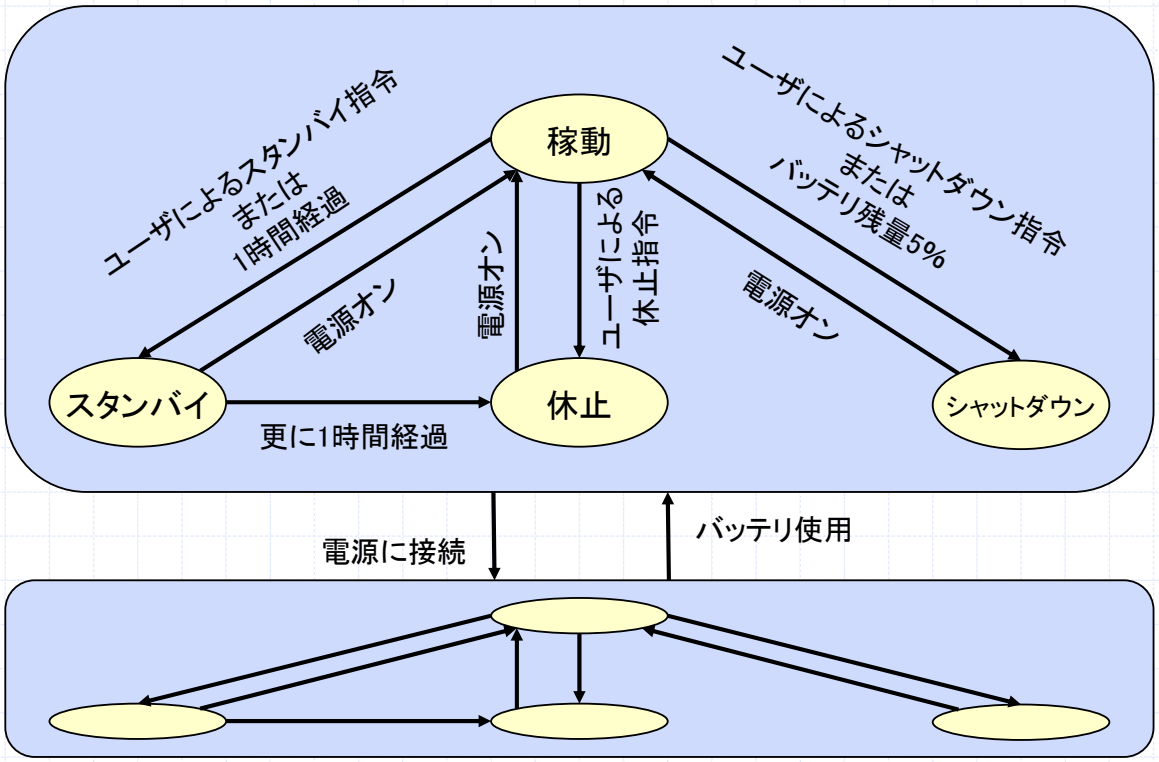


WindowsノートPCにおけるDPMの例

- モニタのDPM

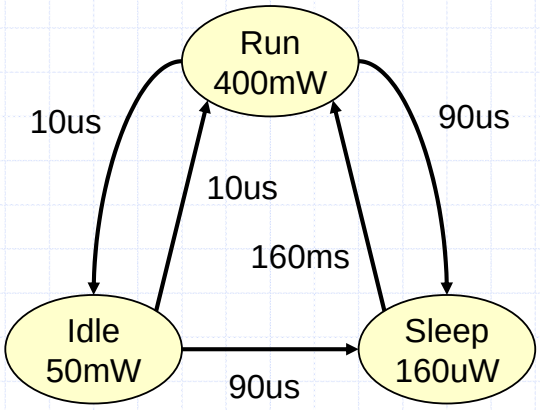


WindowsノートPCにおけるDPMの例



組込みプロセッサの電力モードの例

- StrongARM SA-1100



- ルネサスSH

	クロック	ロジック電源	メモリ電源	復帰時間
通常モード	ON	ON	ON	
Soft-Standby	OFF	ON	ON	即
Resume-Standby	OFF	OFF	ON	小
Ultra-Standby	OFF	OFF	OFF	長

DPMを支える回路技術

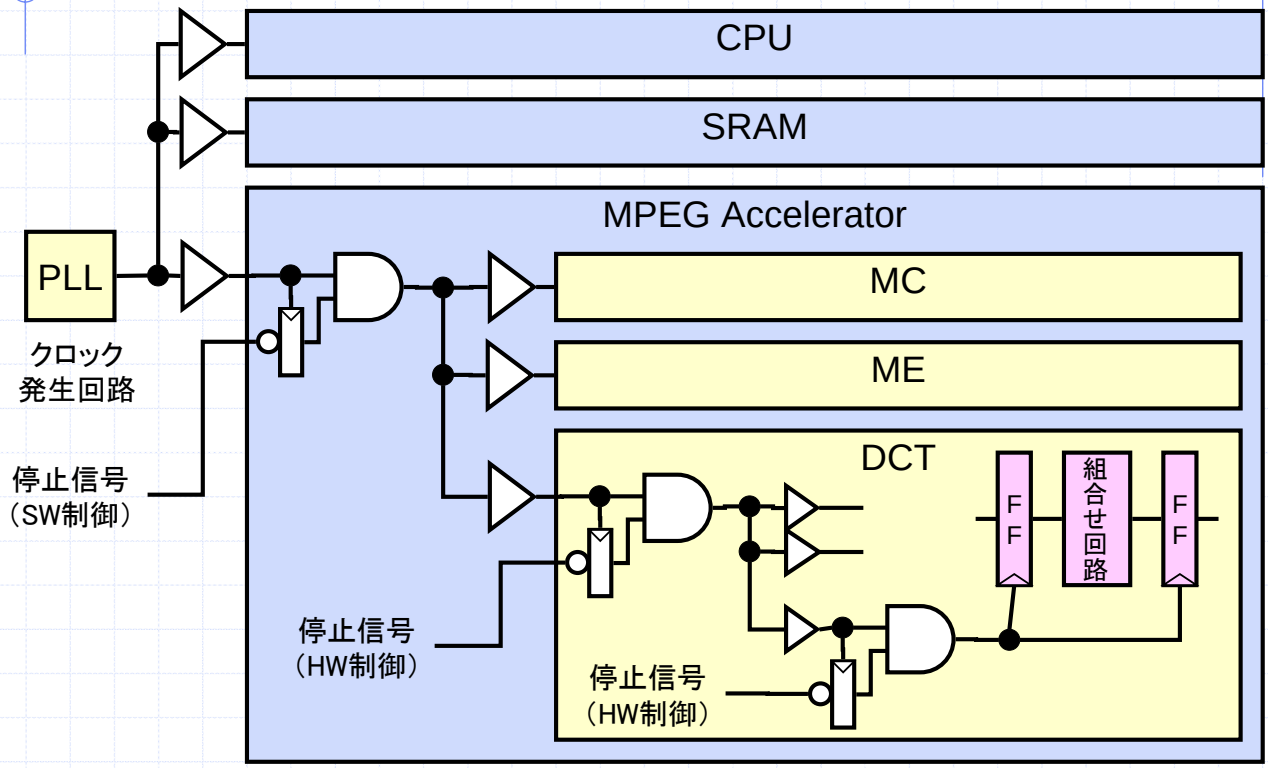
- クロックゲーティング
  - 使用していない回路のクロック供給を停止する
  - 階層的に制御されることが多い
- クロック発振停止
  - 発振器(オシレータ)を停止する
- 電源遮断
  - 使用していない回路の電源供給を停止する
- 動的電圧制御　⇒　後述
  - モジュールの電源電圧を上下させる
  - 電源電圧と同時にクロック周波数も上下させる
- 動的周波数制御　⇒　後述
  - クロック周波数を上下させる

クロックゲーティング

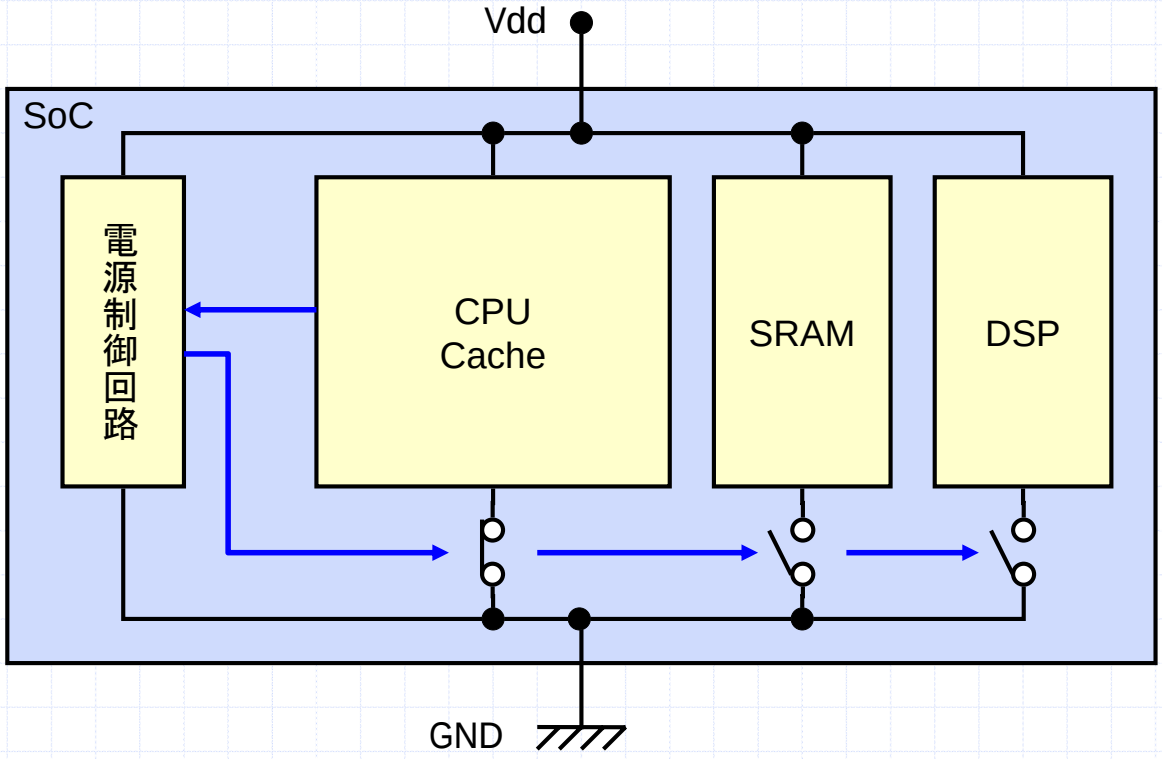
- 使用していない回路のクロック供給を停止する
- 階層的に制御されることが多い
  - 上位モジュールはソフトウェアが制御
  - 下位モジュールはハードウェアが制御
- スイッチング電力の削減
  - 無駄なスイッチング電力の削減
    - 特に停止させない限り、全てのモジュールは並列に動作する
    - 記憶素子(フリップフロップなど)へのクロック供給を停止すると、記憶素子の値が変化しなくなるため、組合せ回路も動作しない
  - クロック自身のスイッチング電力の削減
    - クロックの消費電力はLSIの全消費電力の最大3割程度を占める
      - クロック信号線の負荷容量は大きい
        - 全ての記憶素子に送られる
      - 活性化率は100%

$$P = C \cdot A \cdot f \cdot Vdd^2 + Vdd \cdot Ileak$$

階層的クロックゲーティング

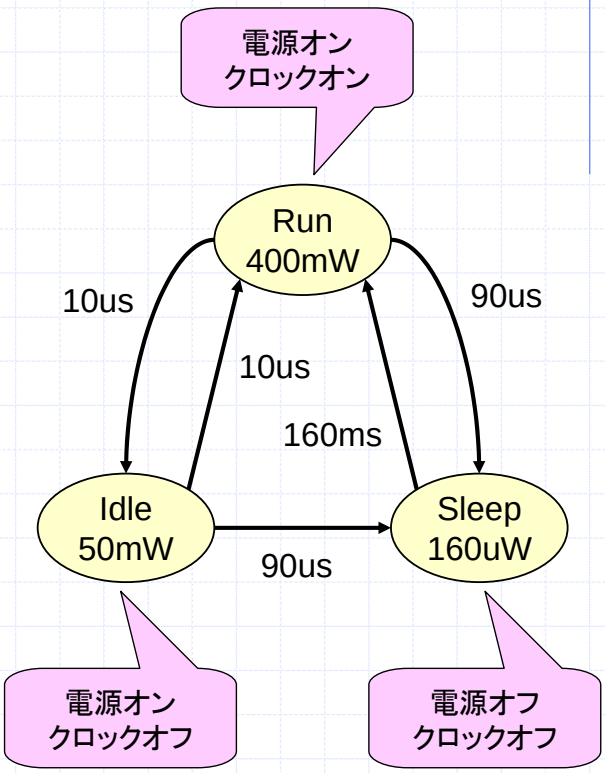


部分的電源遮断



DPMのハードウェア機構のまとめ

- 各コンポーネントは、通常モード（アクティブモード）に加え、1つまたは複数の停止モードを有する
  - クロックのオン／オフ
  - 電源のオン／オフ
- 各モードでは消費電力が異なる
- モードの遷移には遅延とエネルギーを要する
  - モードによって異なる

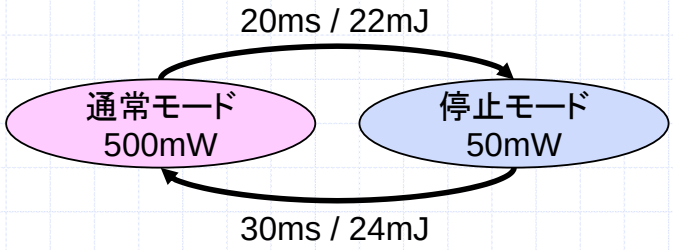


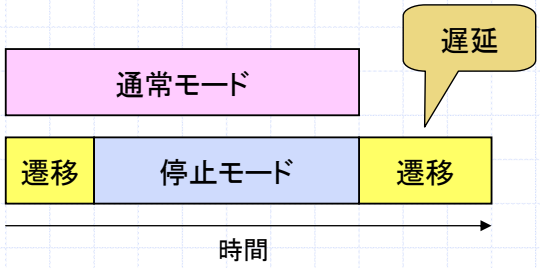
DPMソフトウェア設計上の検討事項

- どのような条件のときに停止モードに遷移させるか？ （DPMのポリシー）
  - タイムアウト方式
    - 一定時間
    - ユーザ指定時間
    - 適応型（学習型）
  - イベント方式
    - ユーザからの指示
    - ユーザからの指示以外
  - イベント方式とタイムアウト方式の併用
- 誰がモード遷移を制御するか？
  - アプリケーションソフト／ミドルウェア
  - （リアルタイム）OS
    - DPMの対象が共有資源の場合
- コンポーネント、アプリケーション（製品）、ワークロード、ユーザ（シナリオ）などによって、最適な制御方法は異なる

DPMポリシーの基本

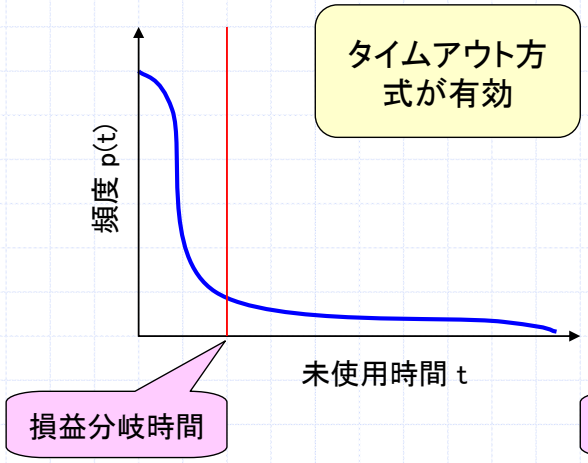
- 十分に長い時間、そのコンポーネントが使用されない場合、停止モードに遷移した方が良い
  - 通常モードを維持した場合の消費エネルギーよりも、停止モードに遷移した場合の消費エネルギーの方が小さくなること
- 損益分岐点となる時間（T_{BE}）は？
  - 通常モードを維持した場合：500mW × T_{BE}
  - 停止モードに遷移した場合：22mJ + 50mW × (T_{BE} − 20ms) + 24mJ
  - T_{BE} = 100ms
  - 当該コンポーネントが100ms以上使用されないならば、停止モードに遷移すべき

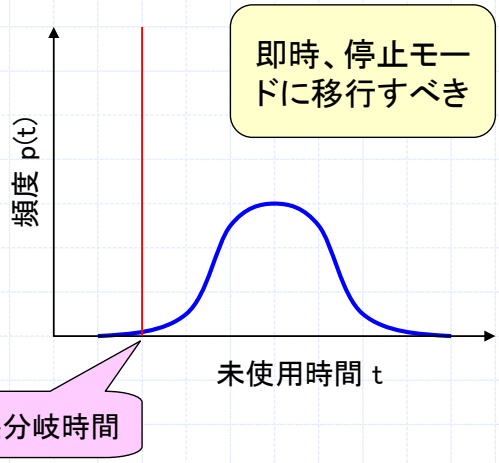




DPMポリシーの基本

- 一般的に、次回そのコンポーネントが使用されるまでの間隔を知ることは困難である
- しかし、統計的な傾向は（多かれ少なかれ）存在する
  - 当然、コンポーネント毎に傾向は異なる
- 未使用時間の確率分布を仮定する





リアルタイム応答性の問題

- モード遷移には遅延が伴う
- モード遷移の遅延が、システムのリアルタイム応答性を悪化させる恐れがある
  - CPU自身が通常モードから停止モードへの遷移を開始した直後に、何らかの要求（割込み等）が来た場合、「通常モードから停止モードへの遷移時間」と「停止モードから通常モードへの遷移時間」の両方がオーバーヘッドとなる恐れがある
- リアルタイムシステムにおいて、次回の使用時刻が不明であるコンポーネントを停止モードに遷移させるのは危険

動的電圧管理

- DVS: Dynamic Voltage Scaling
  - または、DVFS: Dynamic Voltage/Frequency Scaling
- 実行時に電源電圧とクロック周波数を上下させる
  - クロック周波数は電源電圧にほぼ比例
  - 動的消費電力は電源電圧の自乗とクロック周波数に比例
  - 動的消費エネルギーは電源電圧の自乗に比例
- 時間に余裕があるときは、電圧を下げて、ゆっくり動かす

回路遅延 $\tau \approx k\,C\,\frac{Vdd}{(Vdd - Vth)^2} \propto \frac{1}{Vdd}$

$f \propto Vdd$

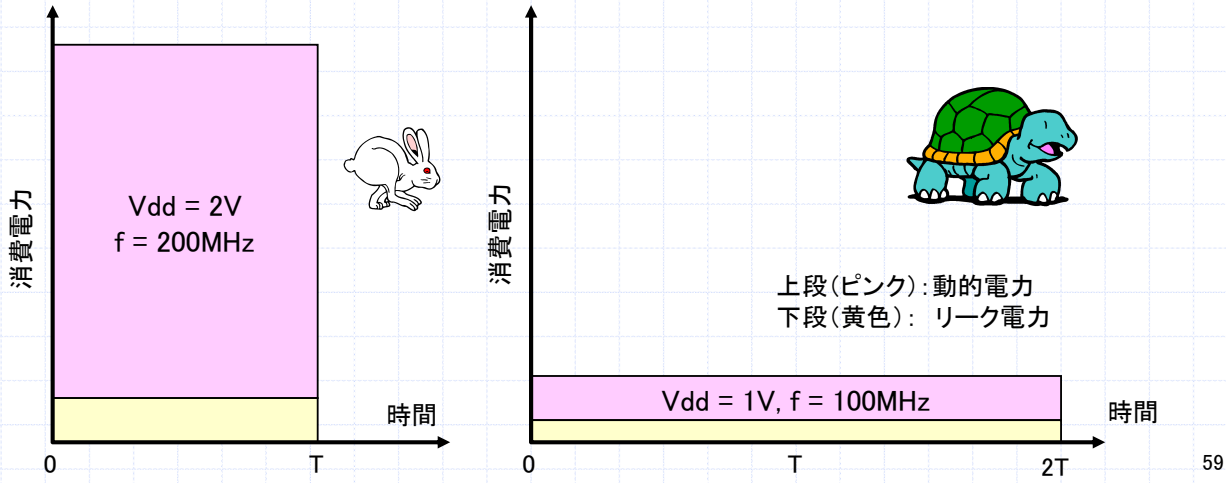
$P = C \cdot A \cdot f \cdot Vdd^2 + Vdd \cdot Ileak$

$E = P \cdot T = C \cdot A \cdot Vdd^2 \cdot Cycles + Ileak \cdot Vdd \cdot T$

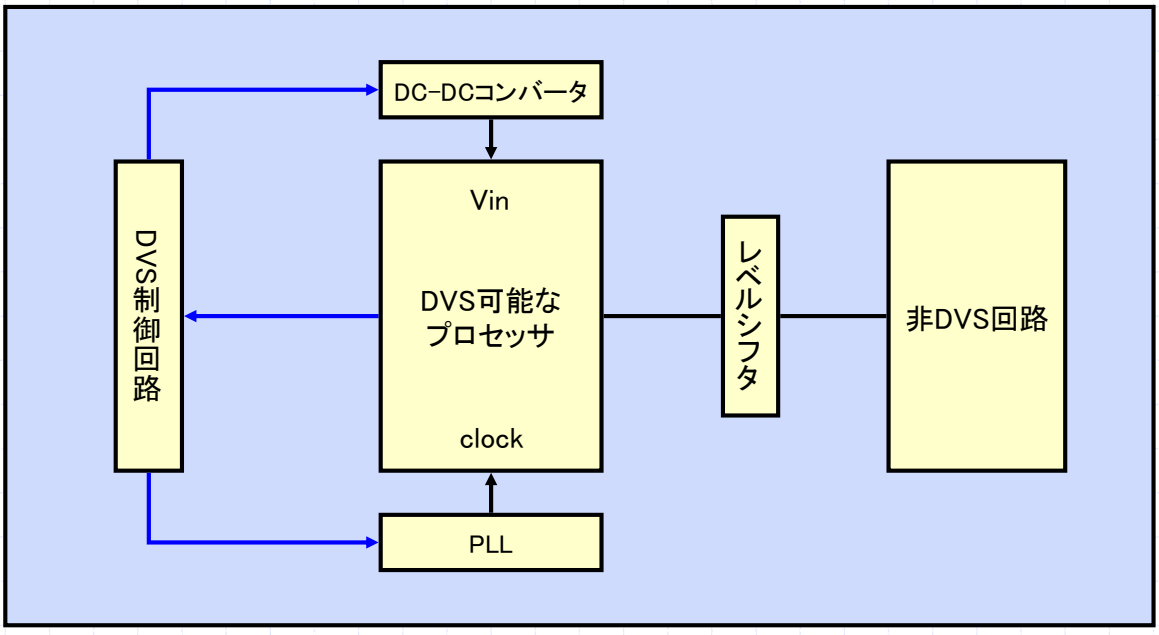
人間も運動する（エネルギーを消費する）ときには、血圧が上がり、脈が速くなる
安静時（エネルギーを消費しないとき）には、血圧が下がり、脈が遅くなる

DVS対DPM？

- 時間に余裕がある場合、余った時間に電源供給を止めるよりも、電圧を下げてゆっくり動かした方が消費エネルギーは小さい（理想的な場合）
- 電源電圧を1/2に下げる（周波数も1/2に下げる）と
  - 動的電力は1/8、動的消費エネルギーは1/4
  - リーク電力は1/2、リークによる消費エネルギーは変わらない



DVSのハードウェア構成の例



DVSハードウェアに関する留意事項

- 設定可能な電圧と周波数の組は有限である
  - 任意の値に電圧と周波数を設定できる訳ではない
- 電圧と周波数を変えるためには遅延を要する
  - 数十マイクロ秒～数ミリ秒
- 電圧を変えるための回路(DC-DCコンバータ)で、電力が損失される
- LSI全体の電圧を変えられるわけではない
  - 電圧を変えられるのは、主としてロジック部分
  - 電圧を下げられない部分
    - メモリ、入出力インタフェースなど
- 電圧が異なる回路間の通信には、電圧の差を調整する回路(レベルシフタ)が必要
  - 1～数サイクルの遅延
- リークによる消費電力は減らない
- 信頼性が低下する恐れがある
  - 信頼性が下がらないように回路設計されてはいるが

DVS可能な商用プロセッサ

Processor	Clock Range	Voltage Range	Transition Time
Transmeta Crusoe	200 – 700 (MHz)	1.1 – 1.65 (V)	300 us
AMD Mobile K6	192 – 588 (MHz)	0.9 – 2.0 (V)	200 us
Intel PXA250	100 – 400 (MHz)	0.85 – 1.3 (V)	500 us
Compaq Itsy	59.0 – 206.4 (MHz)	1.0 – 1.55 (V)	189 us
TI TMS320C55x	6 – 200 (MHz)	1.1 – 1.6 (V)	3.3 ms (1.6 -> 1.1 V) 300 us (1.1 -> 1.6 V)
IBM 405LP	66 – 333 (MHz)	1.0 – 1.8 (V)	13 – 95 us

DVSソフトウェア設計上の検討事項

- いつ電圧を切り替えるか？
- 何ボルトに設定すべきか？
- 誰が電圧を制御するか？
  - アプリケーションソフト／ミドルウェア
    - アプリケーションの性質を活用した詳細な制御が可能
    - 割込み処理やマルチタスク処理の場合は注意が必要
  - （リアルタイム）OS
    - システム全体として低消費エネルギー化が可能
    - マルチタスク環境では原則としてOSが制御すべき

最適な電圧：理想的な場合

- 以下の仮定を行う
  - 任意の電圧に変更可能である
  - 瞬時に電圧を変更可能である
    - 時間と消費エネルギーのオーバーヘッドがない
  - タスクは唯一である
- タスクのデッドラインをD、実行時間をTとする
  - Tは電圧Vの関数であることに注意
    - Tは周波数fの関数、周波数fは電圧Vの関数
- T(V)＝DとなるVが消費エネルギーを最小化する
  - 途中で電圧を変更せず、出来る限りゆっくり動かすのが最適解

出典：
T. Ishihara, H. Yasuura,
Voltage scheduling problem for dynamically variable voltage processors,
International Symposium on Low Power Electronics and Design (ISLPED), 1998.

利用可能な電圧が有限種類の場合

- 以下の仮定を行う
  - N種類の電圧と周波数の組が利用可能である
    - $(V_1, f_1), (V_2, f_2), \ldots, (V_N, f_N)$
      - $V_1 < V_2 < \ldots < V_N$
      - $f_1 < f_2 < \ldots < f_N$
  - 瞬時に電圧を変更可能である
  - タスクは唯一である
- タスクのデッドラインをD、実行時間をTとする
- タスクの消費エネルギーを最小化するためには、高々2種類の電圧を使用すればよい
  - タスクの実行中に1回だけ電圧を切り替えればよい
- $T(V_{ideal})=D$とすると、$V_i<V_{ideal}<V_{i+1}$となるV_iとV_{i+1}（つまり、理想的な電圧V_{ideal}の両隣）が最適な電圧である

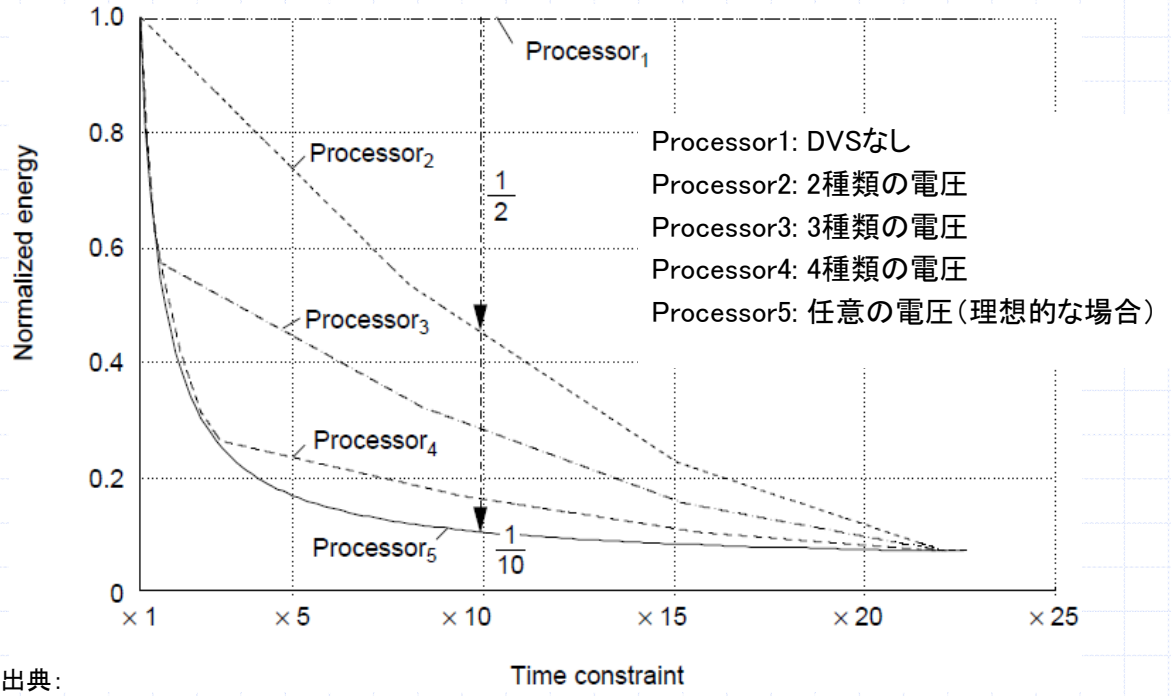
出典：
T. Ishihara, H. Yasuura,
Voltage scheduling problem for dynamically variable voltage processors,
International Symposium on Low Power Electronics and Design (ISLPED), 1998.

利用可能な電圧が有限種類の場合

- 2種類の電圧（V_iとV_{i+1}）を切り替えるタイミングは？
- 以下の記号を定義する
  - D：デッドライン
  - T(V)：電圧Vで実行したときの実行時間
  - α：電圧V_iで実行させる割合
    - 電圧V_{i+1}で実行させる割合は$1-\alpha$
    - 消費エネルギーを最小化するαを求めたい
- $\alpha \cdot T(V_i)+(1-\alpha)\cdot T(V_{i+1})=D$となる$\alpha$が消費エネルギーを最小化する
  - 実行時間がちょうどデッドラインになるように切り替える
  - V_iとV_{i+1}のどちらを先に実行しても構わない

出典：
T. Ishihara, H. Yasuura,
Voltage scheduling problem for dynamically variable voltage processors,
International Symposium on Low Power Electronics and Design (ISLPED), 1998.

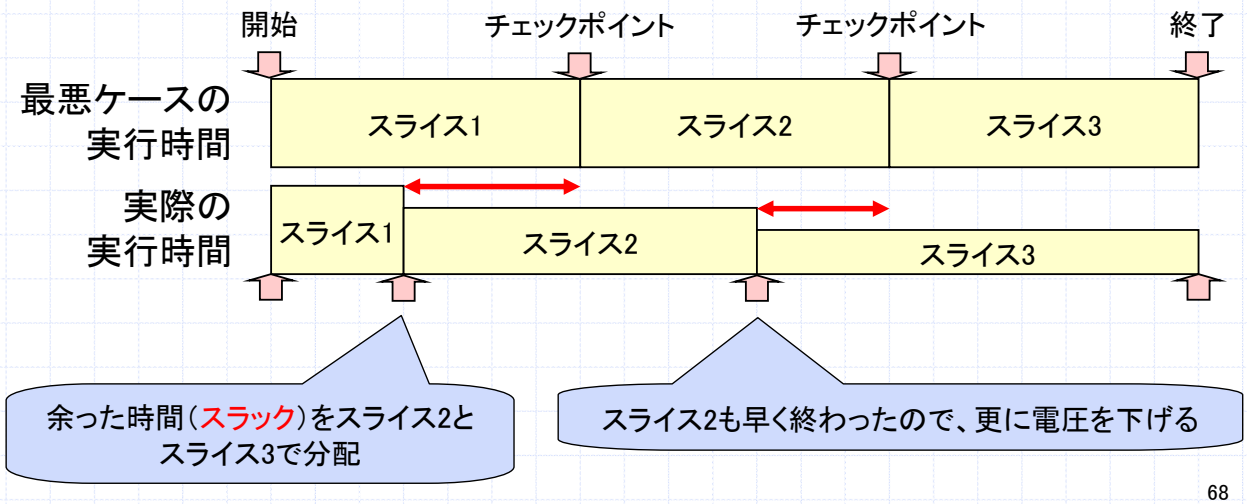
利用可能な電圧の数と消費エネルギー



出典：
T. Okuma, H. Yasuura, T. Ishihara,
Software energy reduction techniques for variable-voltage processors,
IEEE Design & Test of Computers, vol. 18, no. 2, 2001.

実行時間の変動の活用

- プログラムの実行時間は一定ではない
  - 入力データなどに大きく依存する
  - 最悪ケースよりも早く終わることが多い
- プログラムの途中途中（チェックポイント）で実行時間を監視する
- 最悪ケースよりも早くチェックポイントに到達したら、電圧を下げる



マルチタスクシステムのDVS

- 非リアルタイムシステム
  - CPU負荷が高いときは高い電圧で、CPU負荷が低いときは低い電圧で動作させる
    - 過去と現在の負荷から将来の負荷を予測する
- リアルタイムシステム
  - 1990年代の後半から現在まで、DVSプロセッサに対するスケジューリング理論の研究が盛んに行われてきた
    - 従来のスケジューリング理論に対する拡張
    - スケジューリングの際に電圧を決定
  - 大きく2種類に大別
    - 静的DVSスケジューリング
    - 動的DVSスケジューリング

DVSスケジューリング

- 静的DVSスケジューリング
  - 設計時に各タスクの電圧を決定
    - 固定優先度スケジューリングと併用されることが多い
  - 各タスクのリリース時刻と実行時間が既知の場合に有効
  - 実行時間の変動によるスラック時間を有効に利用できない
- 動的DVSスケジューリング
  - 実行時に各タスクの電圧を決定
    - 動的スケジューリングと併用されることが多い
  - 実行時間の変動によるスラック時間を有効に利用可能
  - 電圧を決定するためのオーバーヘッドが大きくなる恐れがある

リアルタイムシステムへの適用

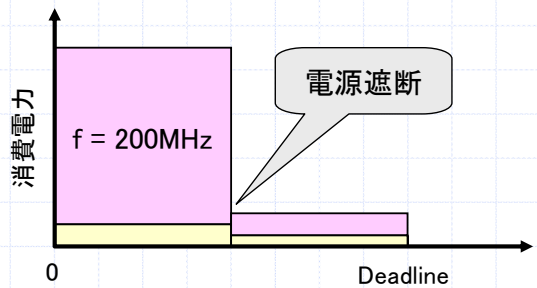
- ハードリアルタイムシステムに対しては、現在、DVSはほとんど適用されていない
  - 主として、電圧と周波数を変えるための遅延が無視できないため
    - 数十マイクロ秒～数ミリ秒
  - 実行時に予期せぬ事態が起こるかも知れないことを考えると、保守的に設計せざるを得ない
    - つまり、最高電圧で動作させ続ける
- ソフトリアルタイムシステムにおいては、QoSと消費エネルギーのトレードオフが可能
  - 時々デッドラインに違反することを許容すれば、消費エネルギーを大きく削減できる可能性がある
  - デッドラインミスの割合（QoS）と消費エネルギーのトレードオフ

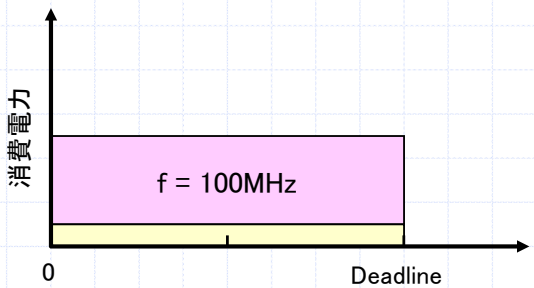
動的周波数制御

- 電源電圧は下げず、周波数だけを下げる
- 消費電力の削減には有効だが、動的消費エネルギーは削減されない
  - リーク電力を考えると、高い周波数で実行させた後、電源を遮断（DPM）した方が有利
- モード切替の時間やエネルギーのオーバーヘッドや、電源遮断の場合でも全ての回路の電源を遮断できる訳ではないことを考えると、周波数だけを低下させた方が有利なことも多い
- リアルタイム応答性の劣化が抑えられる

$$P = C \cdot A \cdot f \cdot Vdd^2 + Vdd \cdot Ileak$$

$$E = P \cdot T = C \cdot A \cdot Vdd^2 \cdot Cycles + Ileak \cdot Vdd \cdot T$$

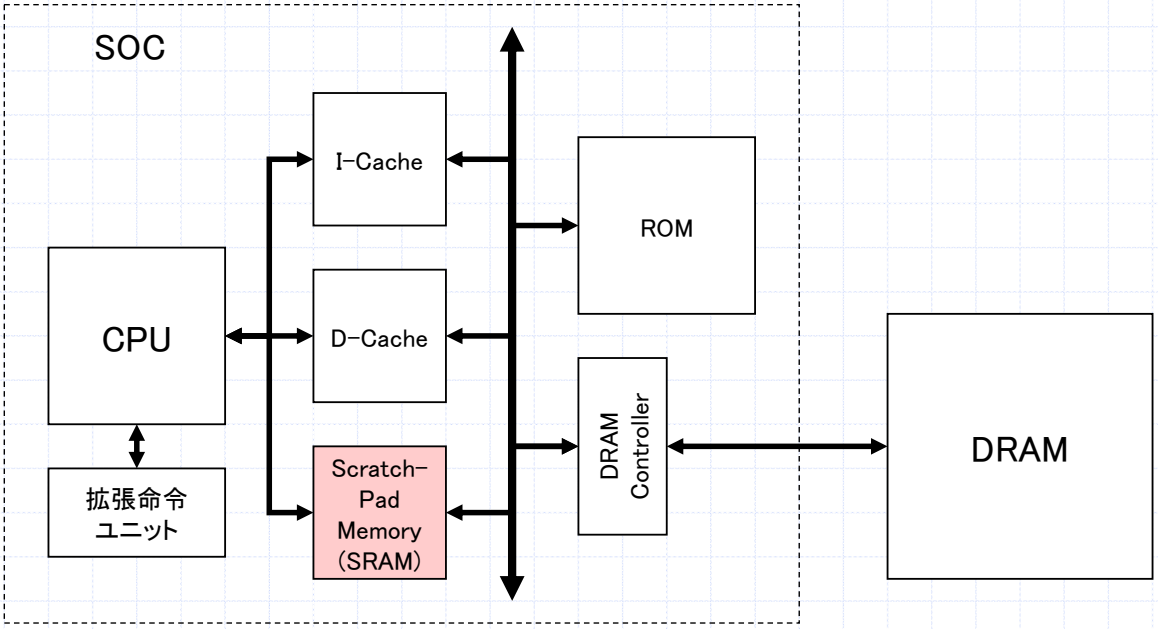




スクラッチパッド・メモリ

- Scratch-Pad Memory (SPM)
- 小容量で高速なオンチップSRAM
- ソフトウェアが明示的に制御
  - キャッシュは、ハードウェアが自動的に制御し、ソフトウェアからは透過である
- 通常、アドレス空間の一部がSPMに割り当てられる
- 従来は、リアルタイム・タスクの命令やデータを配置するために使用されることが多かった
  - キャッシュミスが発生しないため

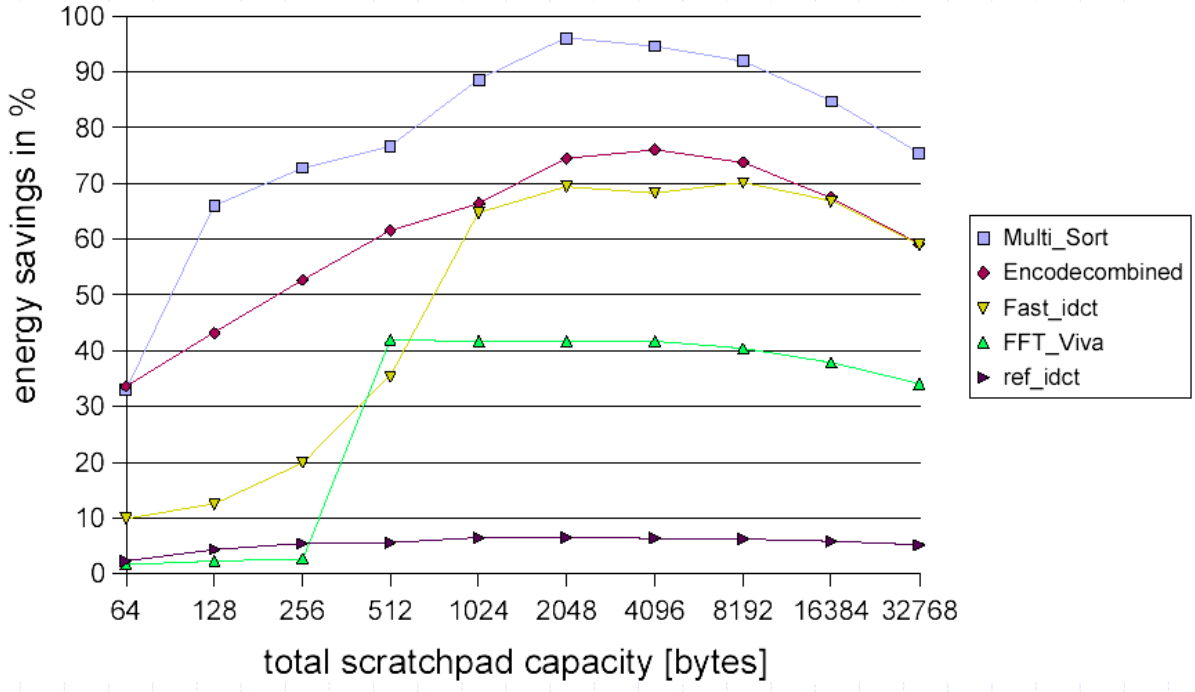
スクラッチパッド・メモリ



スクラッチパッド・メモリ

- SPMはキャッシュよりも低電力
  - タグの読み書きが不要 ⇒ 動的電力：小
  - タグ領域が不要 ⇒ リーク電力：小
  - 置換え制御が不要 ⇒ 動的電力：小
  - 必要なデータだけ読出し ⇒ 動的電力：小
    - キャッシュは1ライン分読み出す
- 頻繁に実行される命令列や、頻繁にアクセスされるデータをSPMに配置することにより、低電力化を実現できる
  - 例：ループの命令やデータ
- 欠点
  - 主記憶・SPM間のデータ転送のオーバーヘッド（時間と消費エネルギー）
  - プログラミングが困難
    - SPMを時間的に有効活用するためには、オーバーレイが必要

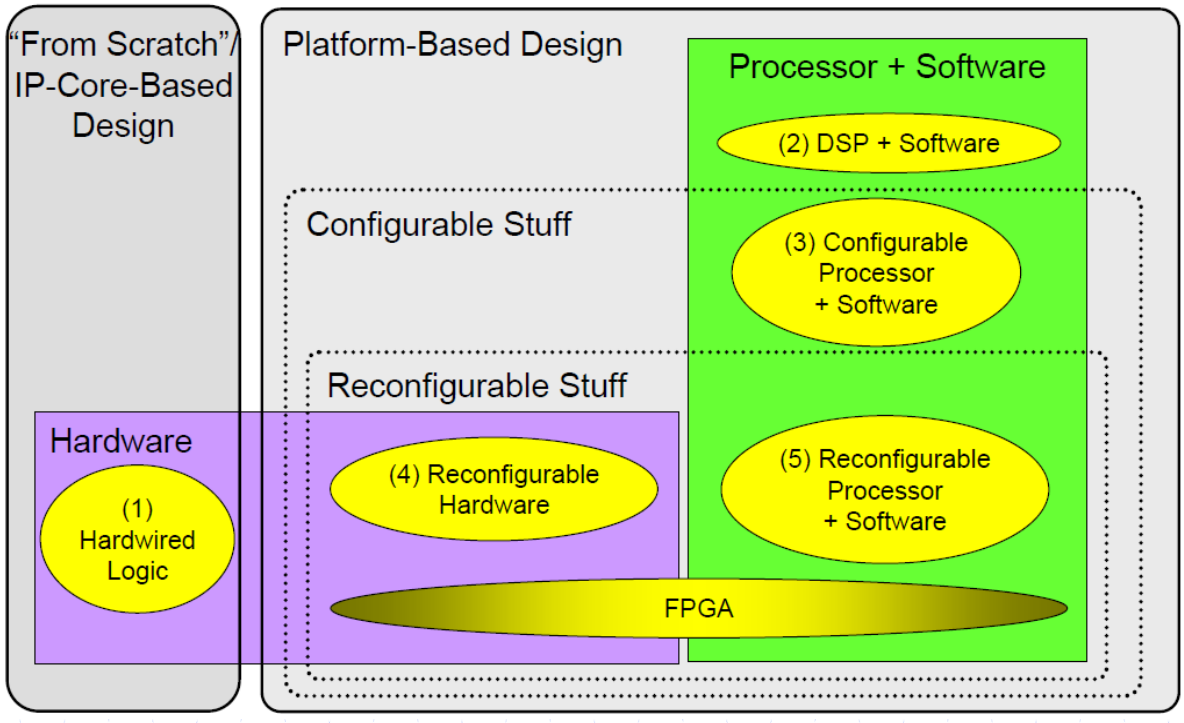
SPMの効果



出典：
P. Marwedel, Embedded System Design, Slides, Kluwer Academic Publishers, 2003.

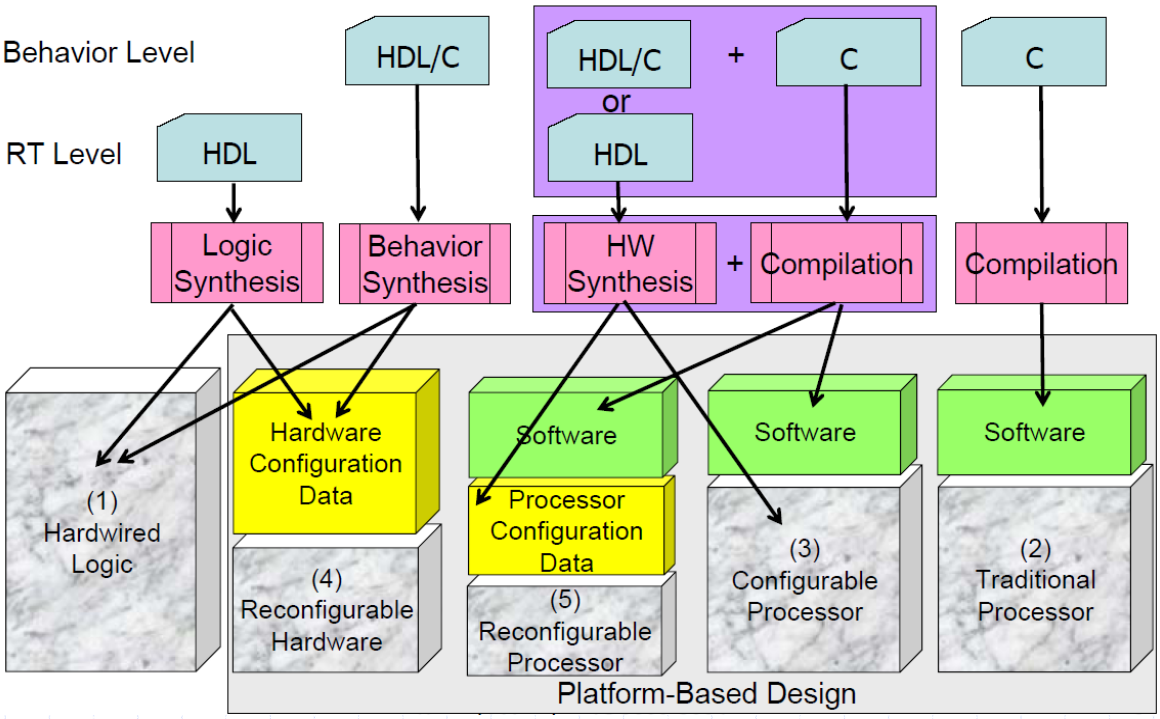
今後の展望

展望１：九州大学・村上教授による



Source: Kazuaki J. Murakami, "Five Ways to Design Future SoC," MPSoC 2004.

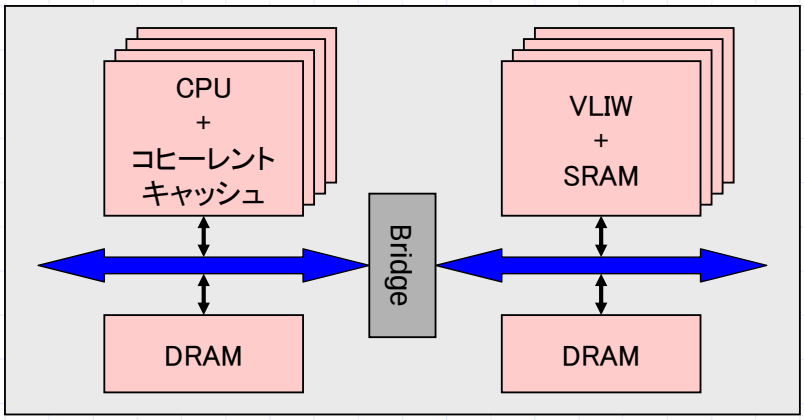
展望1：九州大学・村上教授による



Source: Kazuaki J. Murakami, “Five Ways to Design Future SoC,” MPSoC 2004.

展望2：現在の延長線上

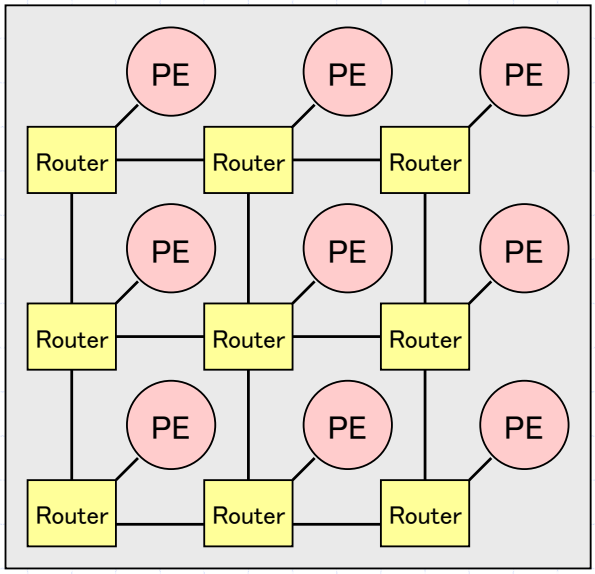
- SMP型アプリケーション・プロセッサ
  - ソフトウェア開発の負担を軽減
- NUMA型メディア・プロセッサ（VLIW/SIMD）
  - 高い演算性能と低消費電力の両立
- 大容量DRAM
  - 高いメモリバンド幅



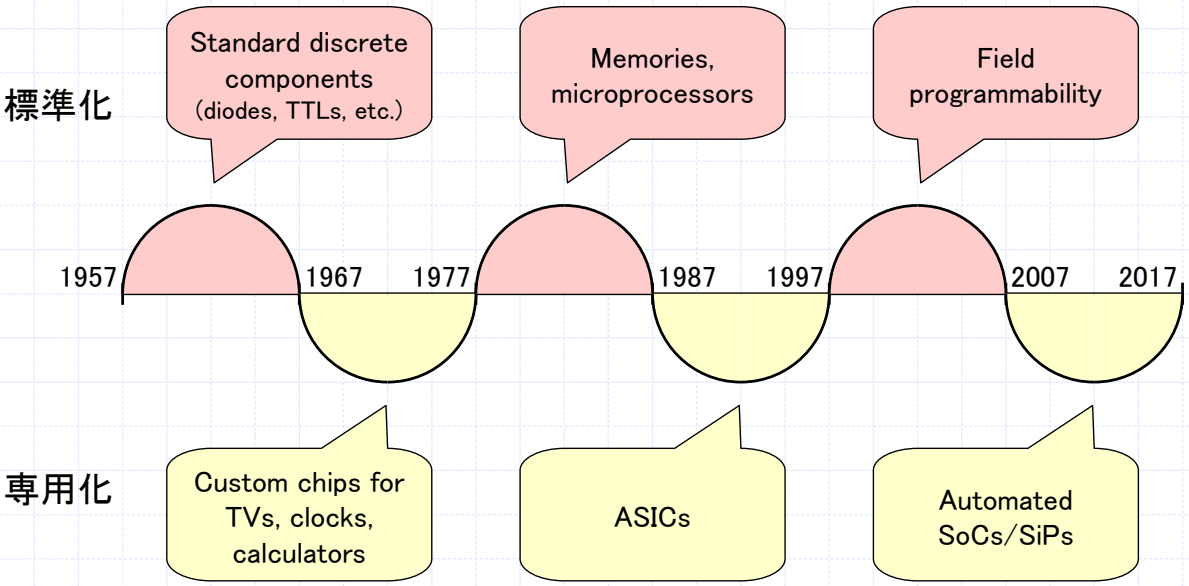
展望3：SoC～MPSoC、そしてNoCへ？

- Networks-on-Chip
- 最近5年間、EDAの分野で活発に研究されている
- 1チップ上に集積されたNORA（Message Passing）型マルチプロセッサ
- ノード内はクロック同期
- ノード間は非同期
- 利点
  - 微細化に伴い増大する配線遅延の問題を緩和
  - 高いスケーラビリティ

- メッシュ結合NoCの例



牧本ウェーブ：2007年は転換期



牧本次生氏（元・日立専務）による発見。
10年周期で標準化（standardization）と専用化（customization）の波が訪れる。