Empowered by Innovation NEC

プロセッサの性能向上の栄光と影
- 速度向上の曲がり角を過ぎて -

2010/10/01

松下　智

NEC　**システムIPコア研究所**

はじめに　- おわび

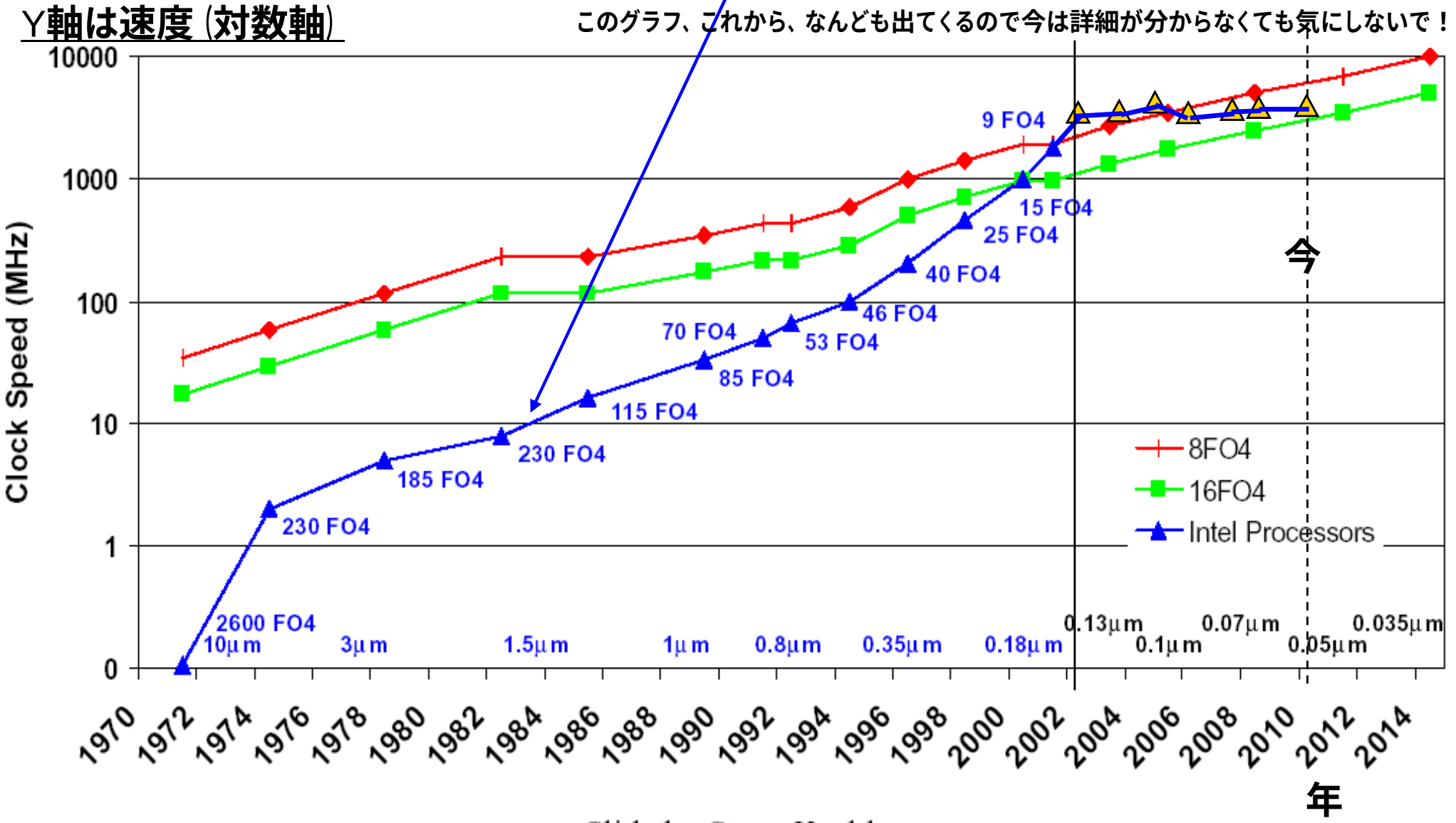
- たいへん、ごちゃごちゃしたスライドばかりですみません
- 時間がなくて、整理できていなくてすみません
- どんどん割り込んでください
- 脳が溶けるといわれています。悪酔いしたらすみません
- 私は、1993年から1997年まで米国SGI/MIPS社というところで、実際のプロセッサ開発をやりました。開発現場の情報とか多少通じていると思うので、前半でも突っ込んでくれれば幸いです
- 後半は、1994年から2005年ごろまで10年余やった研究成果です。私は途中で合流しました。今もって世界最先端だと自負していますが、死の谷を乗り越えられていません
  - 本当に良い技術は、とてもシンプルでわかりやすいです。
  - これもとてもシンプル
  - わかりにくいのは私の説明が悪いのです
- 後半まで、トークが進まない可能性が多々ありますが、興味があったら別途聞いてください

目次

パソコンのプロセッサ (CPU) の速度向上のグラフ

栄光) 2002年までは、10年でほぼ100倍、年率1.55倍の速度向上をしてきた

影) 2002年を境に止まってしまった: **一体何がおきたのか！**



Slide by Steve Keckler

CPUの速度向上の光と影：具体例

M: 100万、 G: 10億（Mの千倍）

- **今のパソコンは昔の大型コンピュータをはるかにしのぐ速度を持っている**
  - 1964年 IBM system/360 model 50 – **当時価格**3.4**億円で0.5MHz**
  - 1987**年に大学にあった、**VAX-II 750 **は、当時価格３千万円で、3.1MHz**
    - **最上位機種の**VAX-II 780**は、当時価格**1**億円で、5MHz**
  - 2004/1**月 インテル・プレスコット**: **3.4GHz** : VAX-II 750**の**1,000**倍の速度**
- **速度向上は**2002**年を境にほぼとまってしまった**
  - MacBookPro 15inch（2009**年**）： **2.53GHz** Core2Duo
  - 2010**年最新の**Desktop**最上位パソコン**：
    インテルCore i7-950 (4コア, 3.06GHz, L3キャッシュ8MB, QPI 4.8GT/s)
  - **携帯の速度向上は、上げどまりに近づきつつも続いている**
    - 2009**年**: iPhone3GS **の**CPU**は** ARM A8: **0.6GHz**
    - 2010**年**: iPhone4**の**CPU**は** ARM A8: **0.8GHz**

    これですら、1987**年の**VAX-II 750**の**260**倍の速度**

なぜ急速に向上してきた速度向上が止まってしまったのか

Intel Pentium4の発展と終焉（速度向上の曲がり角）

Pentium4 各世代の最高性能（最高価格）チップをピックアップ

発売時期	開発コード名	コア速度	2次キャッシュサイズ	プロセスルール	消費電力（TDP ＊）
2000/11月	ウィラメット	2.0GHz	0.25MB	180nm	75W
2002/1月	ノースウッド	3.4GHz	0.5MB	90nm	89W
2004/1月	プレスコット	3.4GHz	1MB	90nm	103W
2005/2月	プレスコット2M	3.8GHz	2MB	90nm	115W
2006/1月	シーダミル	3.0GHz	2MB	65nm	85W

1.0 GHz = **1秒間に10億演算**

プロセスルール：　最小の部品寸法　1nm = 10**億分の1メートル**
65nm ⇔ **人間の最小の細胞の寸法** 6,000nm **のさらに**1/100
（**タバコの煙の粒子** 10nm～500nm : **相当きれいな空気環境でないと作れない**）

注） TDP：Thermal Design Power: **熱設計電力**

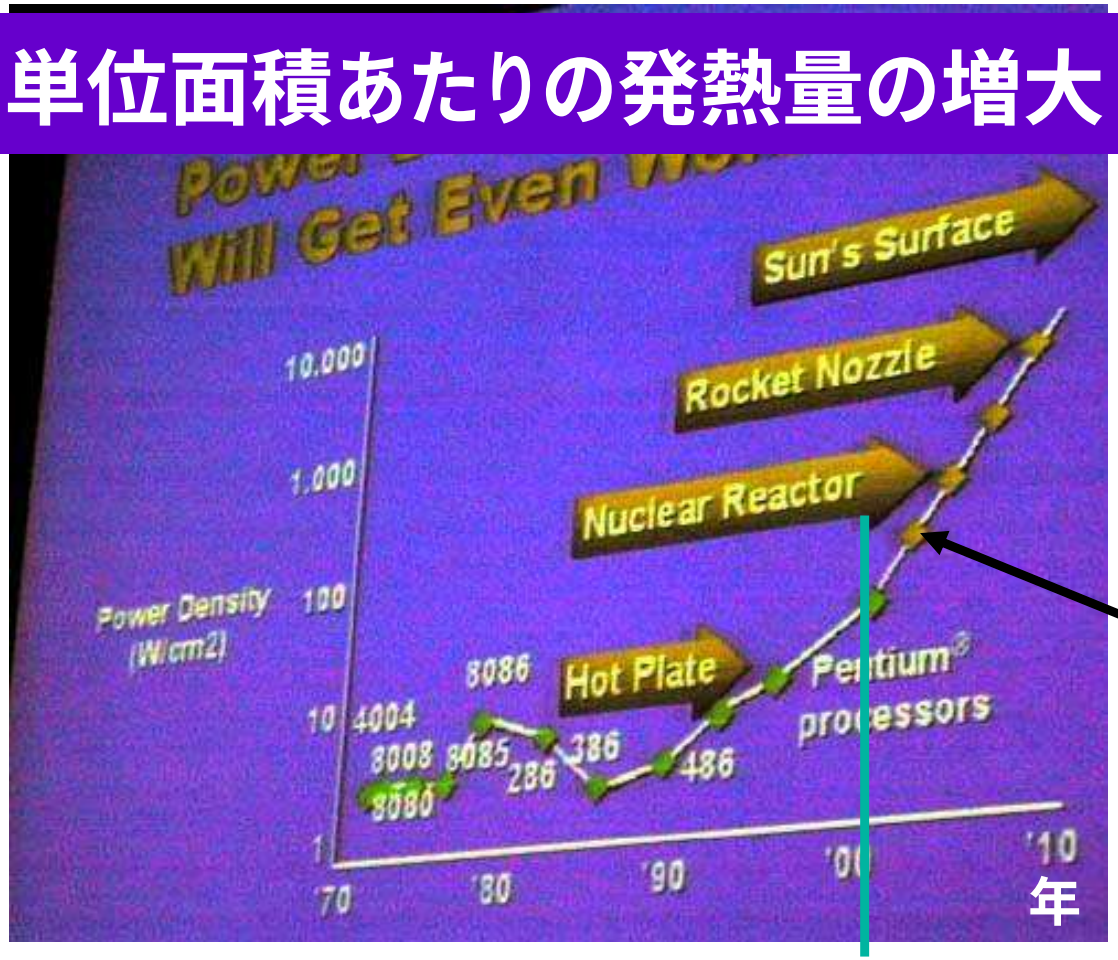
Ref: http://ja.wikipedia.org/wiki/Pentium_4

速度向上の曲がり角を招いた原因: 発熱問題

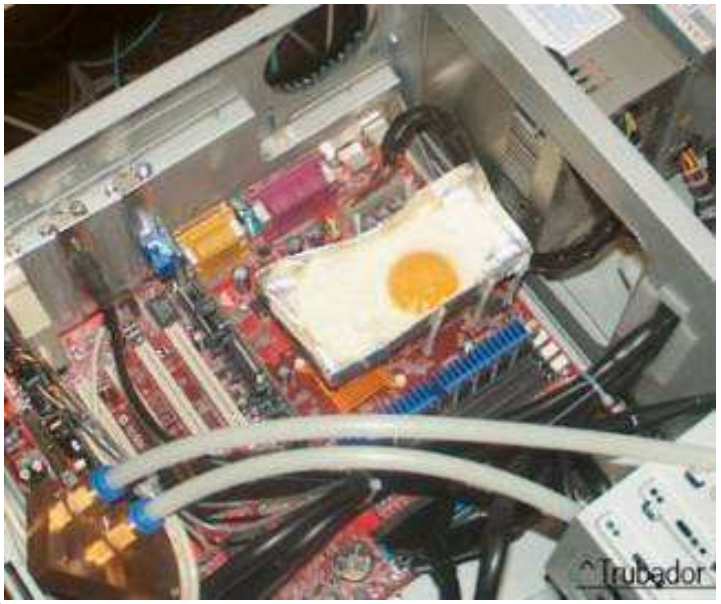
- 2001/2月 国際学会ISSCCの基調講演で、Intel社 副社長パット･ゲルシンガーが方向転換を発表
  - このままのペースだと、
    - 10年以内に消費電力が数千WのCPUができてしまう…
    - **単位面積あたりの発熱量で比較すると、ことの重大さが分かる**

（多少脚色はあるが。。。）

単位面積あたりの発熱量の増大



公演された時点

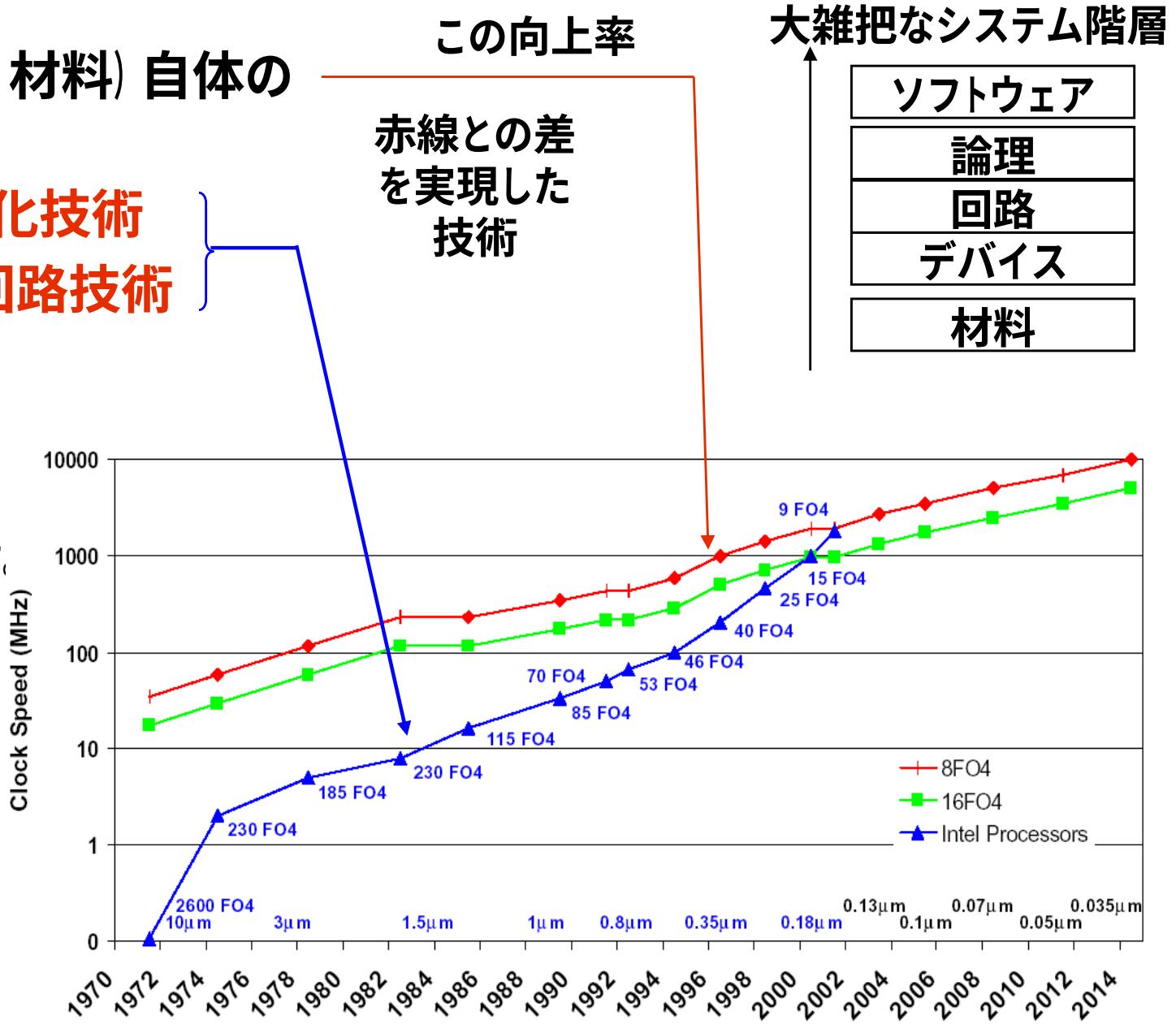


2004～2006年頃パソコンの
CPU上で目玉焼きを
作る遊びが流行った

性能向上の栄光を支えた技術

- 半導体（デバイス・材料）自体の性能向上
- パイプライン並列化技術
- 高速に計算する回路技術
- 命令の同時発行

（同じ速度で性能をあげるための技術：
道路でいえば車線を増やす。
グラフには見えないが
2倍強が限度）



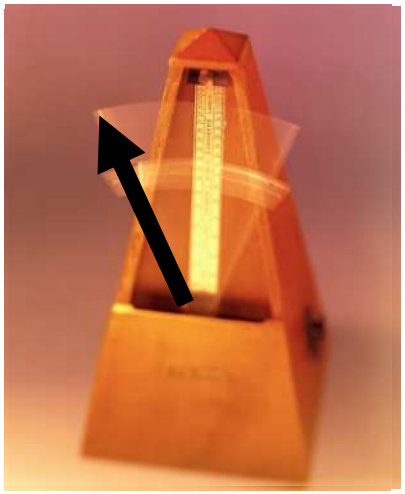
Slide by Steve Keckler

プロセッサ　(CPU) の計算原理

- やりたいことを簡単な処理指令 (機械命令列) に分解し、
  それをメトロノーム (クロック) に併せて順番に処理

R4,R5,R6,R7,R8はレジスタという中間結果の一時保存場所先

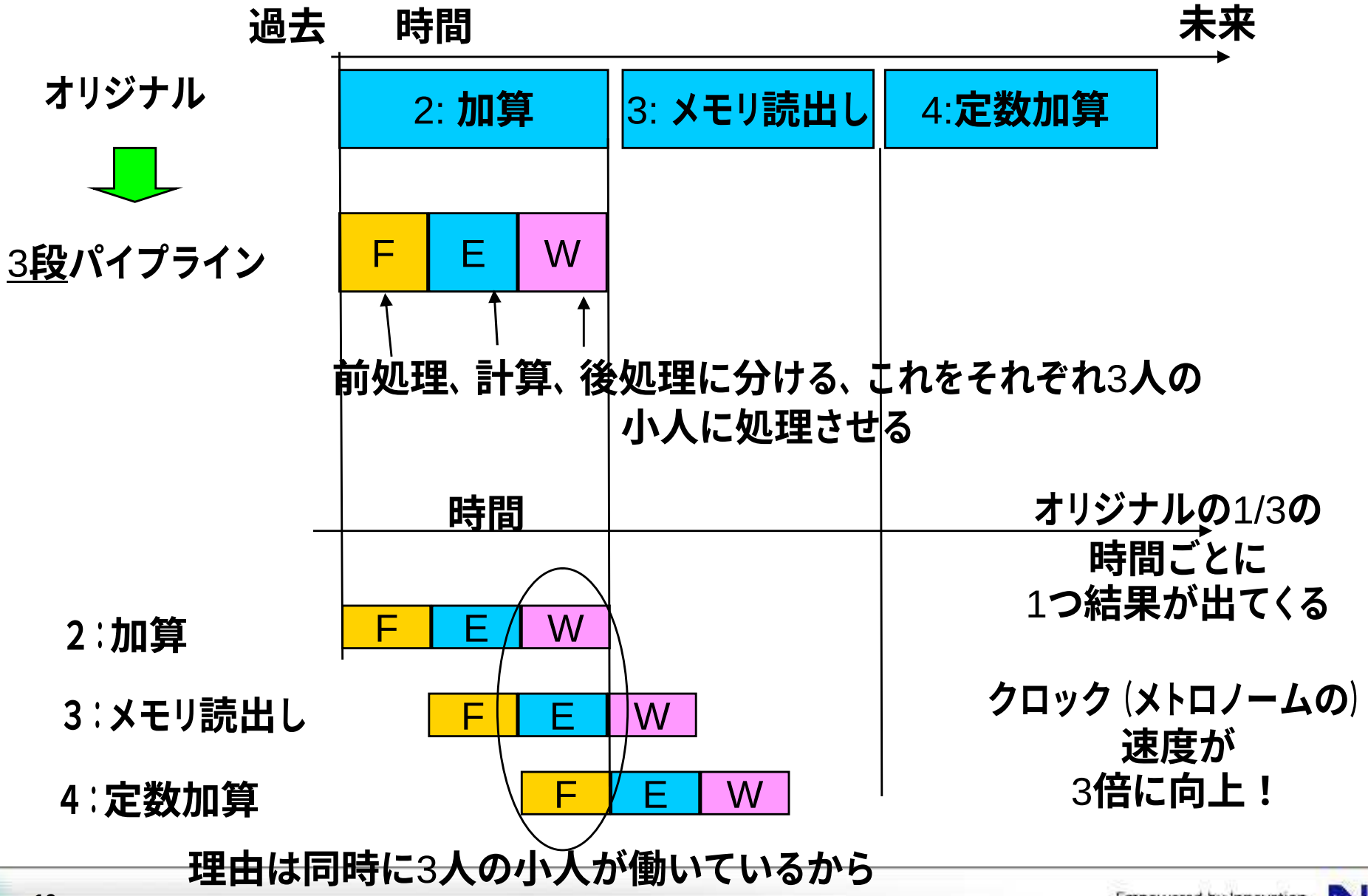
6: R8 <= R6 + R7 ; **加算**
5: R7 <= (R5)　; **メモリ読出**
4: R5 <= R5 + 4 ; **定数加算**
3: R6 <= (R5)　; **メモリ読出**
2: R5 <= R5 + R4 ; **加算**
1: R5 <= 10000 ; **定数代入**



機械命令列

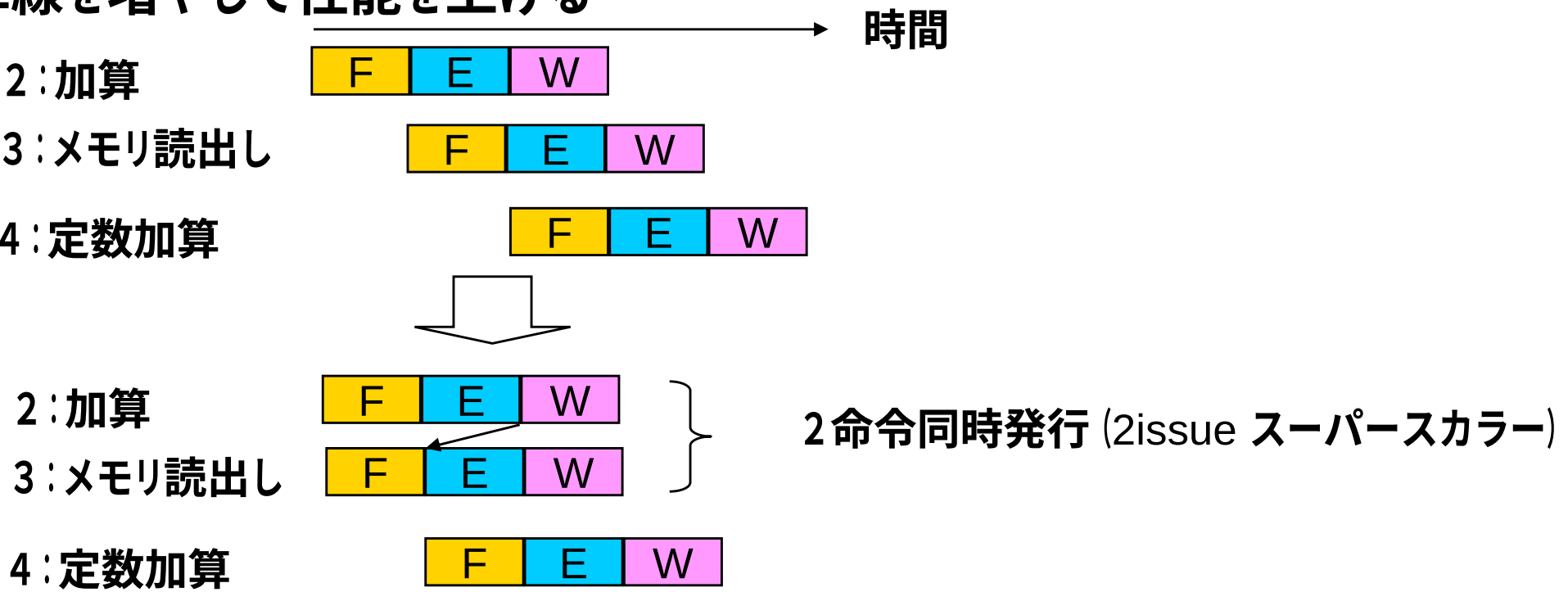
パイプライン処理による速度向上

処理を分割して、流れ作業にすることで処理を高速化する



命令の同時発行（スーパースカラー）の併用

車線を増やして性能を上げる



2命令同時発行（2issue スーパースカラー）

課題）　2：加算　の結果を 3:メモリ読み出し　で使いたいときは同時実行できない

いろいろ理由があり車線を闇雲に増やしても2倍強を限度に性能は向上しない
（理論計算の論文あり）

パイプライン段数、(命令同時発行数)

出荷開始

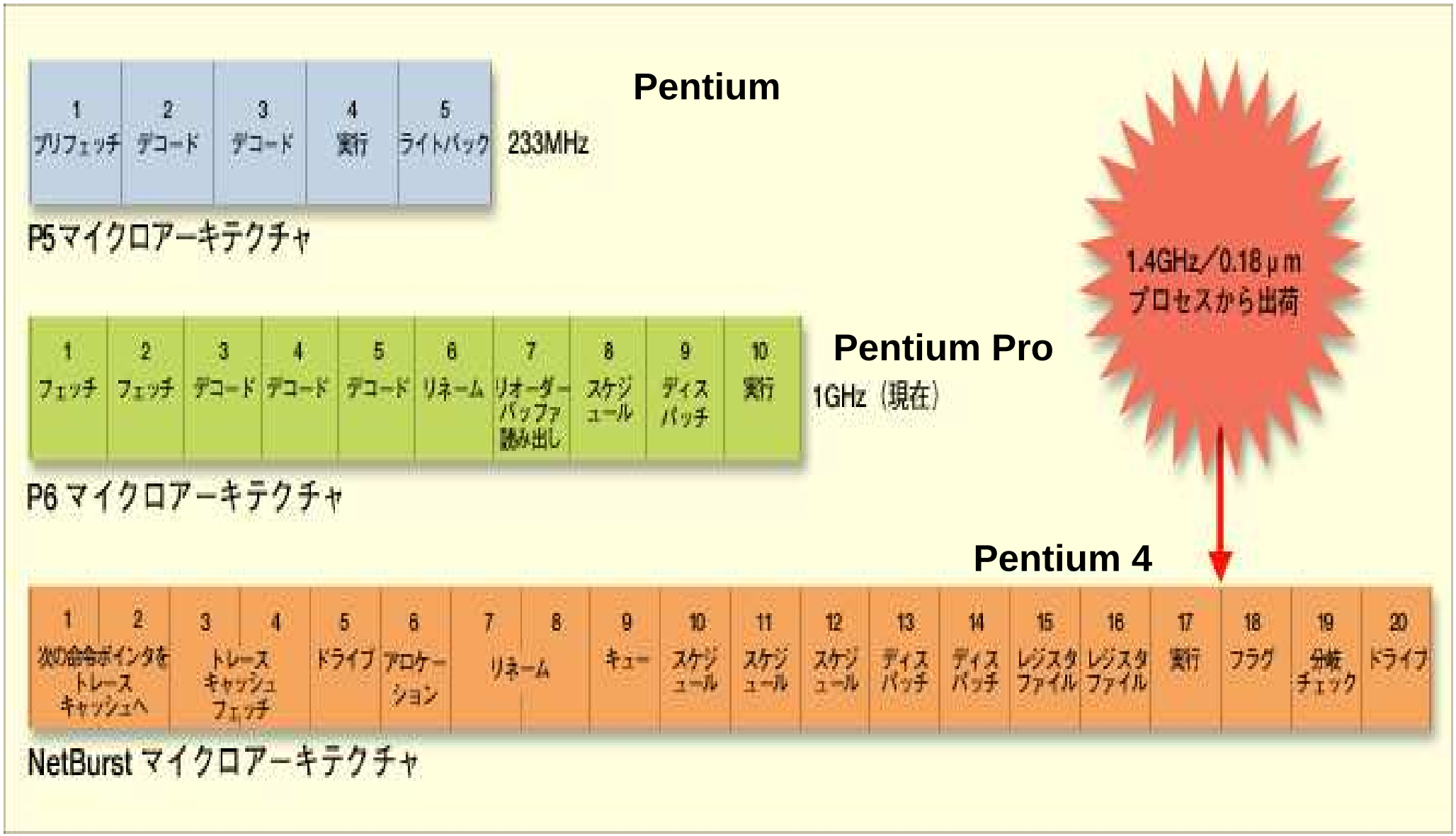
- 1983/5　Intel Pentium:　5段　(不完全な2命令同時発行)
- 1995/11 Intel PentiumPro:　12段　(uOPsを採用、完全な2命令同時発行)
- 2000/11 Intel Pentium4:　20段　(一部に31段という数え方もあり)
  (完全な3命令同時発行)

論文: “The optimal logic depth per pipeline stage is 6 to 8 FO4 inverter delays”で、
IBMは20段が最適と結論

- Intel曰く究極の段数:　45段

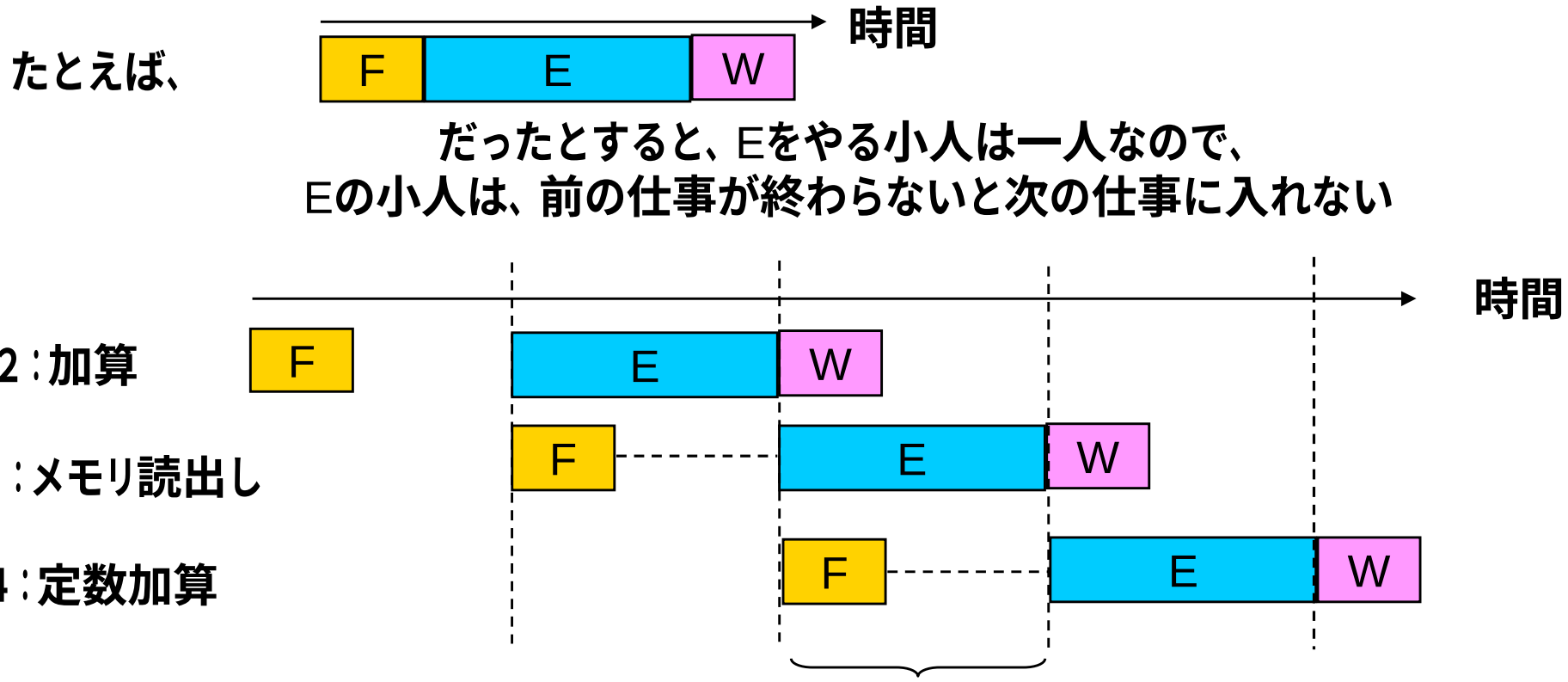
- 2006/1　Intel Core 2 Extream: 14段　(4命令同時発行)
- 2008/11 Intel Core i7:　14段　(4命令同時発行)

Intel CPUのpipelineの進化



パイプライン処理で性能させるのは簡単ではない

細分化した各作業にかかる時間にばらつきがあると性能が上がらない



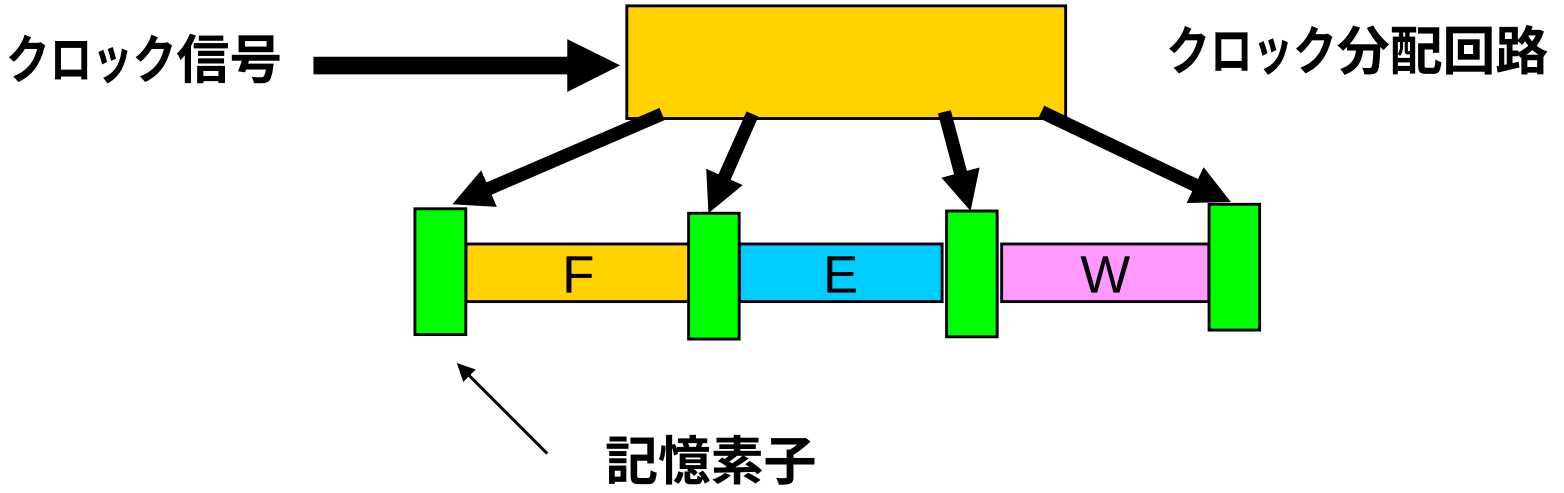
細分化によって記憶素子が必要になる

- よけいな時間がかかる　(仕事引渡しの手間)。**電気を食う**

ほかにも、分岐予測、データ依存等、解決しないとならない問題が山積 (深くは触れない)

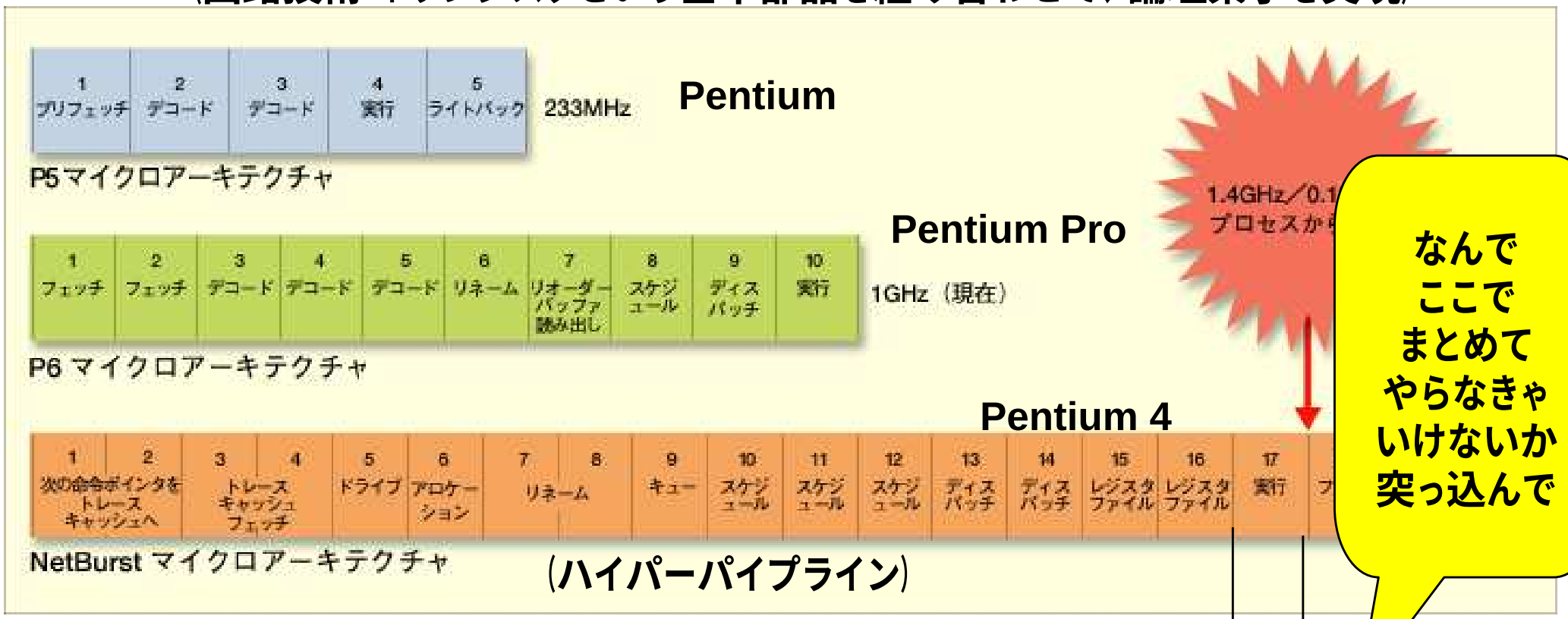
記憶素子とクロックの分配がくせもの

- 高いクロックを、正確に、記憶素子に分配しないとならない
- パイプラインを分割すると、クロックがあがるが、消費電力を増やす要因が急速に増える
  - 記憶素子の数が増える
  - 記憶素子は高速動作しないとならない
  - 高速なクロックを高精度に分配しないとならない
  - 速く動かすために電気を食うさまざまな特殊技術（回路技術）を導入する

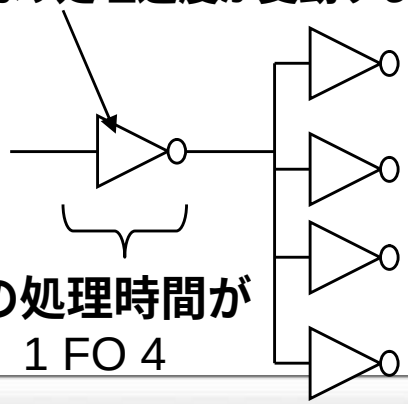


Intel CPUのpipelineの進化と回路技術

(回路技術:トランジスタという基本部品を組み合わせて、論理素子を実現)



インバータ: トランジスタ 2個からなり、入力信号を反転するだけのもっとも簡単な論理素子
負荷により処理速度が変動するので典型的な負荷数4で性能を示すのが一般的



この処理時間が
1 FO 4

➡ 9 FO 4は、1 FO 4の9倍の時間という意味

(10進20桁分)の足し算や
引き算をここでやる
この間を9 FO 4でこなす
すごい回路技術

すごい回路技術 - 1

- 7GHz => 0.14ns=100億分の1.4秒 ： 秒速30万Km （月まで1.2秒）の光ですら、2.8cm しか進めない時間* で、32bit (10進10桁分)の加減算をこなす
  - CPUのクロック3.5GHzの倍速で32bit計算を2回やって64bit（10進20桁分）の結果を得る
  - parallel pass-gate network - Diffusion Connected Network (DCN)
  - 高品質な（レイアウト）設計が必須
  - 低電力（低電圧）にならない、ＬＳＩ微細化に耐えられない古い!!!技術

（今の手法は、CMOS+CML）

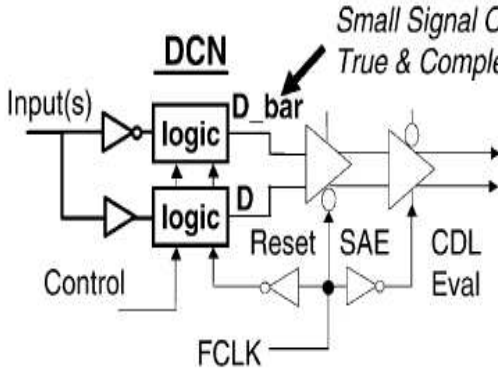
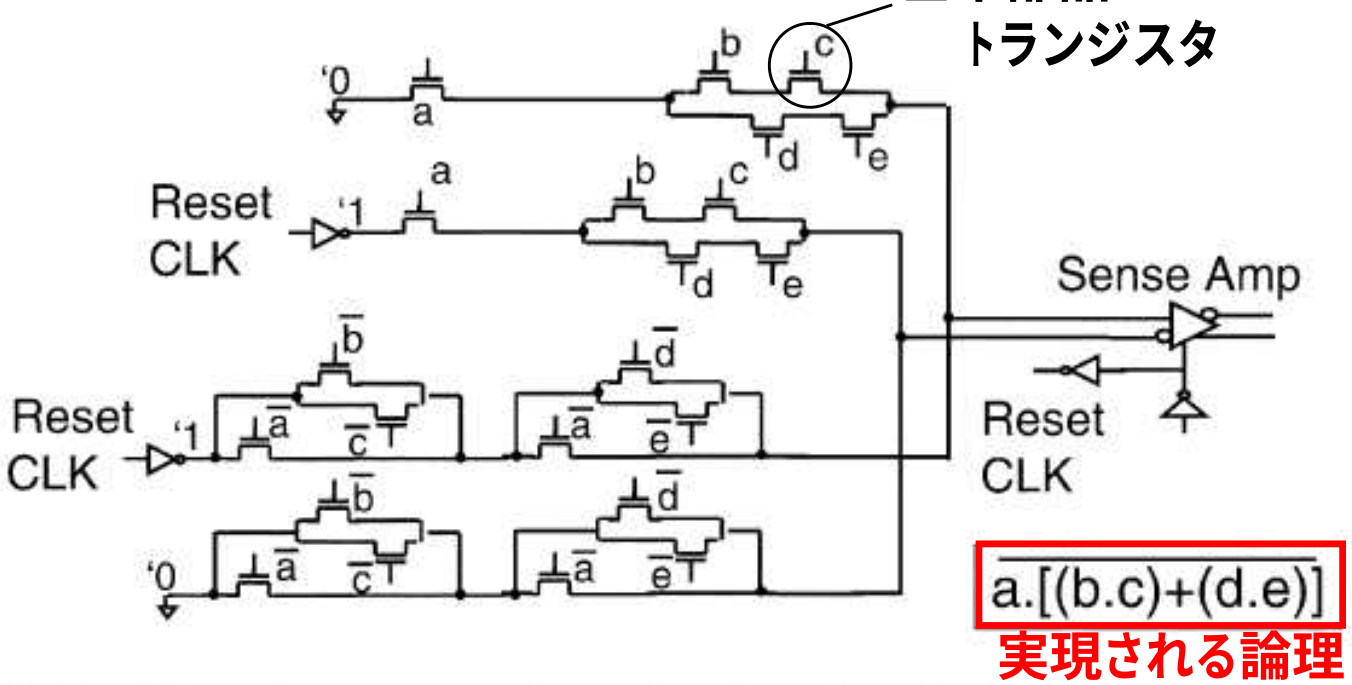


Fig. 3. Block diagram of chosen LVS circuit topology.

基本部品:NMOS トランジスタ



実現される論理

Fig. 5. Illustration of a complex random logic function DCN.

注） ＊ 同軸ケーブル･光ファイバー上での速度（誘電率考慮）。チップ上での速度は、充放電律速なのでもっと遅い

出展: “Low-Voltage Swing Logic Circuits for a Pentium4 Processor Integer Core”, Daniel J. Deleganes, et.al. , 41st DAC, 2004, pp.678-680

すごい回路技術 - 2　(なんか込み入っているなと思って欲しいだけ)

センスアンプとCDL

- CDL: Cross-Coupled Domino – ここで電源電圧の10%振幅のアナログ差動信号をフルスイングのCMOS信号に変換する

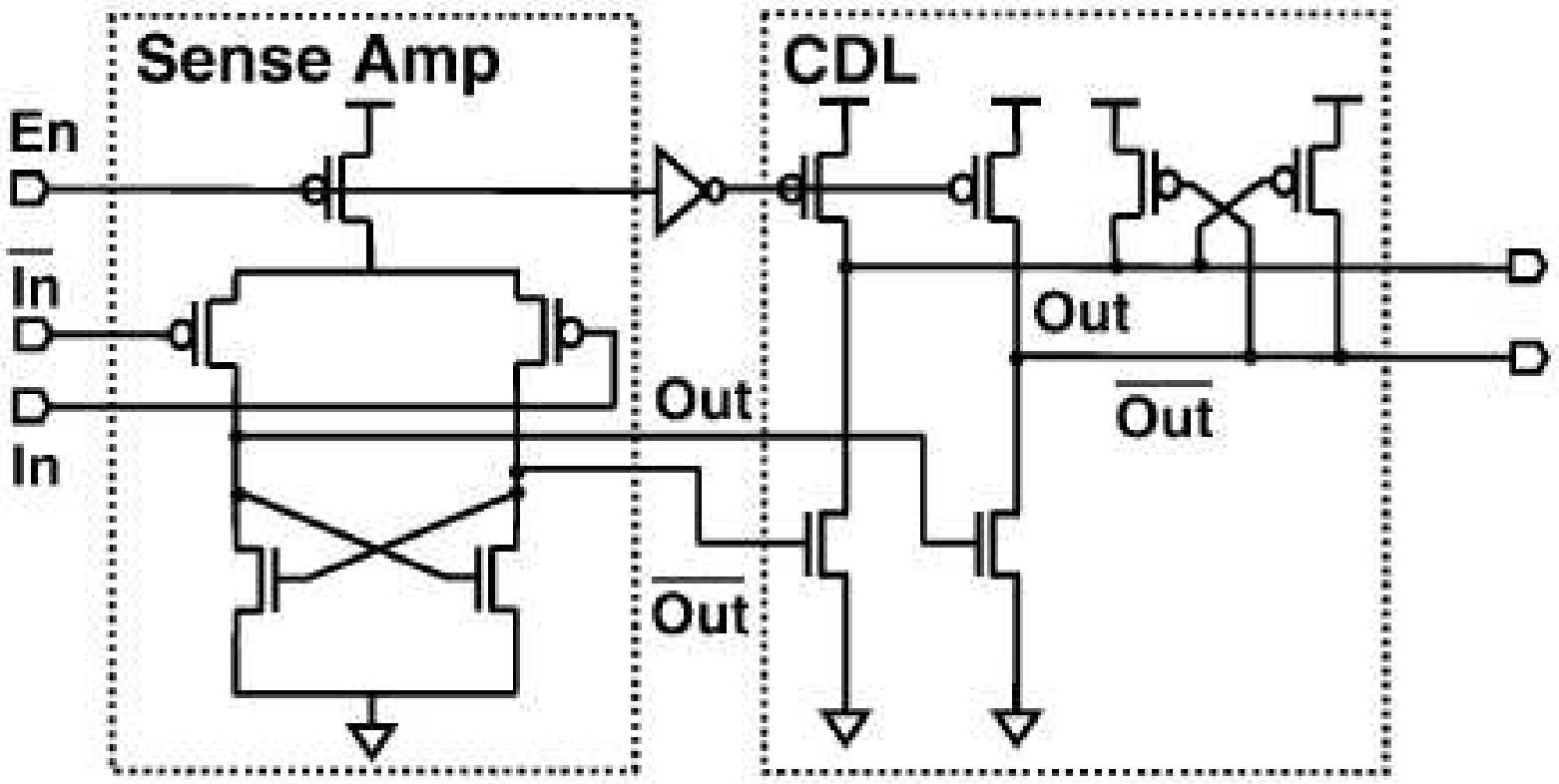
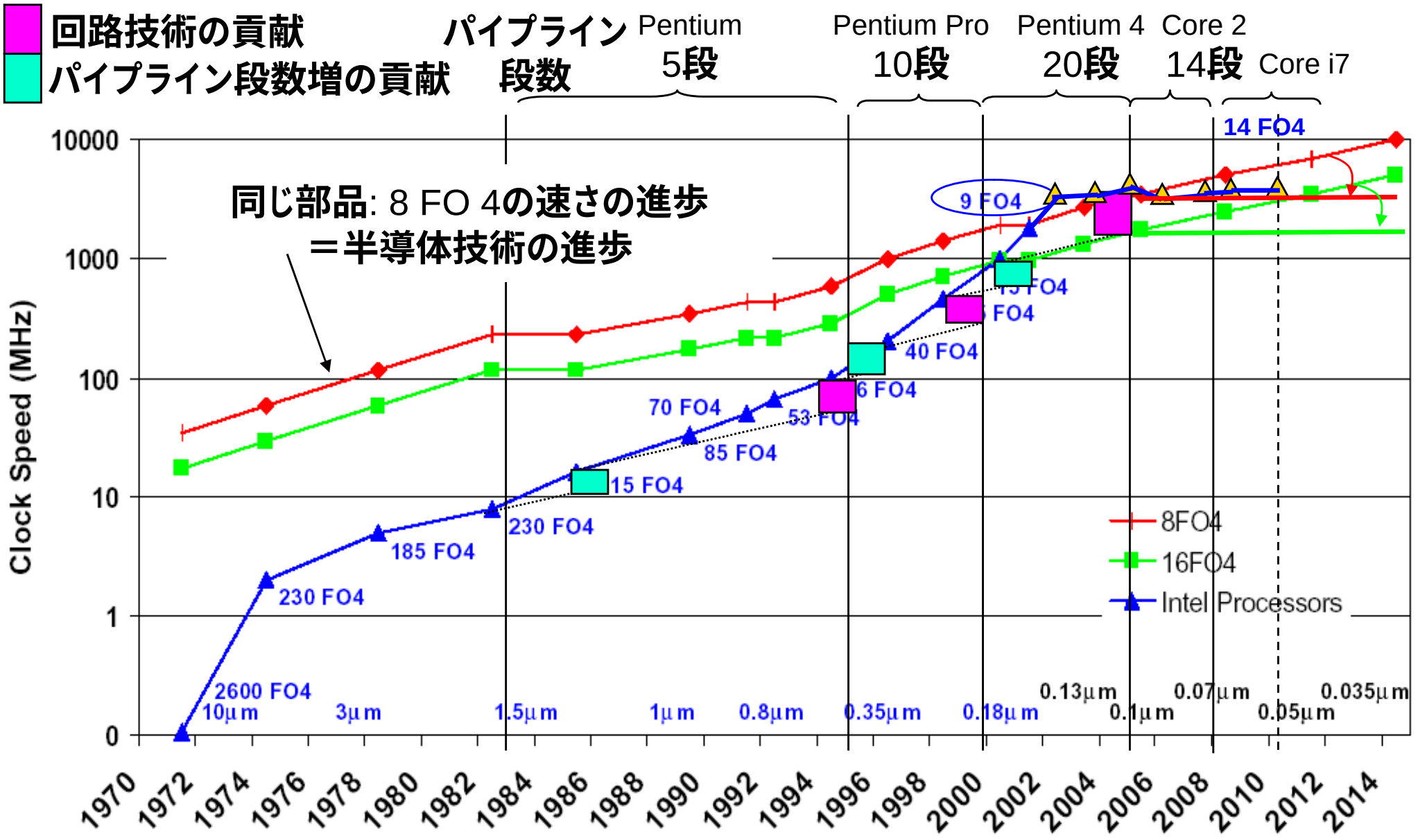


Fig. 6.　Typical sense amp and CDL pair.

パソコンCPUの速度向上のグラフ　(再掲)



Slide by Steve Keckler

Pentium4の速度向上を支えるために出てきた新技術の数々

どれも一発技術なので、つぎつぎに新しい技術を投入してきた

ハイパーパイプラインを支えるさまざまな技術

- 命令デコードと実行パイプライン本体の分離: NetBurstアーキテクチャ
- 高精度な分岐予測
- 分岐を超えた命令供給技術：Trace Cache

回路技術 – どんどん電力を食わせる諸刃の刃でもあった

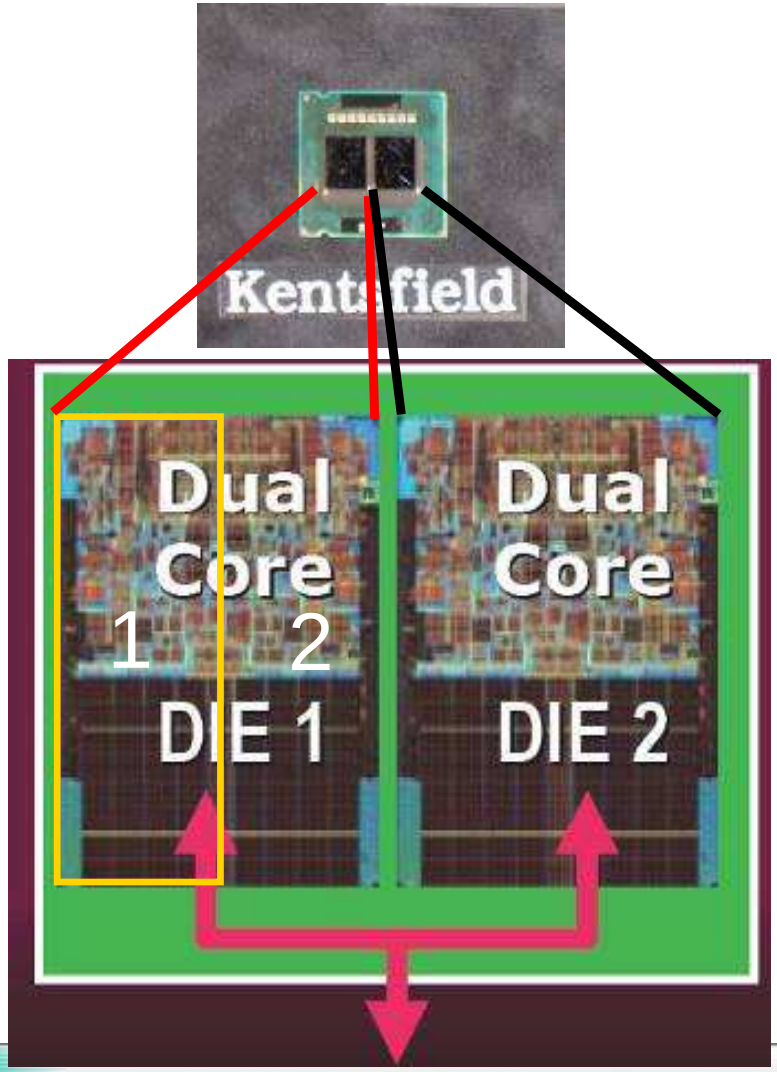
- Domino ロジック: ドミノ倒しのようにぺたぺた信号が倒れていく方式
- DCN回路技術　(前述の足し算回路)
- 超高精度なクロック分配系

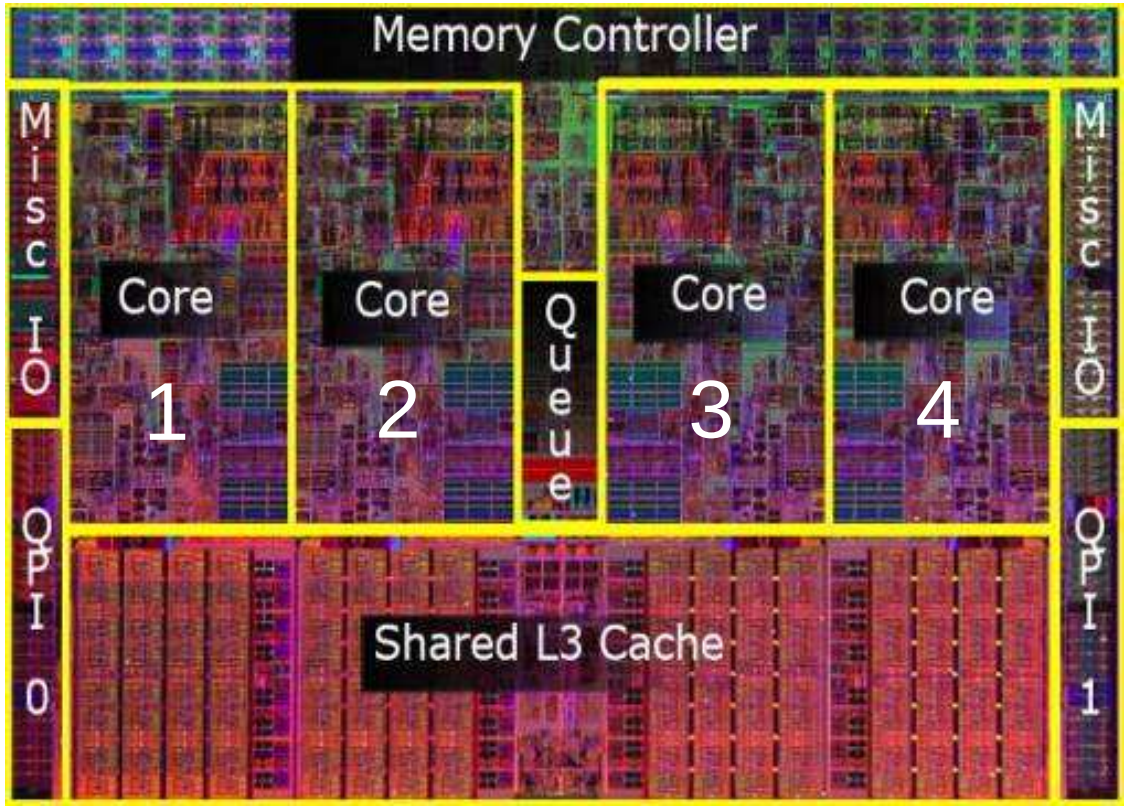
2002年以降、半導体技術の進歩分の性能向上すら得られてない

- 電力をバンバン食わせるなら性能は向上するが、現実的な電力を狙うと半導体技術の進歩での性能向上は得られなくなっている
- 発熱限界を超えないように速度リミッター(Speed Step2)を導入している
  - チップ表面温度が90度を超えるとスピードダウンし、110度を超えるとシステムがシャットダウンする。

速度向上の曲がり角を乗り越える：マルチコア技術(1)

- **複数のCPU（コア）を1個のチップにいれる**
  - PlayStation3のCellも、ARM MPCoreも、AMDも、Intelも、その他もろもろのプロセッサ（Cavium, RMI, PowerQUICC, SH, Sparc）も、最近は皆この手法
  - 版で押して複製するので設計は簡単だが、実は落とし穴が。。





ブルームフィールドのチップ写真 4コア/die

速度向上の曲がり角を乗り越える：マルチコア技術(2)

- Intel Core 2 Extreamシリーズ:

リリース時期	コード名	コア速度 (GHz)	コア数	2次キャッシュサイズ	プロセスルール	消費電力 (TDP)
2006/7月	コンローXE	2.9	2	4MB	65nm	75W
2006/11月	ケンツフィールドXE	3.0	4	8MB	45nm	130W
2007/11月	ヨークフィールドXE	3.2	4	12MB	45nm	150W

- Intel Core i7シリーズ：機能拡充: x2 マルチスレッド、L3キャッシュ内蔵、メモリコントローラ内蔵、チップ間接続QPI

2008/11月	ブルームフィールド	3.2	4	256KBx4 L3: 8MB	45nm	130W
2009/6月	ブルームフィールド	3.3	4	256KBx4 L3: 8MB	45nm	130W
2010/3月	ガルフタウン	3.3	6	256KBx6 L3: 12MB	32nm	130W

速度向上の曲がり角を乗り越える：マルチコア技術(3)

完全に無関係なことなら、同時に別々にやらせられる

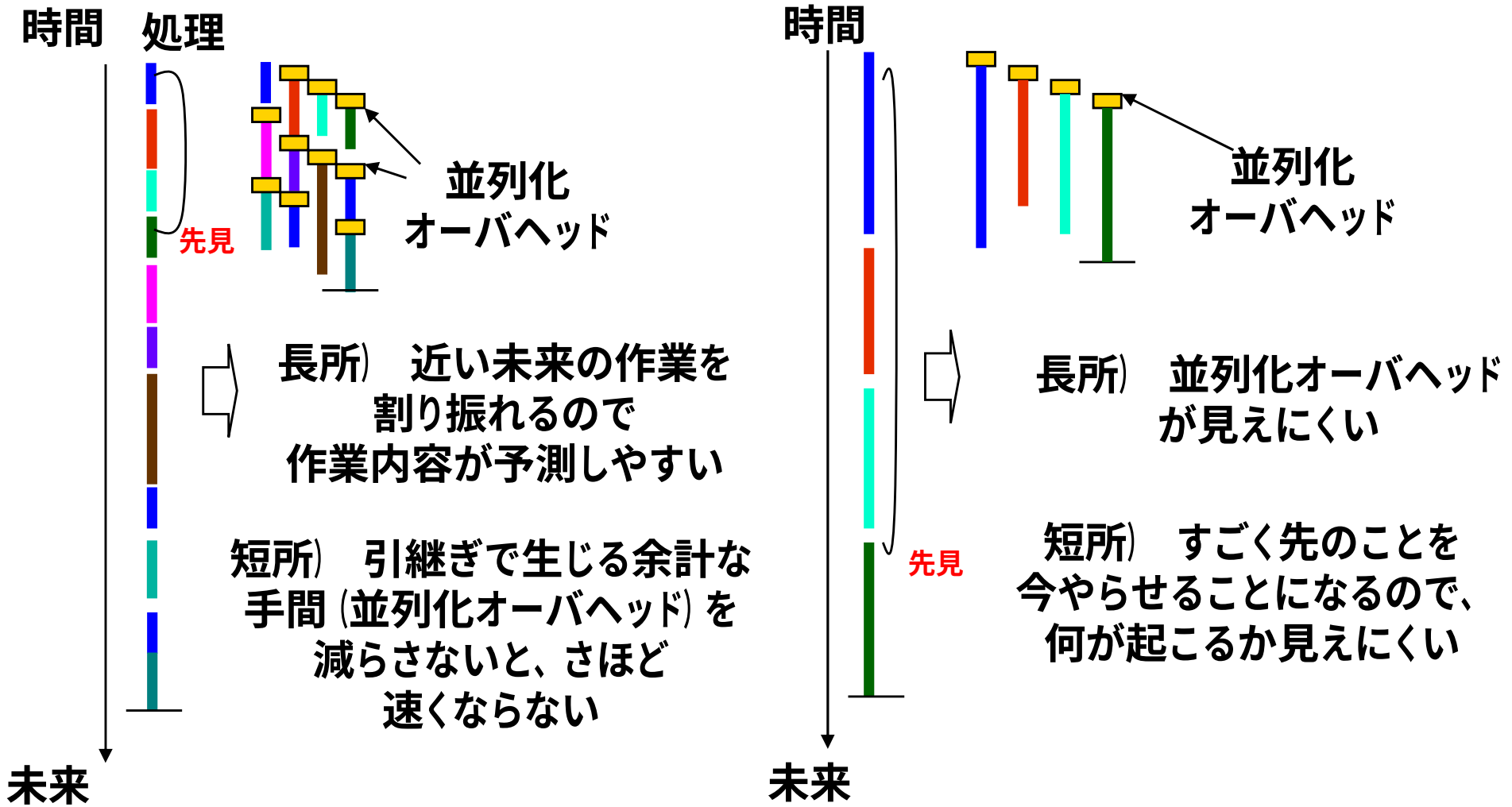
- 映画を見ながらワープロを使うとか、でも一人が相手ではせいぜい2～3個同時が限度
- インターネットの向こう側では、同時にたくさんの人を相手にweb pageを提供したりするので、ここでは使える

一個のプログラムに対しても並列処理で性能が出せるという期待

- スレッドレベル（中粒度）並列処理という
- ただし、人手でプログラムを書き換えるのは大変
- そもそも並列に思考すれば良いというのが、1985年ごろの第5世代コンピュータ国家プロジェクトの発想 ⇒ 大失敗
  - 人間は、そもそも順番に物事を考えており、並列には思考できない！ （私見）
    - 結局、役に立つことを書こうとすると、並列には書けなかった。
  - LSIは並列に動いている。これを人間が設計できているのは、以下の理由（私見）
    - ソフトウェアに比べて、桁違いに設計規模（状態数）が小さい
      - » ハード 10万FF：2^(10万)状態，ソフト 10MBデータ： 2^(8000万)状態
    - クロックという単位を導入して並列に動作する範囲を、さらに制限している
    - 基本部品の接続という概念で画像化して、処理の流れを可視化できている
    - 複雑な処理は、ステートマシンというものを使って、結局、逐次化して書いている。もしくは、難しいことは、全部CPU上のソフトウェアに追い出している
    - さまざまな支援ツールが用意され、設計･検証方法論が確立している。それでも検証工数の爆発が大問題になっている。

並列処理は、仕事の割り振りと一緒

- 一人でやると未来にやらなければならないことを今他人にやらせること
- 大勢に仕事を振るためには、先を読まないとならない
- 順番にやらないとできないこと（逐次処理部）=どうしても他人に任せられないこと=の存在

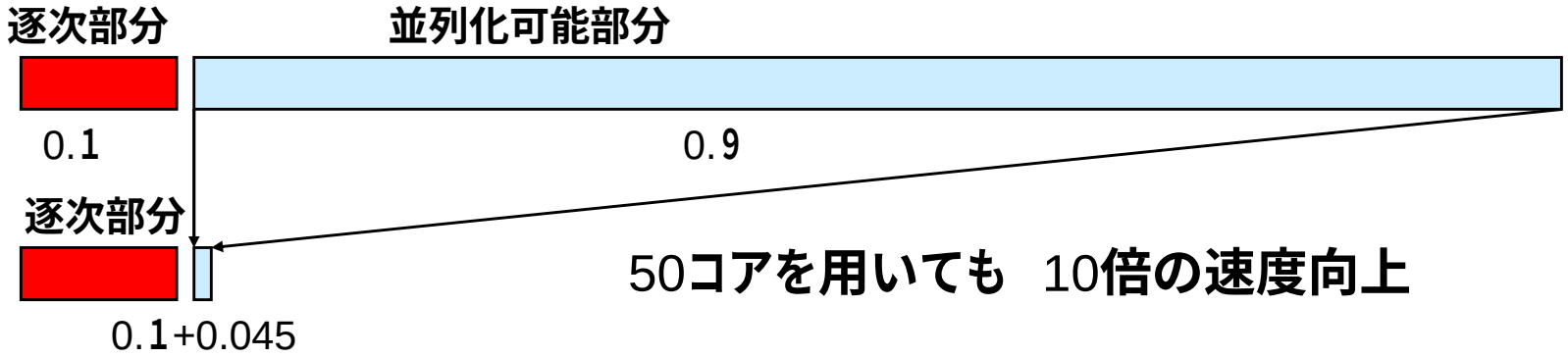


細かく切る（細粒度並列）：これまでの手法

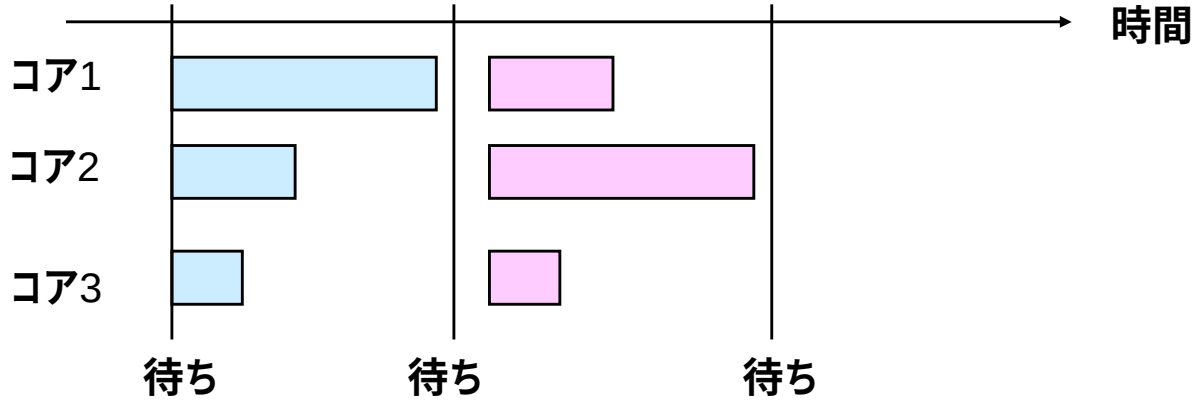
大きく切る（粗粒度並列）：マルチコアでの手法

並列処理の宿命

- Amdahl's Law: **一人でしかやれない部分（逐次部分）があると何百人つぎ込んでも逐次部分が性能を支配する**



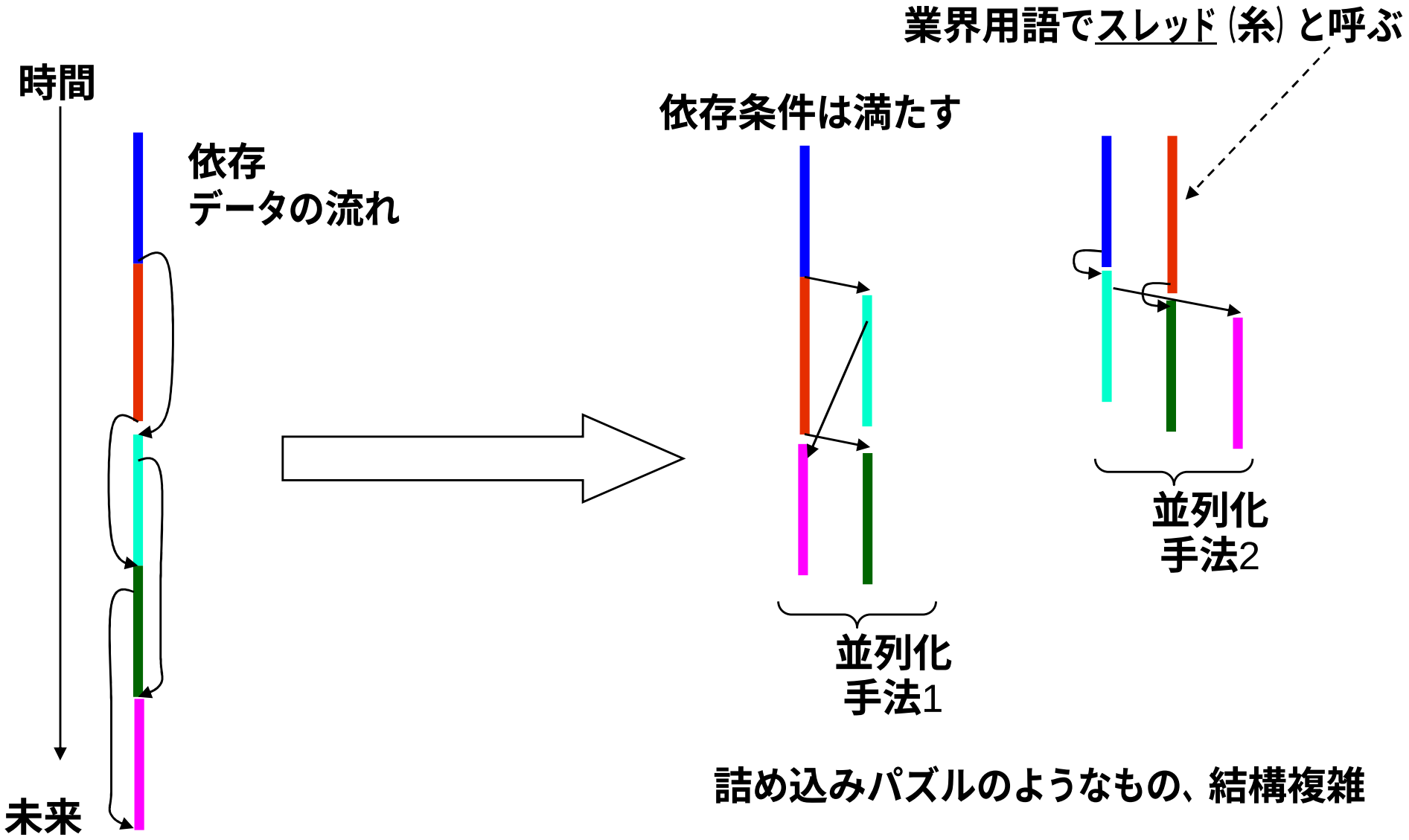
- **負荷ばらつき：　待ち（同期）が必要な場合、一番仕事のおそい奴が性能を支配する**



- **並列化オーバヘッド：仕事を分割するための損失　（通信時間、同期時間）**

並列化は難しい

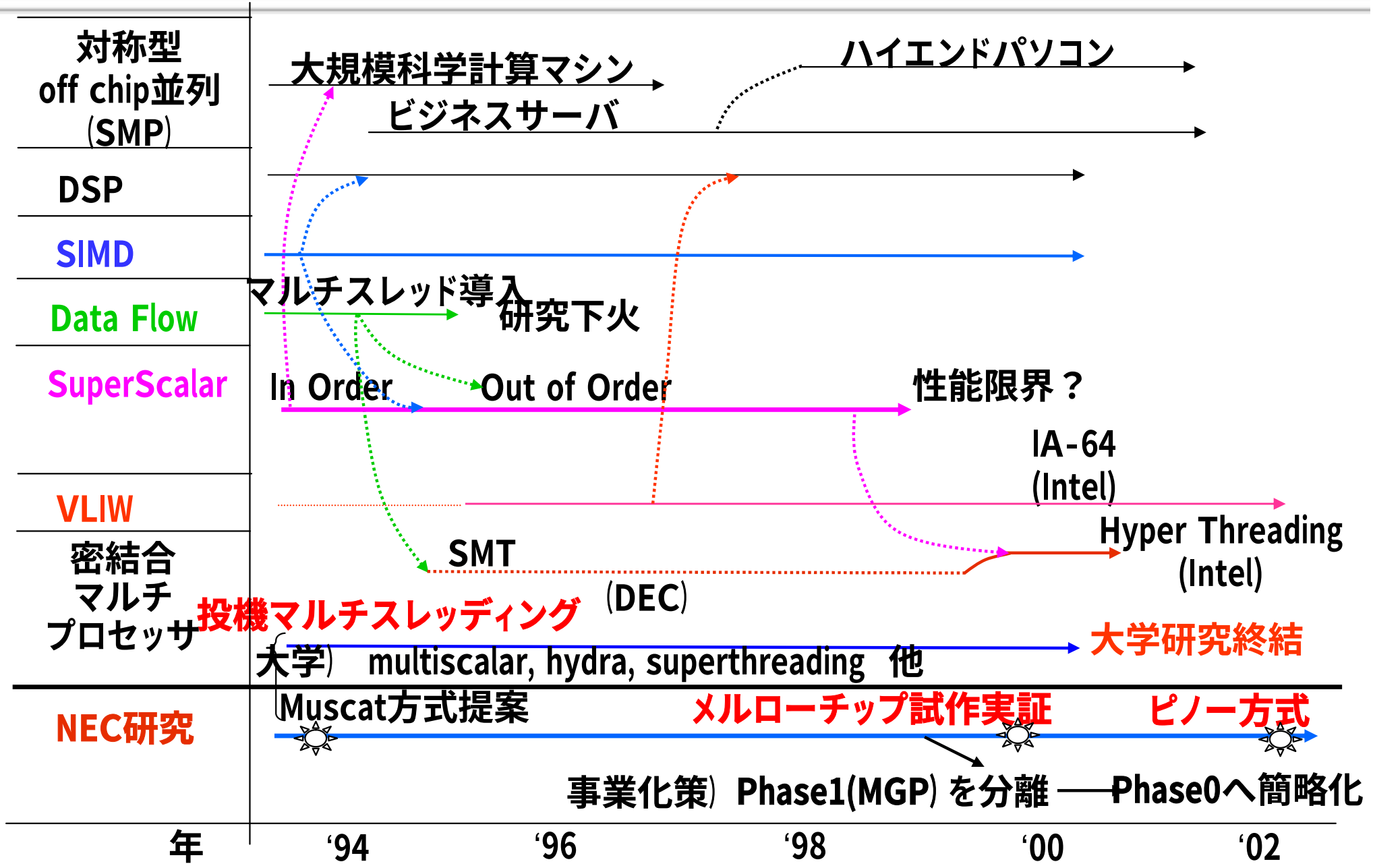
最良の並列化法はよく考えないと気づかない



詰め込みパズルのようなもの、結構複雑

各色の中は、順番にやらざるを得ないとする

並列処理研究と事業化の推移 (2002年版)

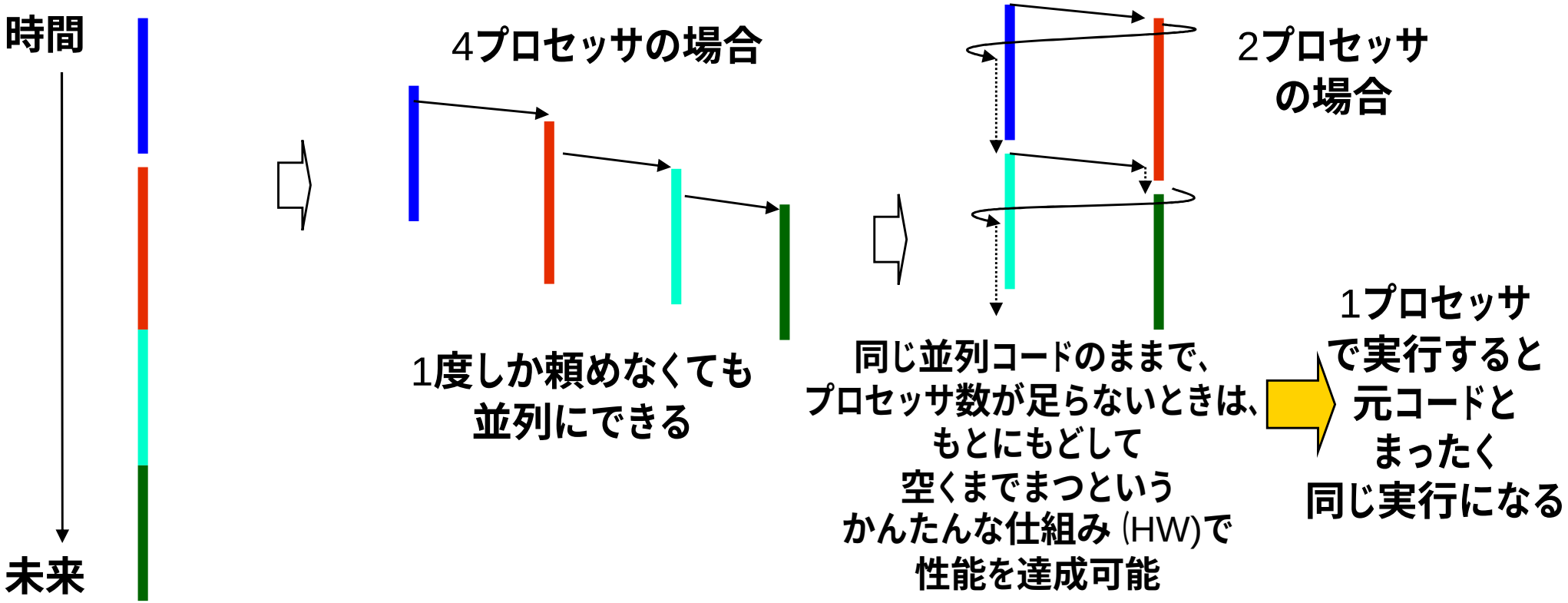


課題を解決するための最新技術

1. 並列化支援ツール
   - ツールがあっても先を読むのは人間
2. 自動並列化コンパイラ： 早稲田大学　笠原研、Intel
   - 笠原研: コンパイラという変換ツールが先を読めるように、プログラムの書き方を制約
   - Intel: 完全自動ではごく単純なことしかできない。性能を出すために人間の指示(OpenMP)を要求
     - 静的解析のみ: 信用に足らない仮実行情報（トレース）は用いない。ギャンブルはしないいさぎよさ。---- しかし、しょせん性能に限界あり！
3. 完全自動並列化
   - 投機マルチスレッディングという世界的な研究動向：1992年～2000年
     - ギャンブル（投機実行）：（だめならやり直し）という機構をハードウェアに入れることにより、従来の難しさを緩和する
     - ハードウェアの支援により、仕事引渡・受け取りのオーバヘッドを削減する
     - トレースドリブン最適化： とりあえず適当なデータで何度か実行して、プログラムの動きを見て考える。実際のデータと試したデータと違うので、どうしても本実行は、ギャンブルになる。
     - ギャンブルが外れたらやり直す
   - NECの技術: メルロー、ピノーも上記の流れ

NECの提案

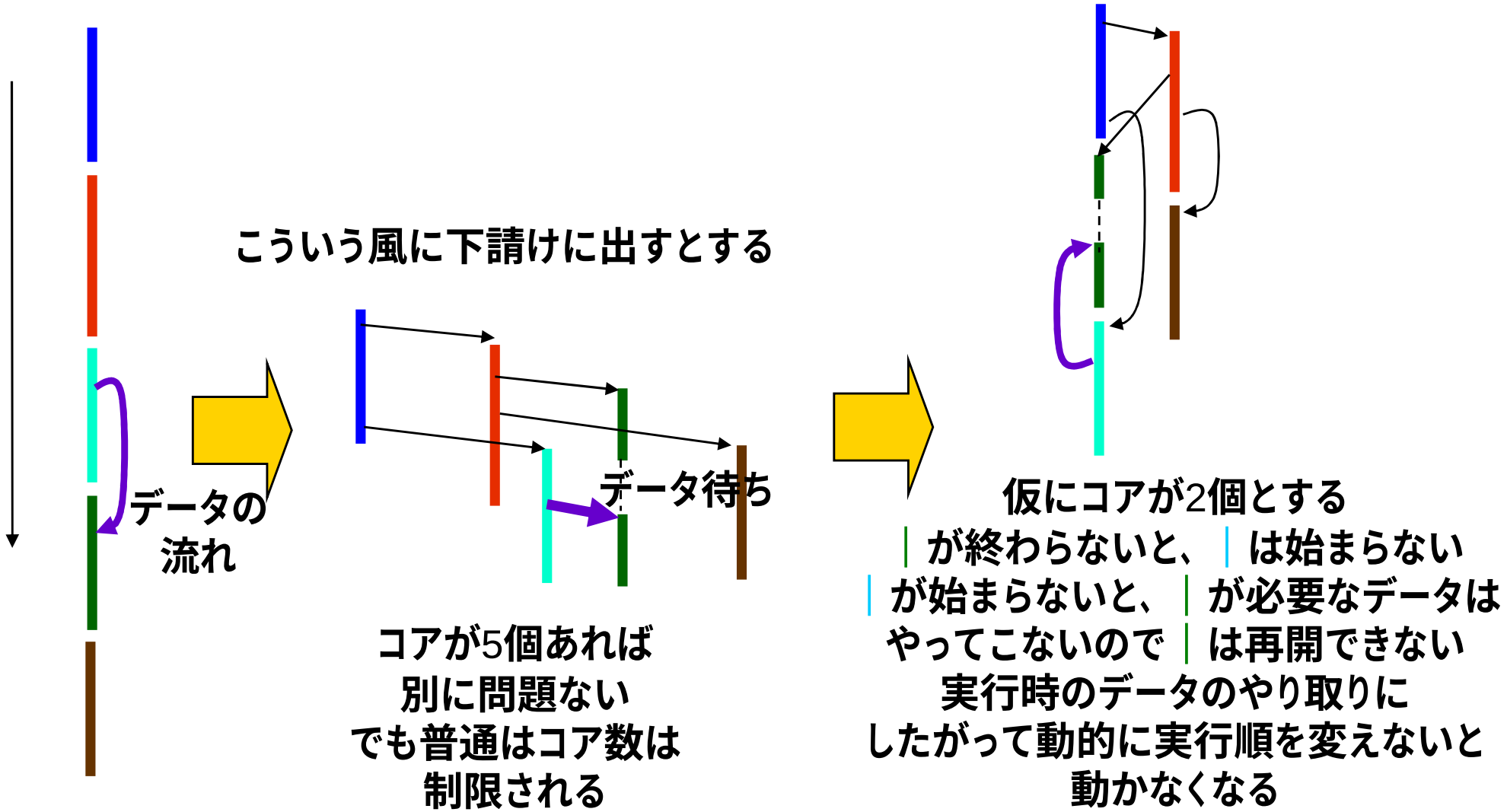
- **アイディアの根本：　1回しか下請けに出せれない　(下図)**
  - 2000年にチップを試作し国際学会発表した初期版**メルロー**の基本アイディア
    - ハードウェアを簡単化、それでも引き出せる性能はほとんどかわらない
    - 下請けに出す時点 (Fork命令の挿入位置) はソフトウェア (コンパイラ) で解析して事前に決める
- **ハードウェアにより動的に一回下請けを保証**
    - これが次世代版**ピノー**での基本アイディア
    - バイナリーコード変換という手法でプログラムを変換。**メルロー**よりだいぶお手軽
    - 実行前に保障するのはきわめて困難　(分割コンパイル、関数の入れ子読み出し等)



下請け一回: メルローのアイディア

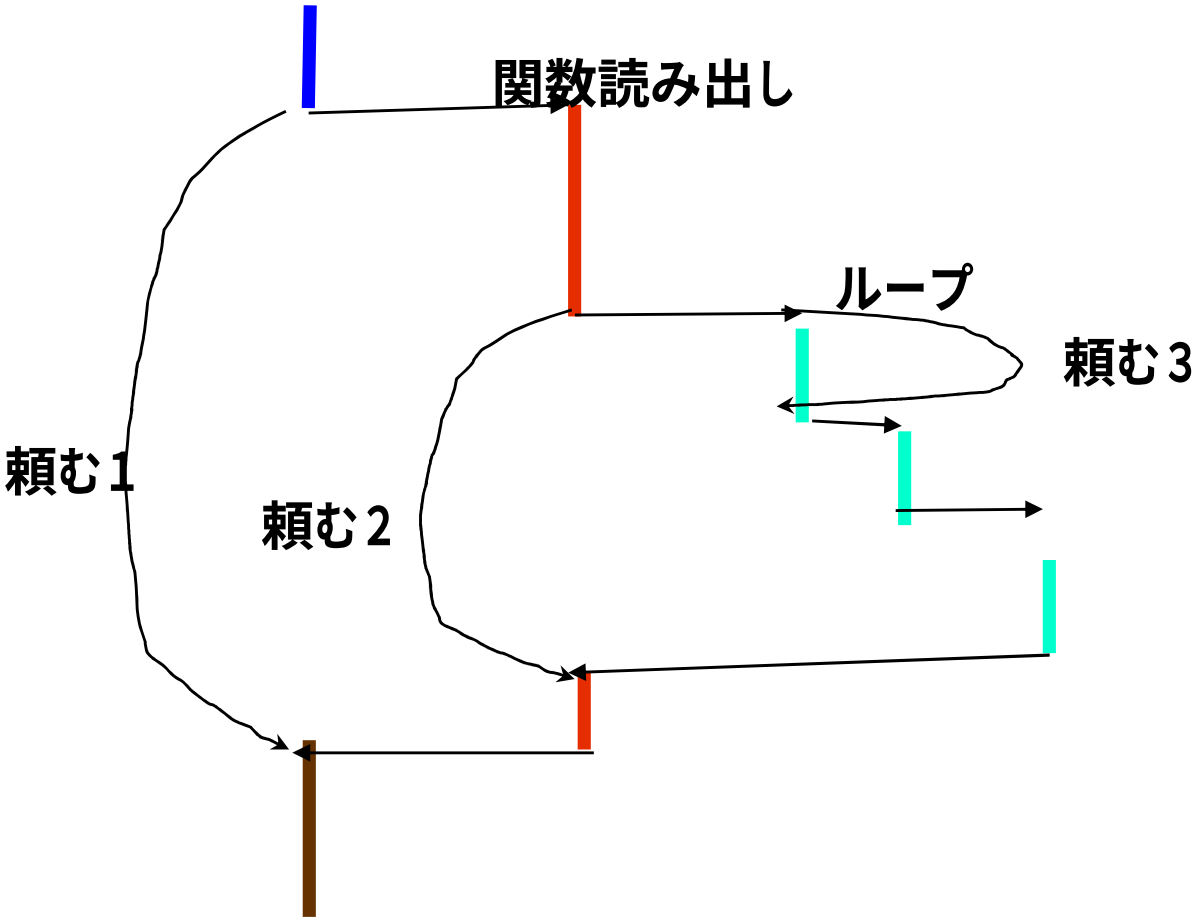
動的スケジューラという複雑なハードウェアが必要になる

- 一度しか下請けに出せなければ、スレッドの順序関係が一意に定まるので、古いものを先に実行すれば、必要なデータは必ず先に生成され問題は起きない。



実行時下請け一回保証: ピノーのアイディア

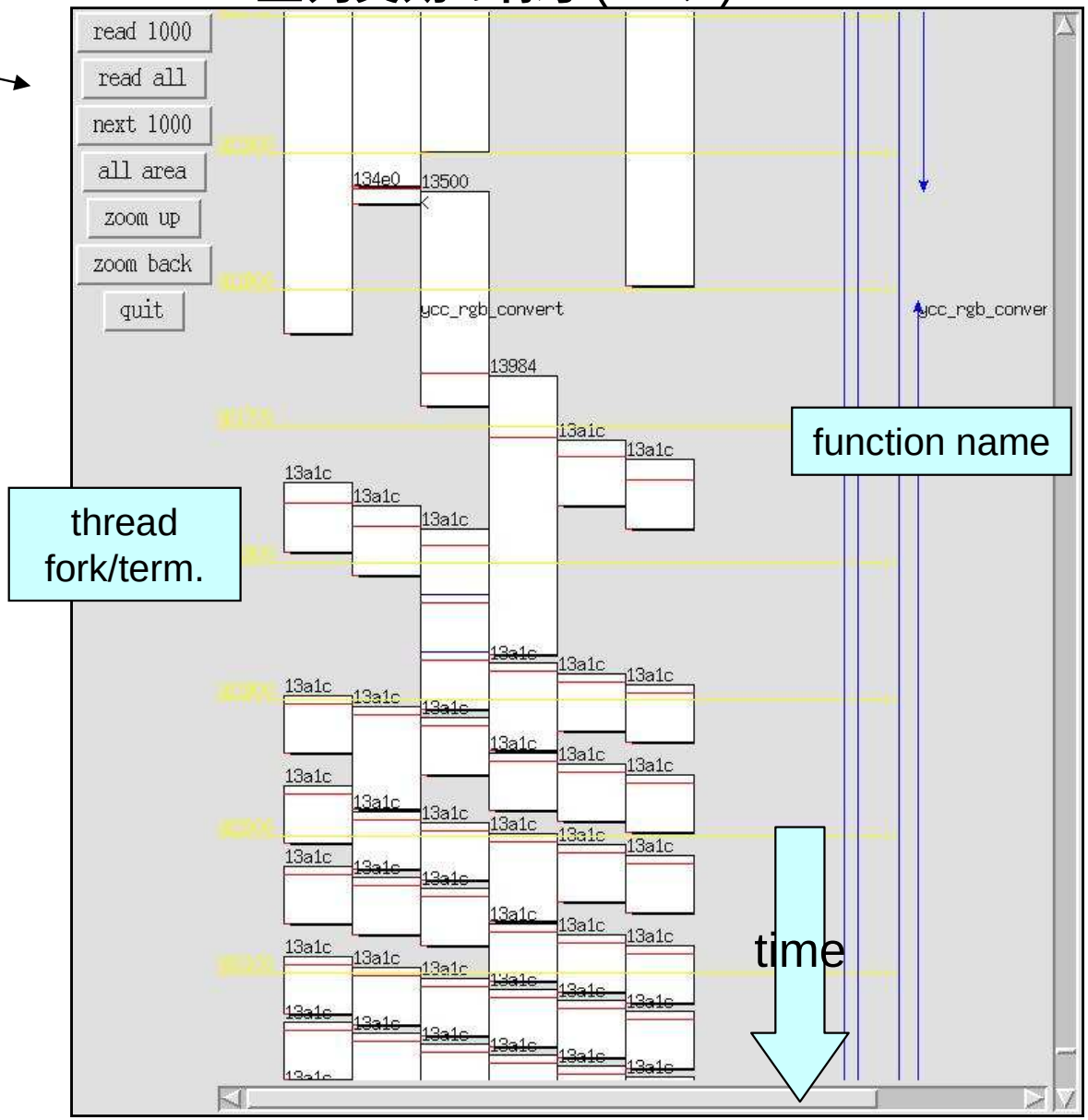
- どこまで自分でやるかあらかじめ決めにくい
- だから実行時に、最後のやつにきめて、先の依頼を自動的にキャンセルする
- ギャンブル機構をキャンセルに用いる



実際どんな感じに動くか

- Thread Activity Visualizer
  - dynamic linked to amrsd
- For Tool Debugging or Hand Assembling.
  - CFP Inst. Level Sim detects tool bugs.
  - CFP lint is Feasible (future work?)
- Applicable to RealTrace, EmbTrace -> Need Detailed Info.

並列実効の様子(6コア) on JPEG



その他のピノーの新アイディア

細かい仕事の切り分け（細粒度）でも、大雑把なきりわけ（粗粒度）でも、どちらでも効率よく実行できるハードウエアにした

- ギャンブル実行・仕事受け渡し・結果取り消し/やり直しの仕組みを軽量化 – 細粒度に対応
- 並列化を指示するために追加する命令量は元のコードの1.4%と小さく押さえ込んだ
- かつ非常に将来にわたってたくさんのギャンブルをうてれるようにした。（大量の資金とギャンブル結果のチェック機構を効率的に実装） - 粗粒度に対応

仕事の切り分け、割り振り方の組み合わせは天文学的な数になる

- ごく簡単なプログラムでも可能な組み合わせは、2^40 = 1兆を超える。複雑なものでは、2^1000 = (1兆)^25 を超えるだろう
- 解析的手法（WIS）や経験的手法（ヒューリスティックス）を組み合わせ、数年に渡りチューニングした結果、数分という短時間で、人知を超える準最適な解を得る手法を確立
- 高性能向けに最適化しても、ギャンブルが外れる確率は性能重視で5-15%程度
- ギャンブル度合いと性能は天秤。用途に合わせて調整可能

その他のピノーでの新アイディア – 続き

粗い情報で変換が可能

- いくつかのサンプルデータを使って、命令シミュレータというもので実行した大雑把なデータを用いて高精度な並列化が可能

完全自動並列化なので、もとのプログラムが正しく動いていれば間違いが入る余地はない

オブジェクトコードから並列オブジェクトコードへの完全自動並列化

- プログラムの元データ(ソース) が不要

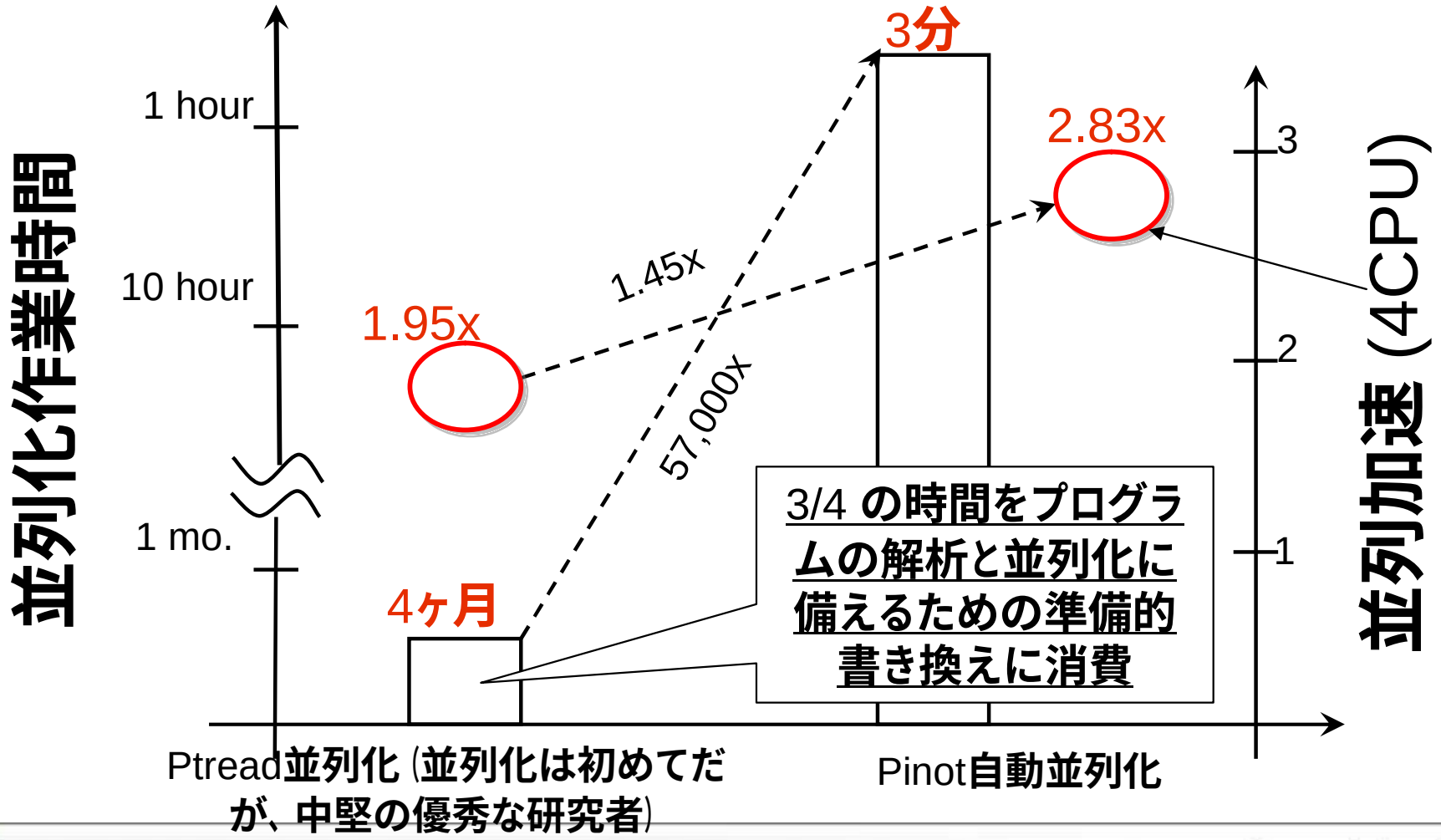
世界的にみて、いまだに他の誰も実現できていない

- 優秀な研究員、動作解析・図示化ツール群、多量の計算機パワーを長期にわたって投入し、徹底的にチューニングした成果 – そう簡単にはマネはできない

効果

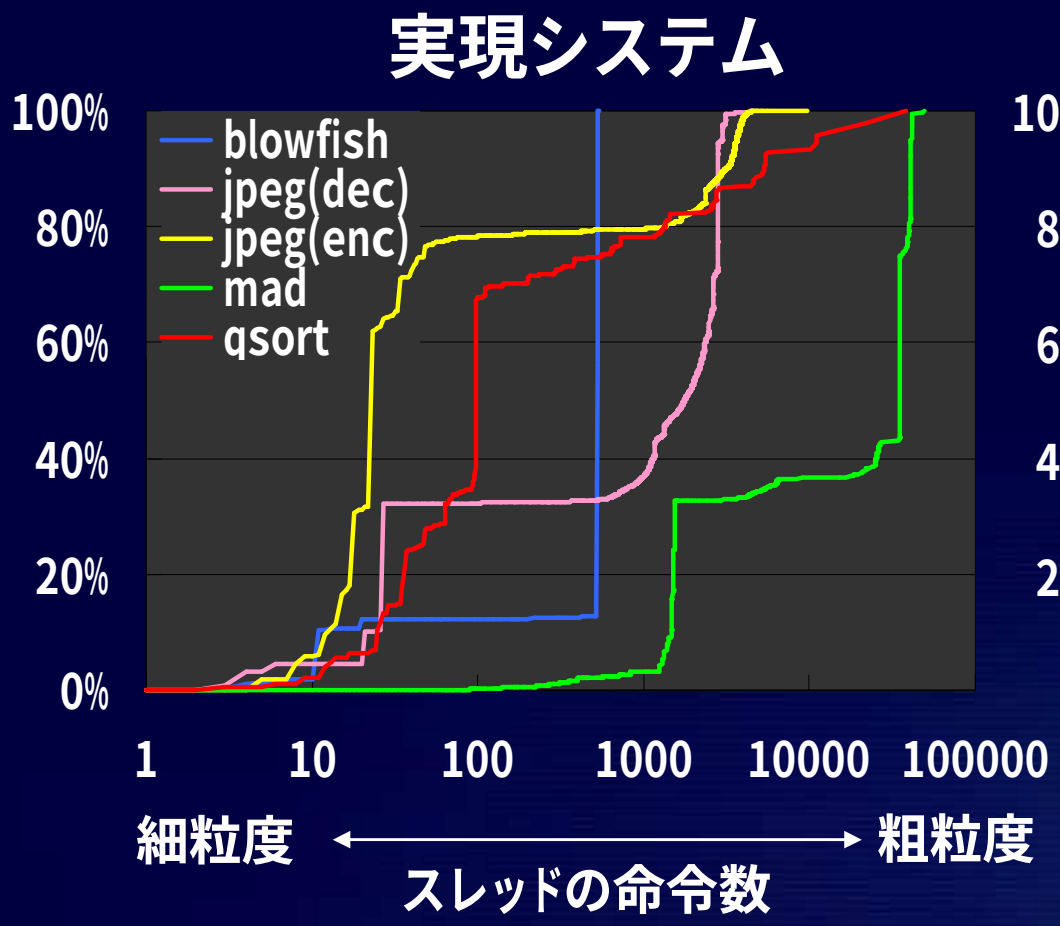
並列化しにくいプログラムにて実験

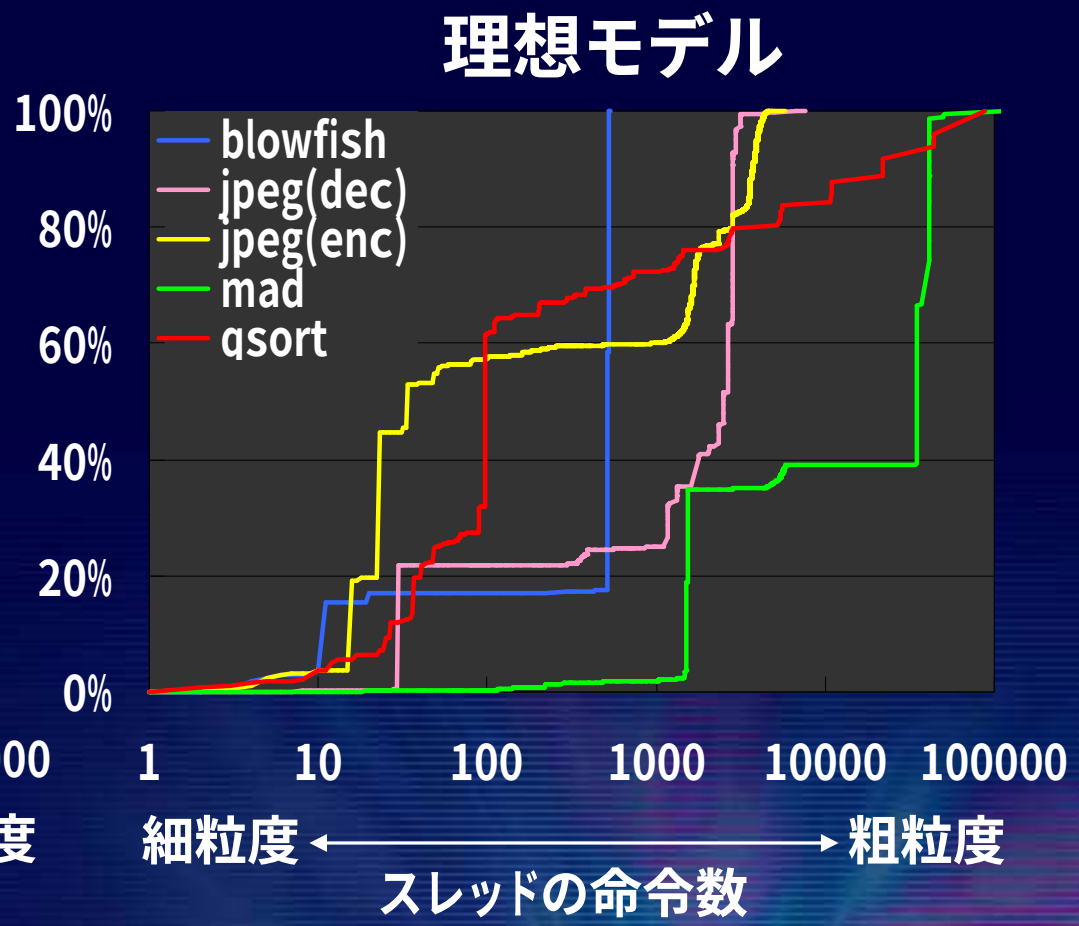
- 音楽圧縮プログラム（AACエンコーダ）での比較
- 人間は、Stereoの右と左の並列性しか利用できず。それでも最大値2倍のうち1.95倍まで出せたのは優秀
- 2.83倍は人知を超えた手法ならでは



実現できる並列化粒度の解析（結果）

理想モデル用に最適並列化し実行したときと、実現したシステム向けに最適並列化実行したときと、各アプリで利用している並列粒度の分布が同じ： 理想条件でなく、現実システムでも最適に並列性が引き出せる





スレッド粒度の分布解析

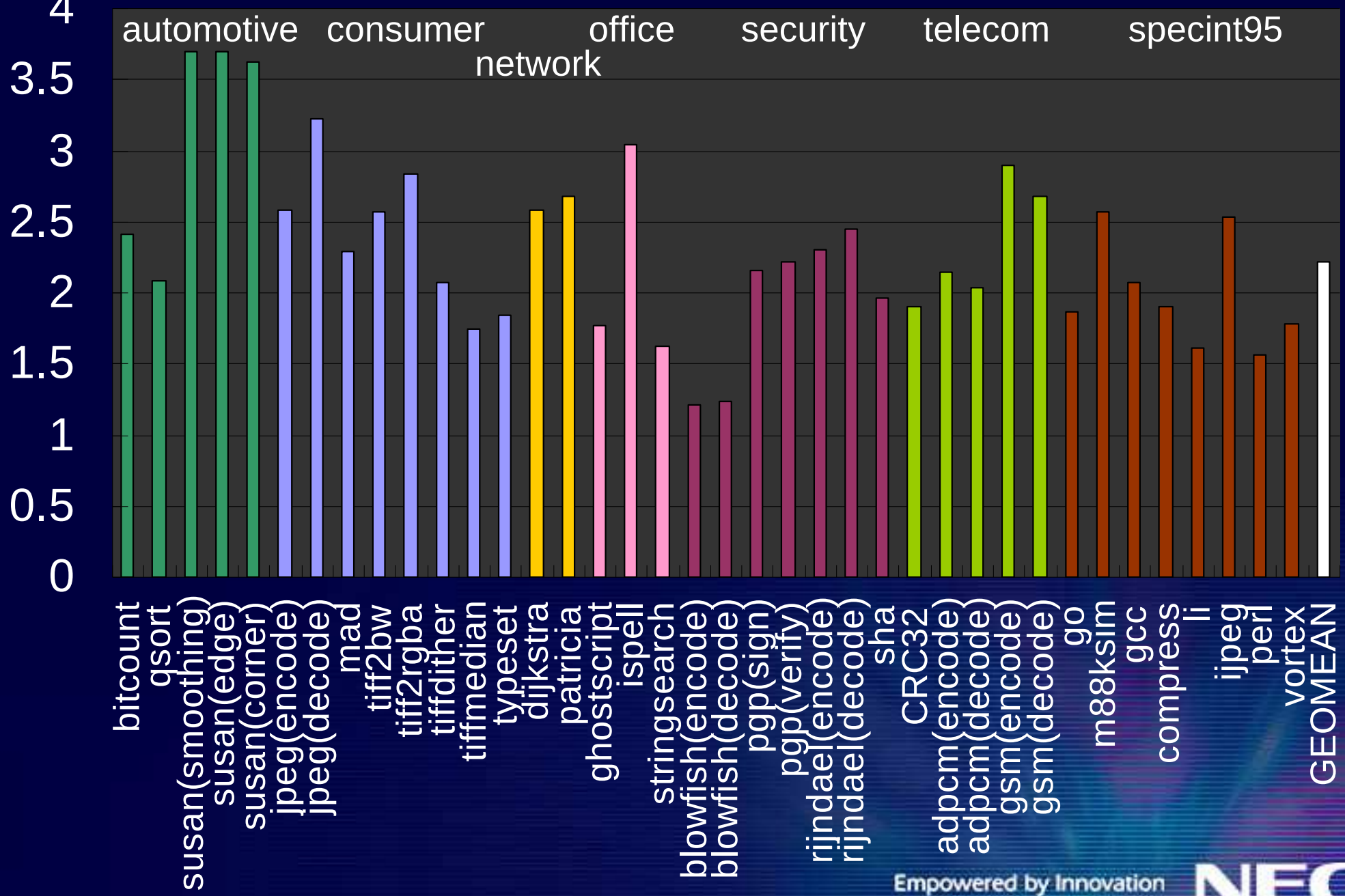
実現できる並列化粒度の解析 (条件)

	実現システム	理想システム
コアのタイプ	32-bit RISC, no FPU, single-issue, 5-stage	
I$, D$(VRC: **投機情報格納部**) **のサイズ**	I/D=16/16KB, 4way, 32B/line, LRU	**無限個**
MM latency, bus width	20-cycle, 32-bit	No
コア数	**4コア**	**16コア**
Thread spawn latency	2-cycle	
Inter-core reg. bus	32-bit = 1reg./cycle	**無限個**
Inter-core VRC bus	64-bit	**無限個**

- Cycle accurate simulator
- MiBench (embedded apps) and SpecInt95
- gcc-3.4.2 –O3 –msoft-float

4コア時の性能向上 – 出るものもあり出ないものもあり

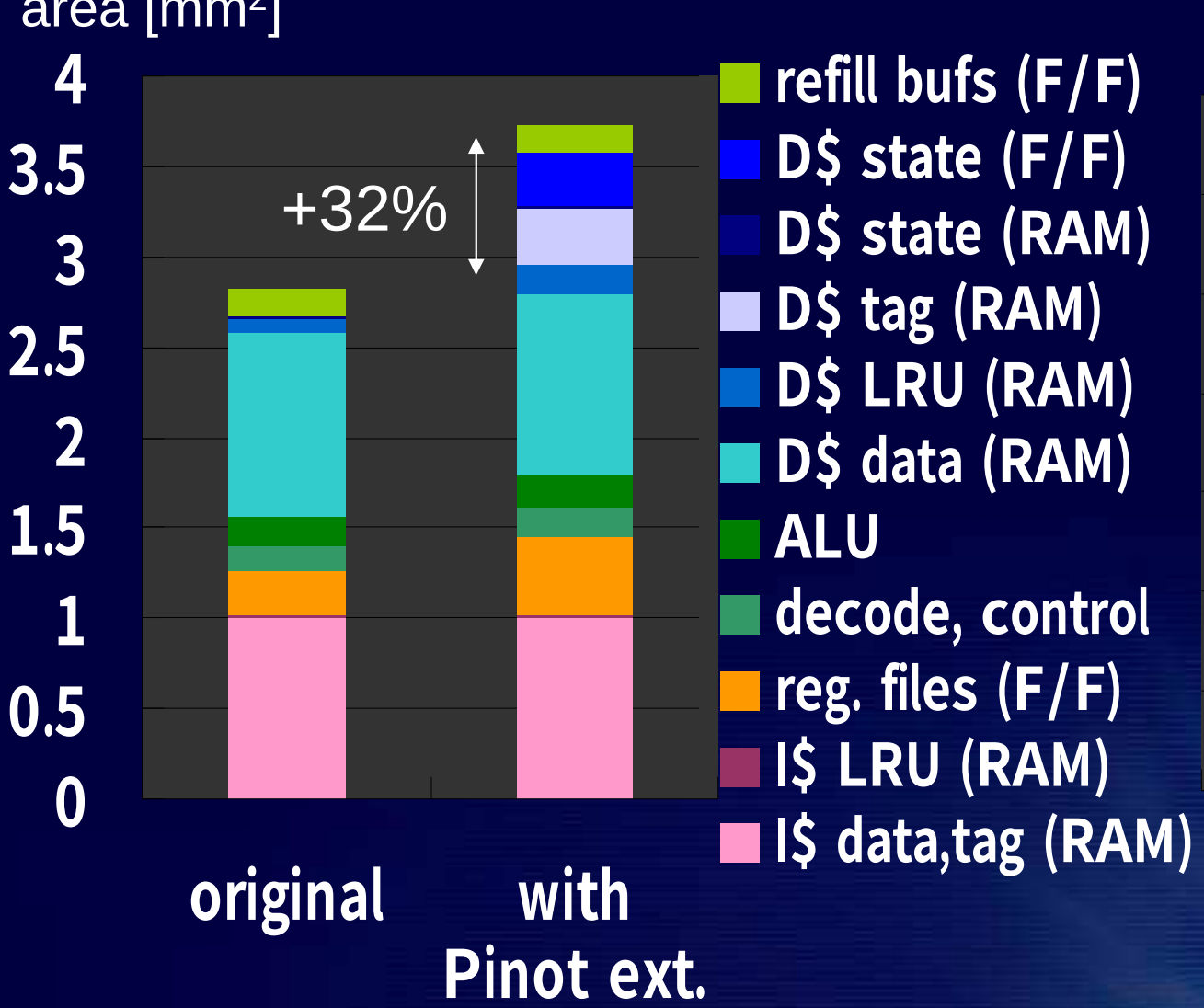
並列加速度　　**従来非常に性能が出しにくいといわれていたものでも2倍近くでている**

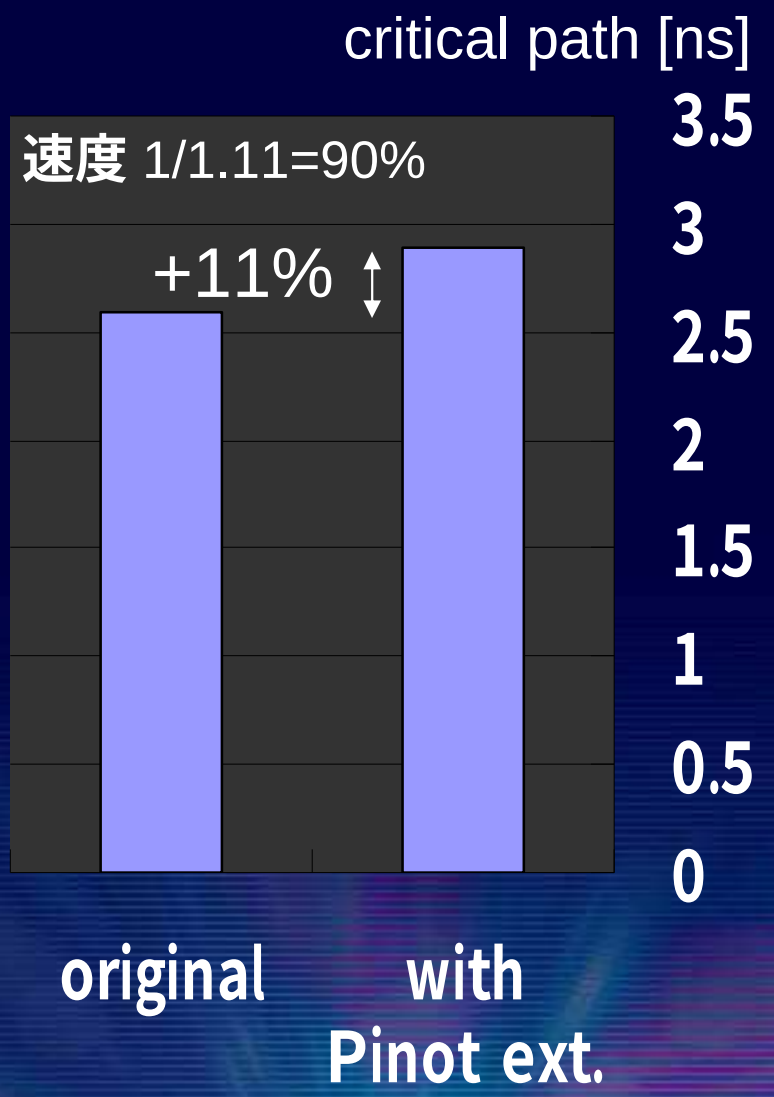


Empowered by Innovation　NEC

弱点: CPUハードウェアの一部改造要、32%の面積増加、10%の速度低下

- 130nm のstandard cell + compiled RAM で論理合成し評価
- マルチコアにするにも面積増加は伴う、その分は加味していない、速度低下についても同様（要検討）



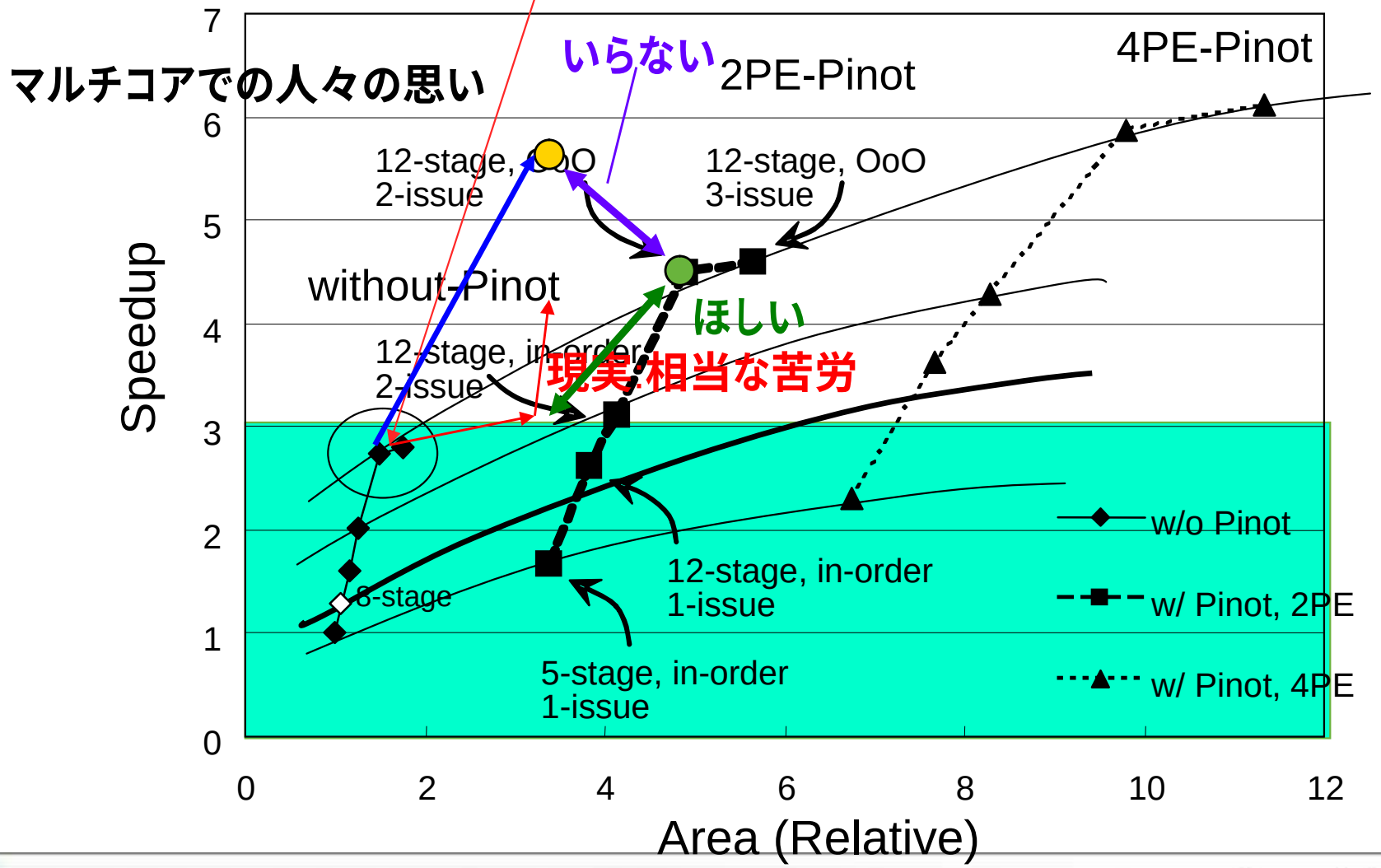


Reg. files and D$ speculative states are dominant

Empowered by Innovation NEC

弱点は克服される – でも、こんな分かりにくい図では。。工夫不足！渇

- 2命令同時発行のスーパースカラ (OoO) の方が効率がよい。しかしその先打ち手はない
- 半導体の微細化で若干の面積増はだんだん気にならなくなる
- 速度低下については、OoO化後だと条件が変わる (要再評価)
- 人手による並列化の難しさが認知され、自動並列化の価値が見直される



ピノー状況

- 2005年に、最難関の国際学会 IEEE Micro38で発表
  毎年日本からの聴講はあるが、当時3年ぶりのアジアからの発表だった。。
- 実証Hardware (FPGA) を試作
- 事業化のため、2005年まで、海外のプロセッサの大手と2年以上にわたって共同評価
  - いろいろな事業化アイディアを出し、いいところまでいったのだが。。。。
- 2005年の国際学会発表以後、今でも世界最先端技術
  - 重要特許多数
  - 本技術はいまだに時期尚早と見る人あり
  - 平均性能向上2.2倍ではコア数を使いきれないと誤解している人がいる。使い方の再アピール必要
  - 面積増、遅延増の弱点のインパクトが強すぎた
  - 新しいやり方への心配も多かった
- 国際的な、データセンター (Google, Amazon) 研究の流行のなかで、陰が薄くなっている
  - 当時のメンバーは皆違う仕事に移った
  - 最近、海外のつまらない並列化評価ツール (critical blue)が脚光を浴びた。
    Pinotに比べ相当遅れている
    – が時代はこういうものでも欲しいと
    マルチコアの難しさにようやく気づきつつある

FPGA実証システム

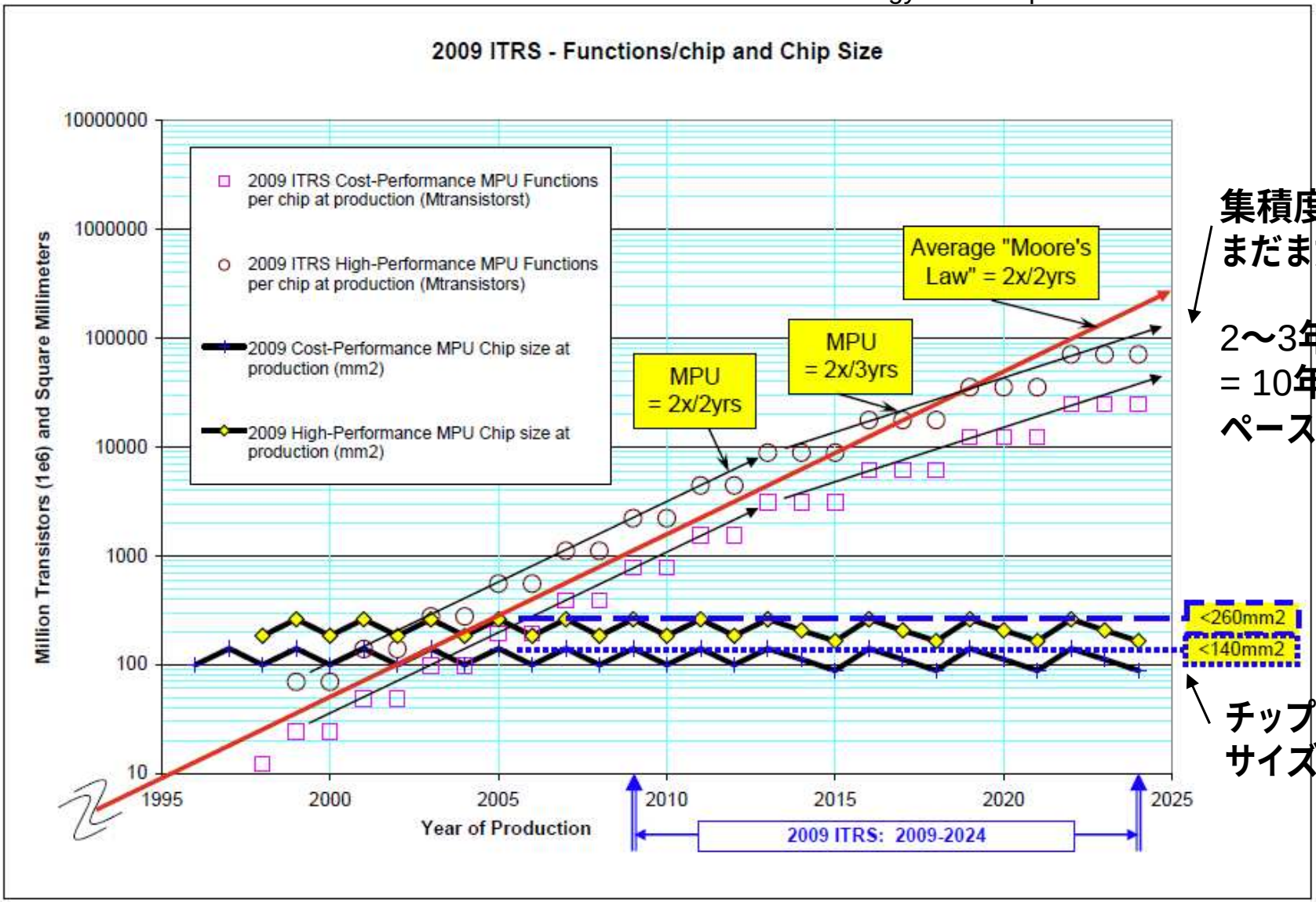
まとめ

- 1980年後半から2000年までのプロセッサの華々しい速度向上
  - 1年で1.55倍、10年で100倍
  - それを支えたさまざまな技術
- 2000年になぜ曲がり角を迎えたのか
  - 消費電力/熱密度問題
- 曲がり角を乗り越える技術
  - マルチコア技術
- NECのマルチコア自動並列化技術: MP98-CFP
  - 1994年ごろからの世界での研究開始とほぼ同時期に研究を開始
  - Merlot : LSIを試作し、2000年に国際学会ISSCC等で発表
  - Pinot: 第2世代版、FPGAで試作、自動並列化ソフトを作成、性能実証 2005年に国際学会で発表
  - 研究所に累々としている死の谷を乗り越えられなかった技術の１つにはさせたくない！

Empowered by Innovation

LSIの集積度予測　(ITRSロードマップ 2009)

ITRS: International Technology Roadmap for Semiconductors



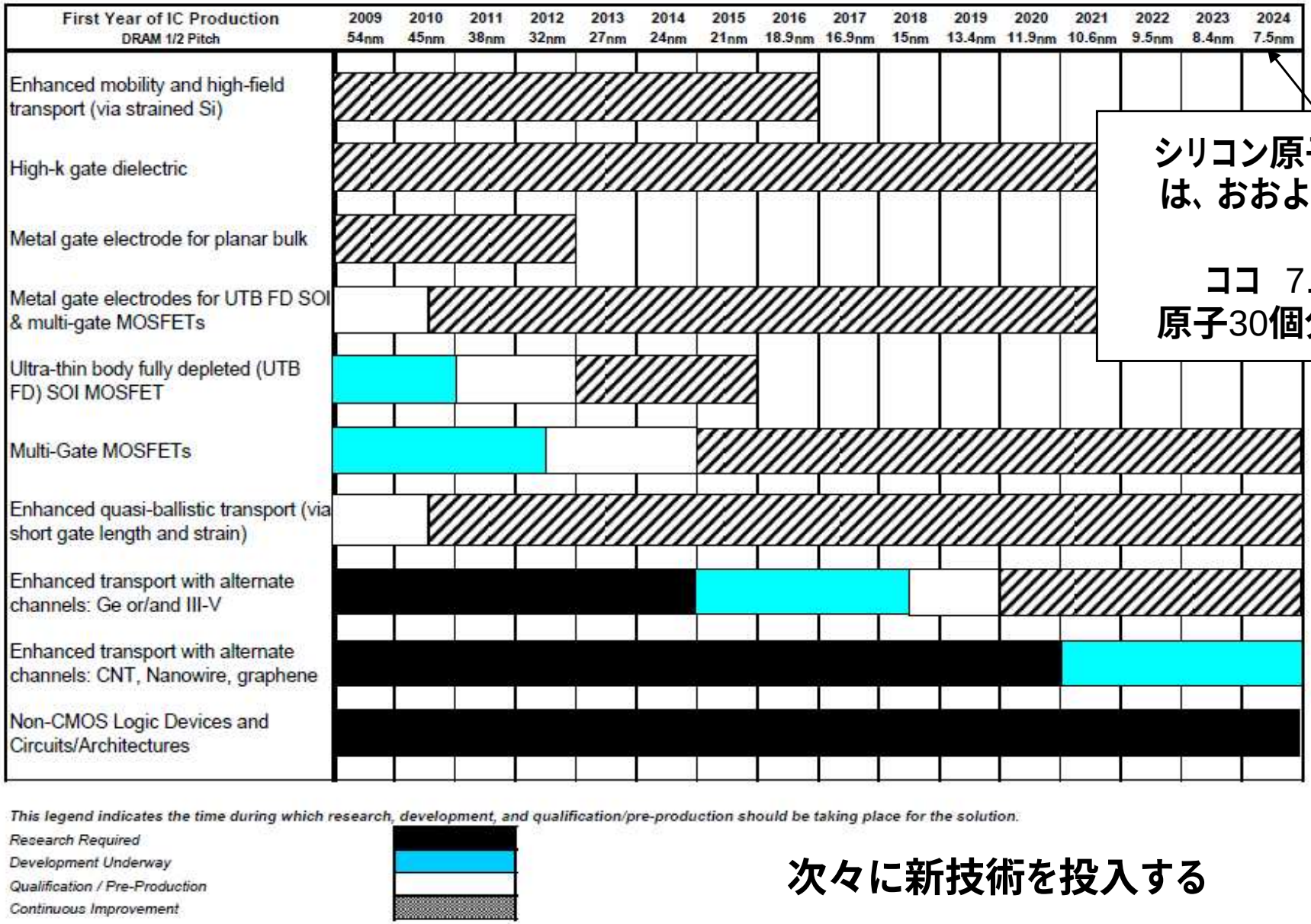
集積度向上は
まだまだ進む

2〜3年で2倍
= 10年で100倍
ペース

チップの
サイズはほぼ一定

Figure 10b　2009 ITRS Product Technology Trends:
MPU Product Functions/Chip and Industry Average "Moore's Law" and Chip Size Trends

LSIを微細化するための技術　(ITRSロードマップ 2009)



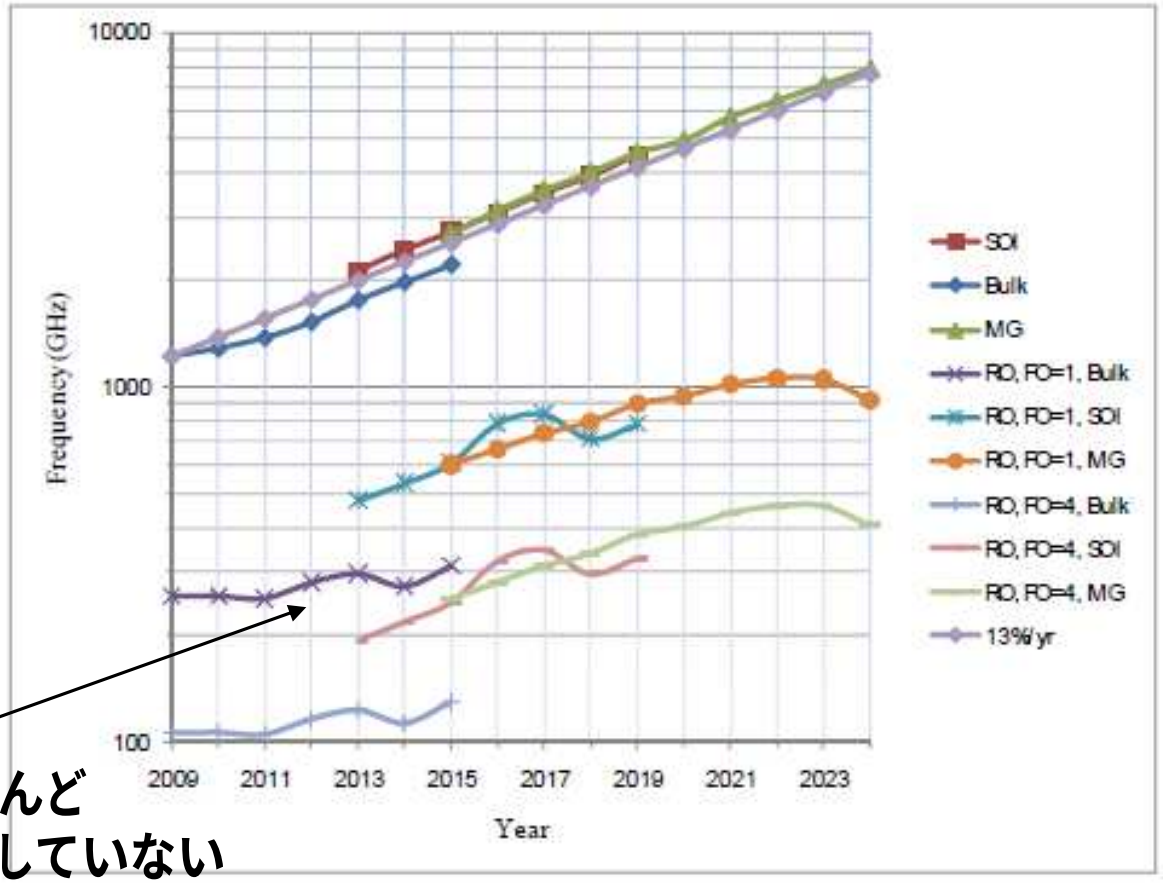
シリコン原子の大きさは、おおよそ0.25nm

ココ　7.5nmは原子30個分のサイズ

次々に新技術を投入する

Figure PIDS2　Logic Potential Solutions

LSIの性能予測　(ITRSロードマップ 2009)



最近は、ほとんど
素子の速度が向上していない

用語

SOI: Silicon
　on Insulator
MG: Metal Gate
RO: Ring
　Oscillator
FO: Fan-Out

Figure PIDS1　Scaling of Transistor Intrinsic Speed of High-Performance Logic

In the legend section, the first set of 3 symbols represent the inverse of CV/I (1/τ). The second set represent inverse of the ring-oscillator delay per stage, for fan-out of 1. The third set represents the same but with fan-out of 4. The last curve is the reference of 13% increase per year.

ゲーム機の性能予測 – 構成予測　(ITRS2009より)

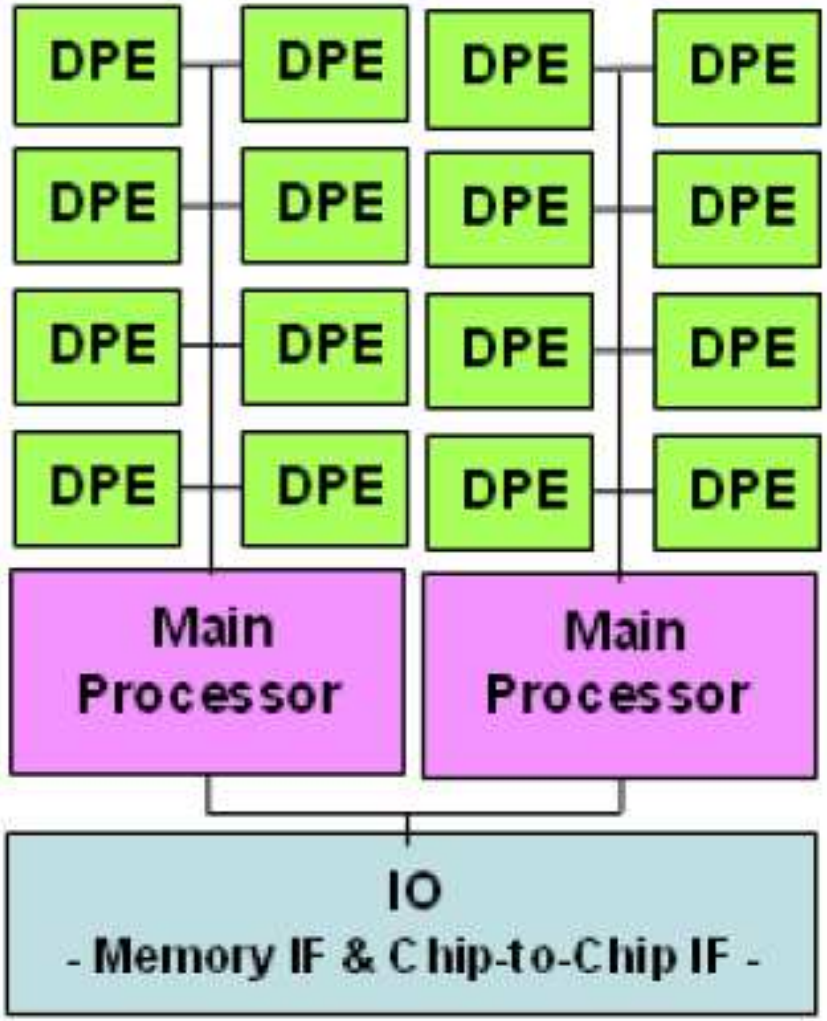


Figure SYSD8　SOC Consumer Stationary Driver Architecture Template

Consumer Stationary Driver : **ゲーム機など**

ゲーム機の性能予測 – 性能予測　(ITRS2009より)

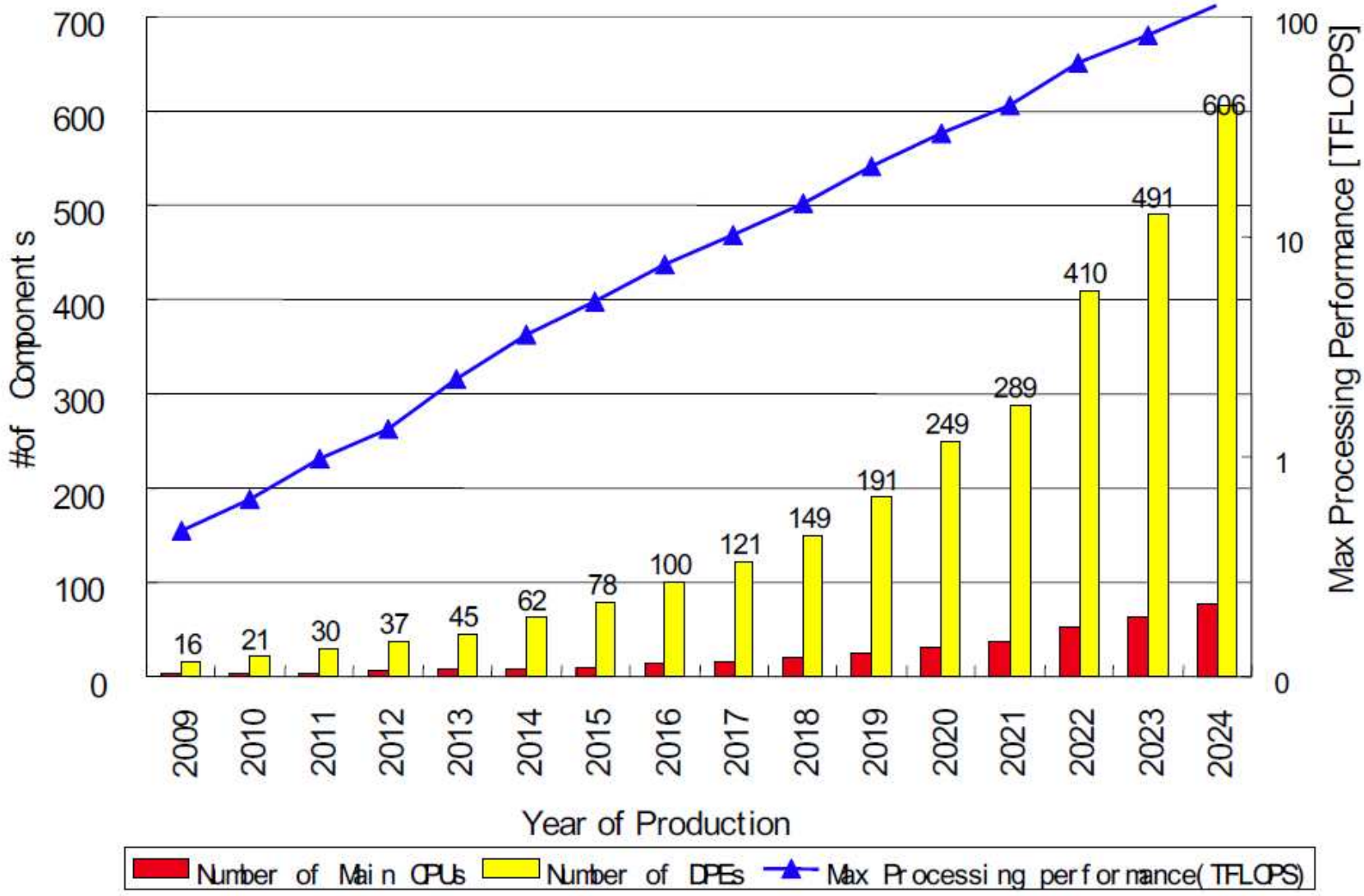
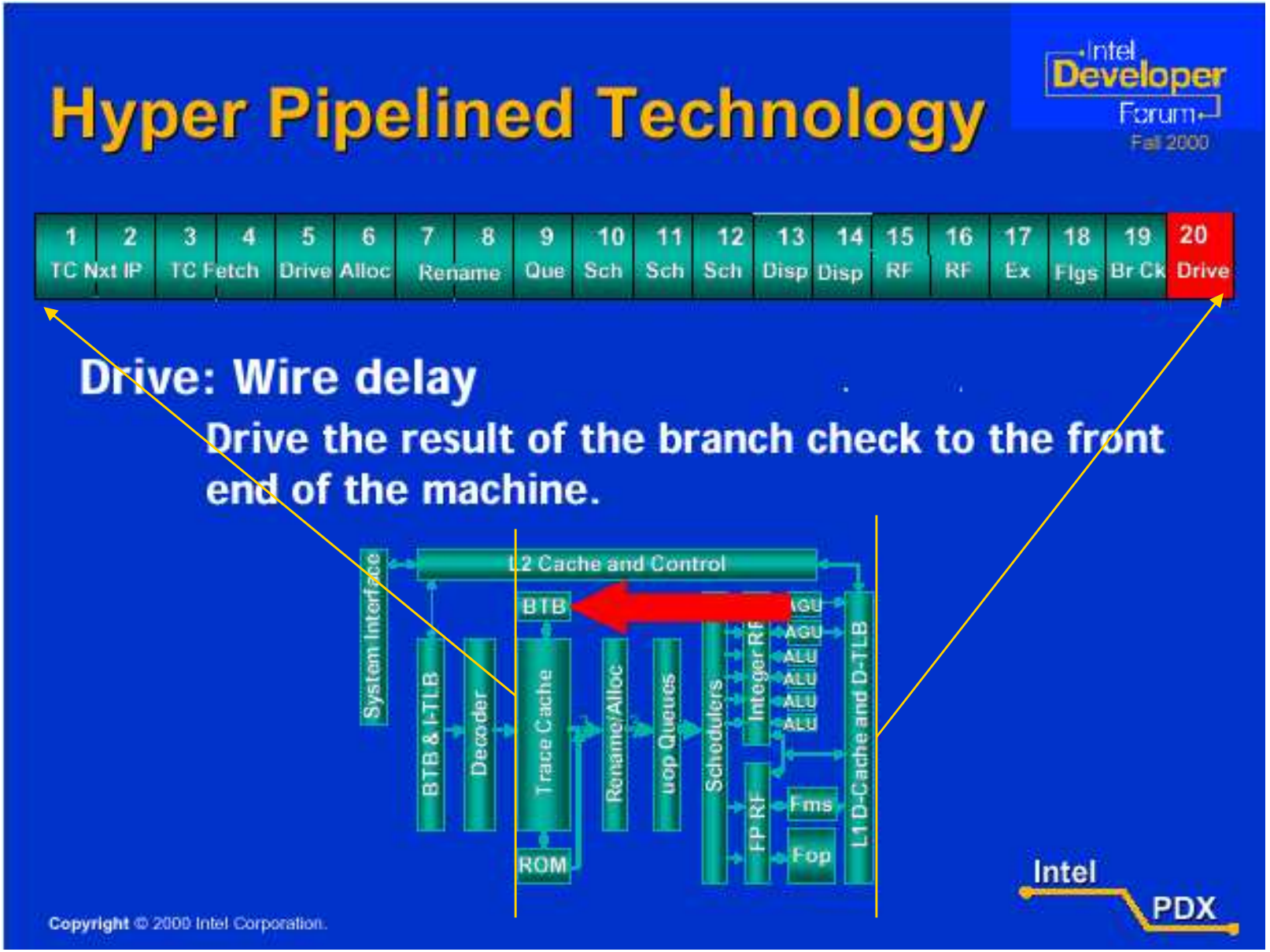


Figure SYSD9　SOC Consumer Stationary Design Complexity Trends

Consumer Stationary Driver : **ゲーム機など**

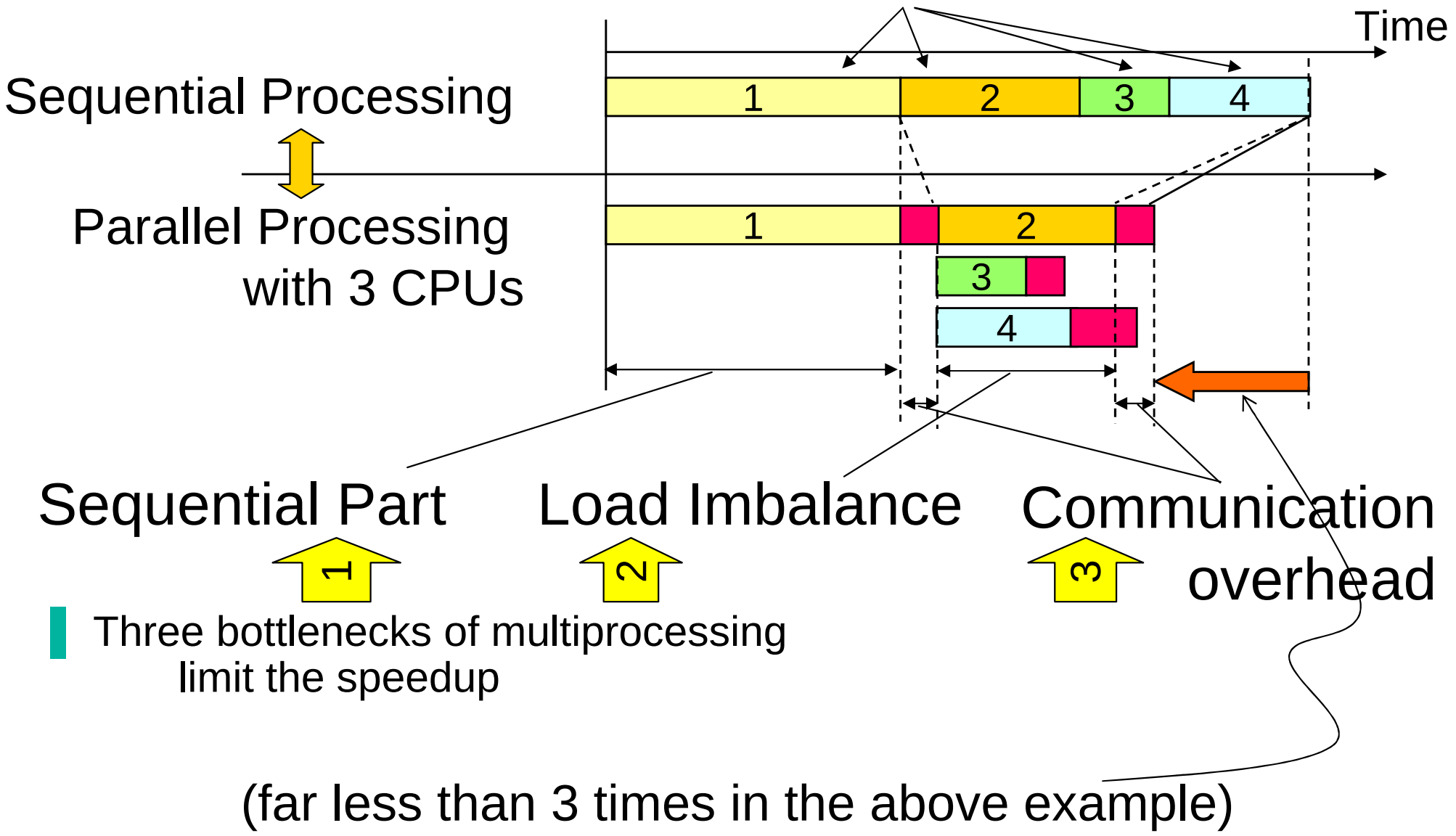
Intel pipelineの進化



http://itassociate.com/PCJ/PCJ1/topics/Pentium4.html

Bottlenecks of Multiprocessing

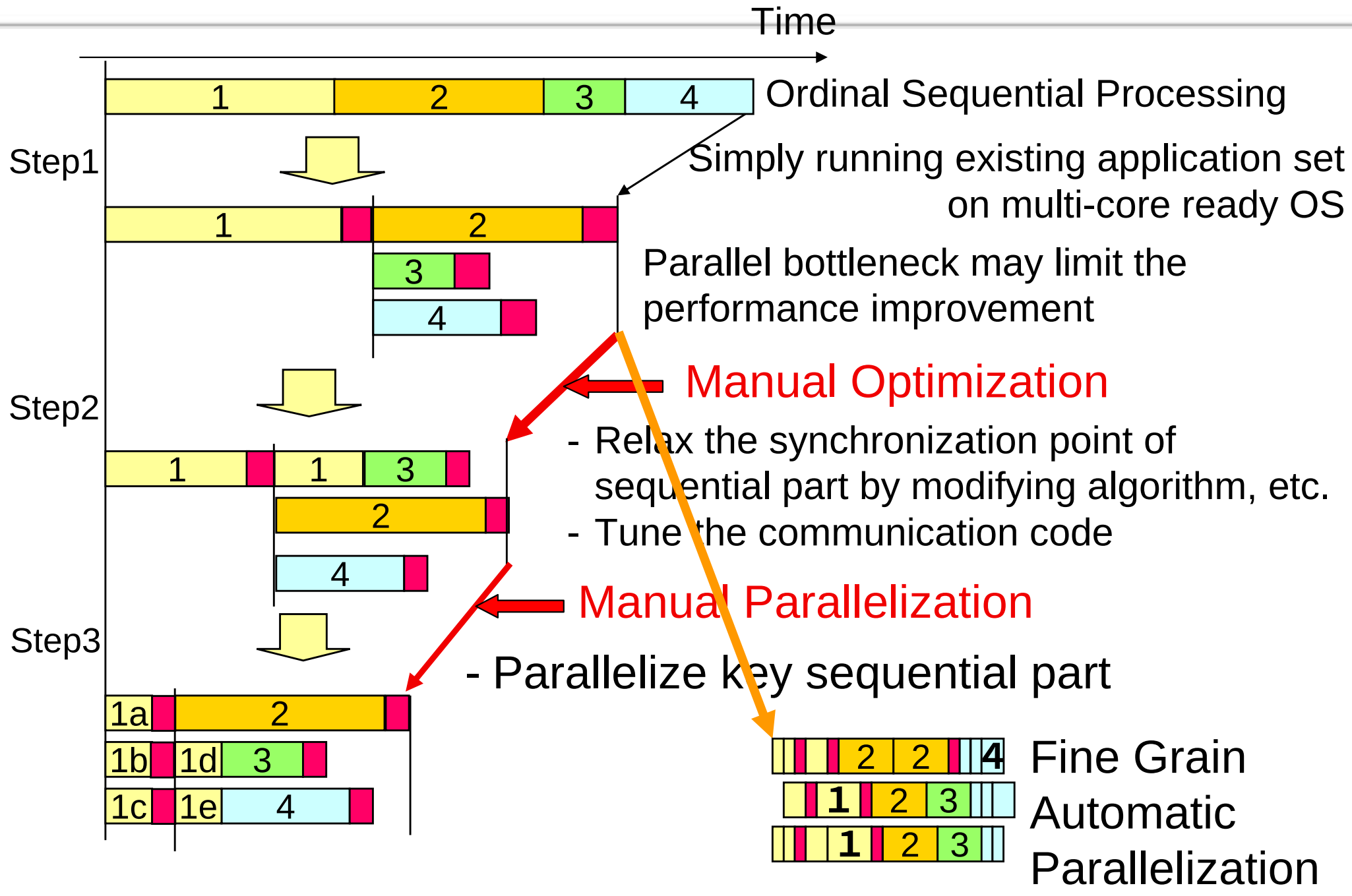
Example) System activity consists of 4 tasks (or 4 threads)



- Three bottlenecks of multiprocessing limit the speedup

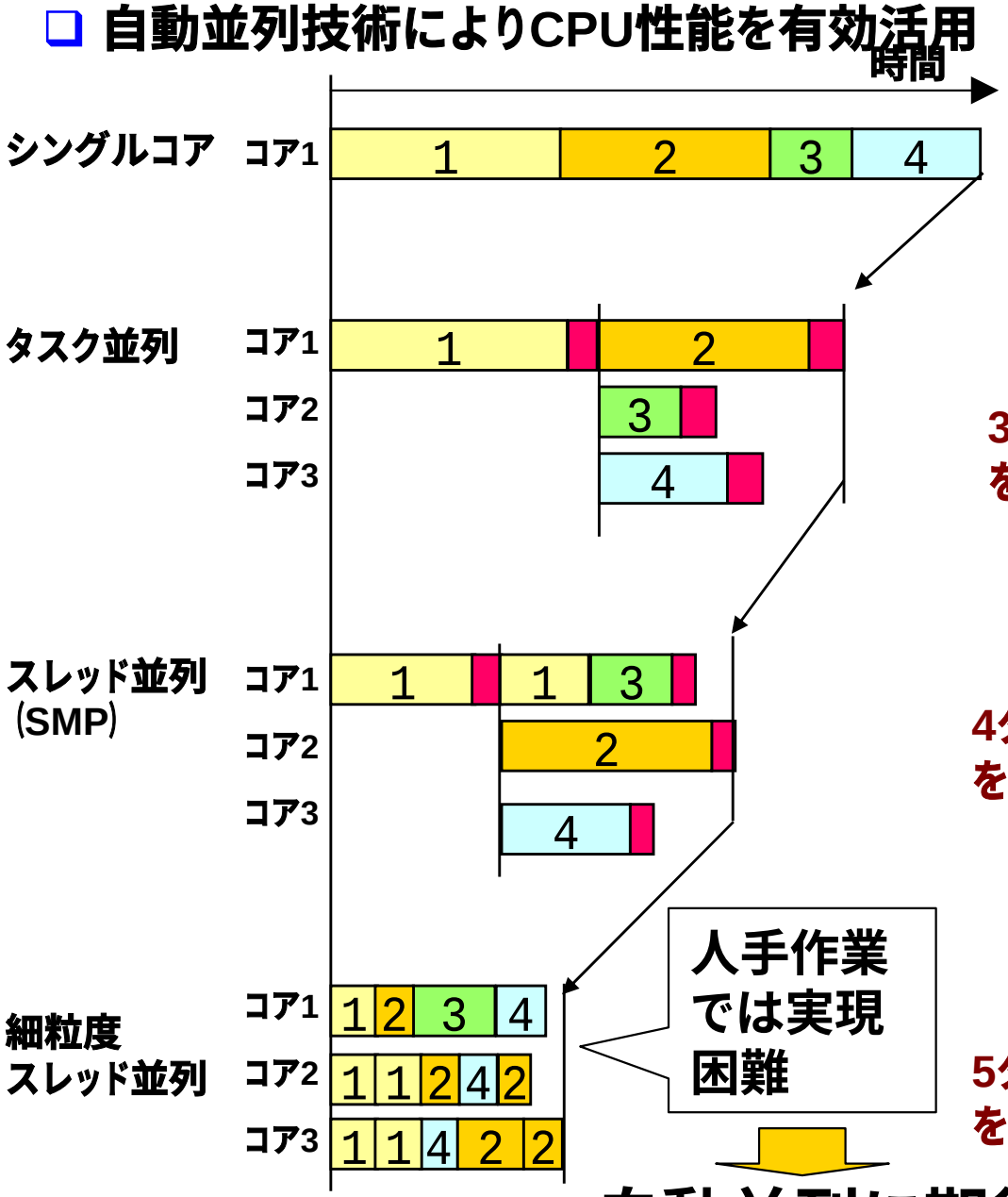
(far less than 3 times in the above example)

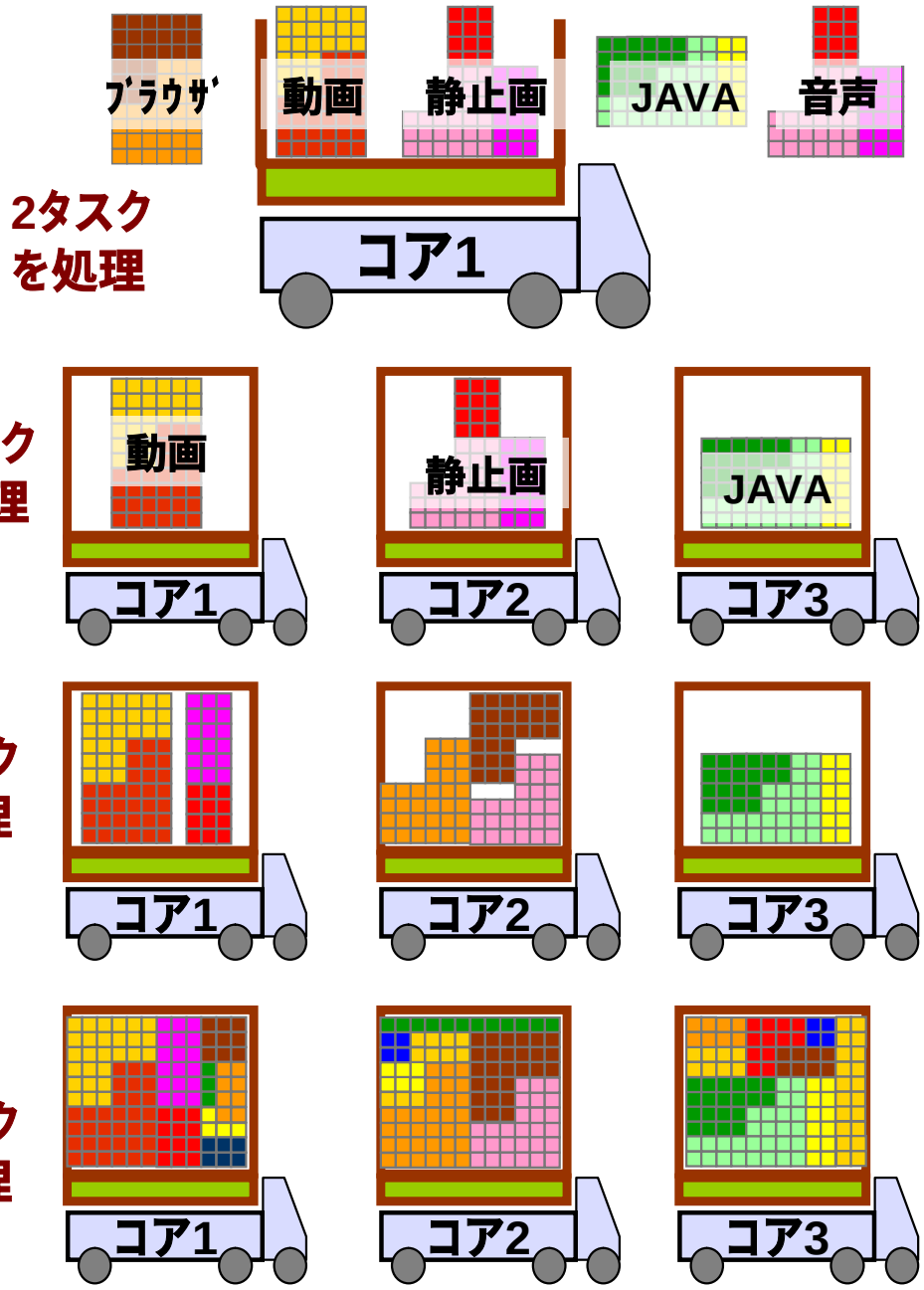
Performance Optimization for Multi-core



マルチコア技術の進展

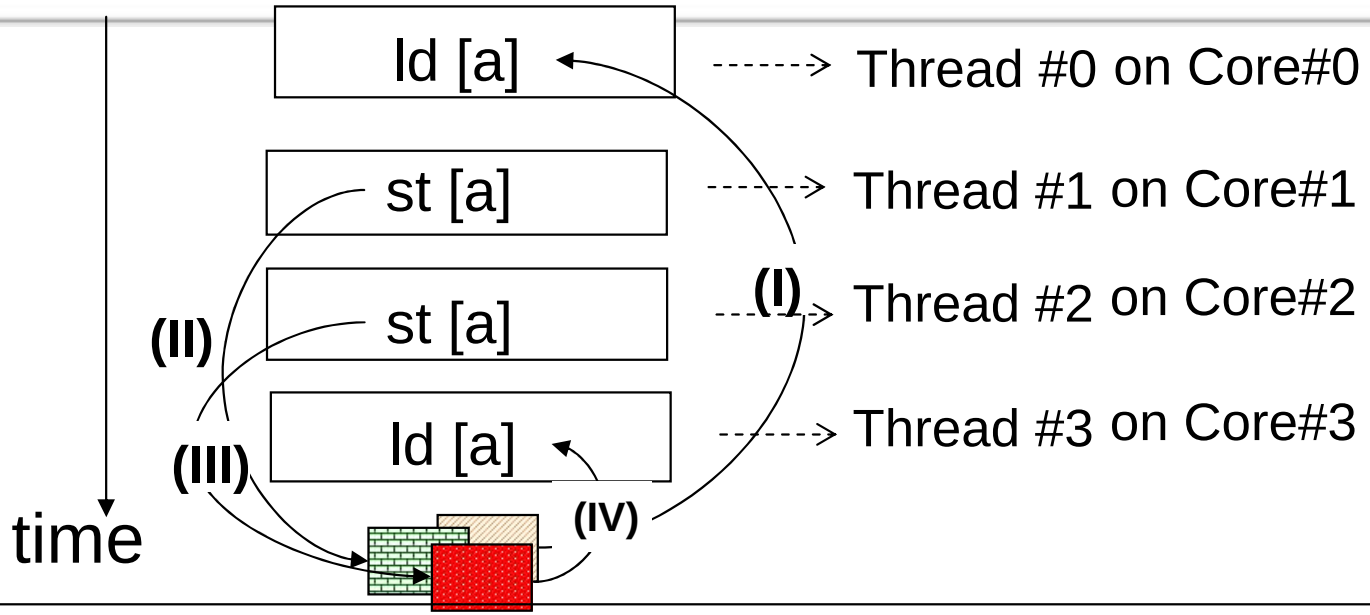
- 自動並列技術によりCPU性能を有効活用



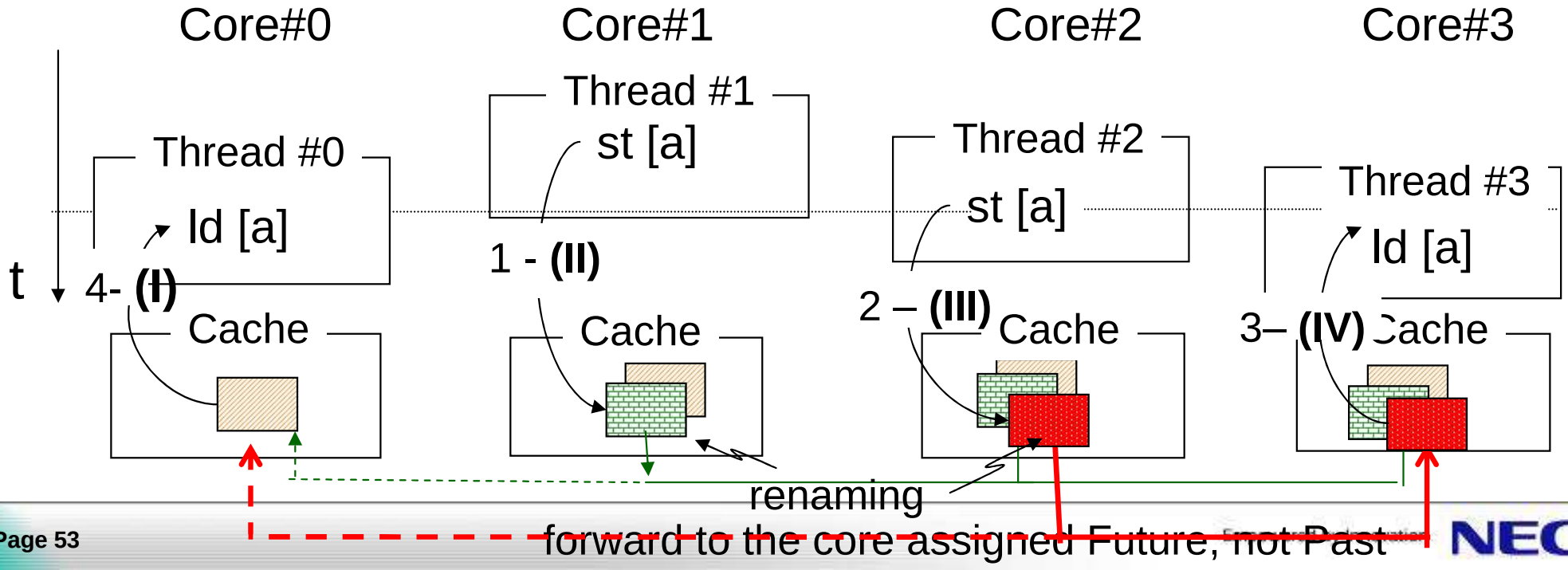


Memory Access Ordering Resolution by HW

Sequential processing

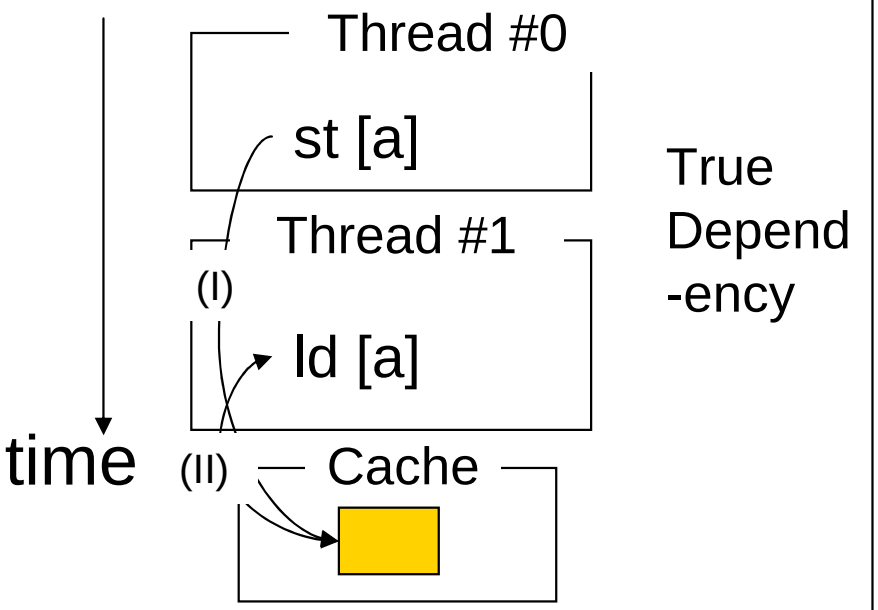


- Scenario 1: only the latest write is propagated (Memory renaming)

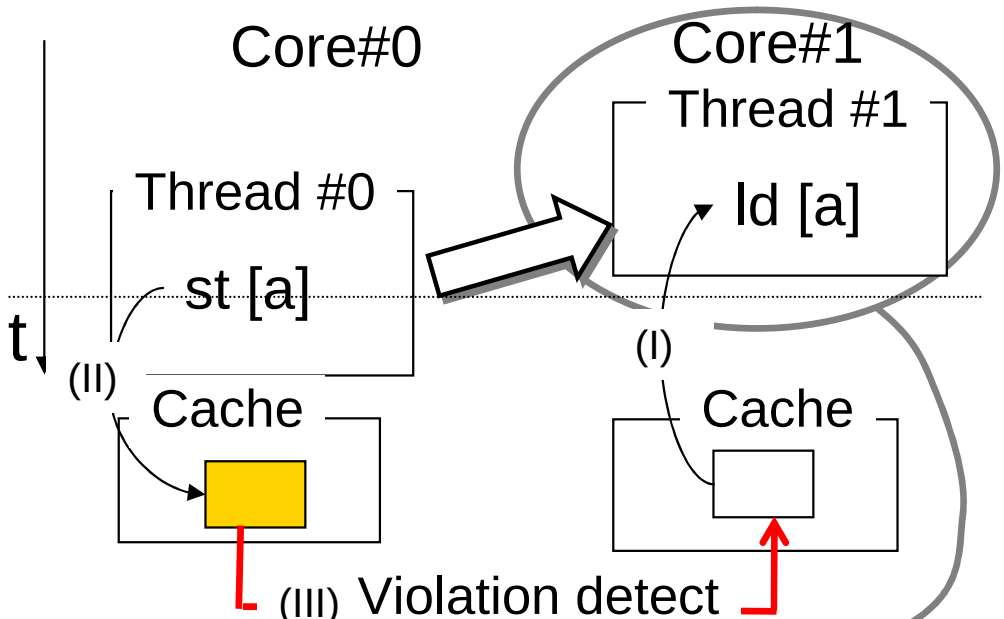


Memory Access Ordering Resolution : cont.

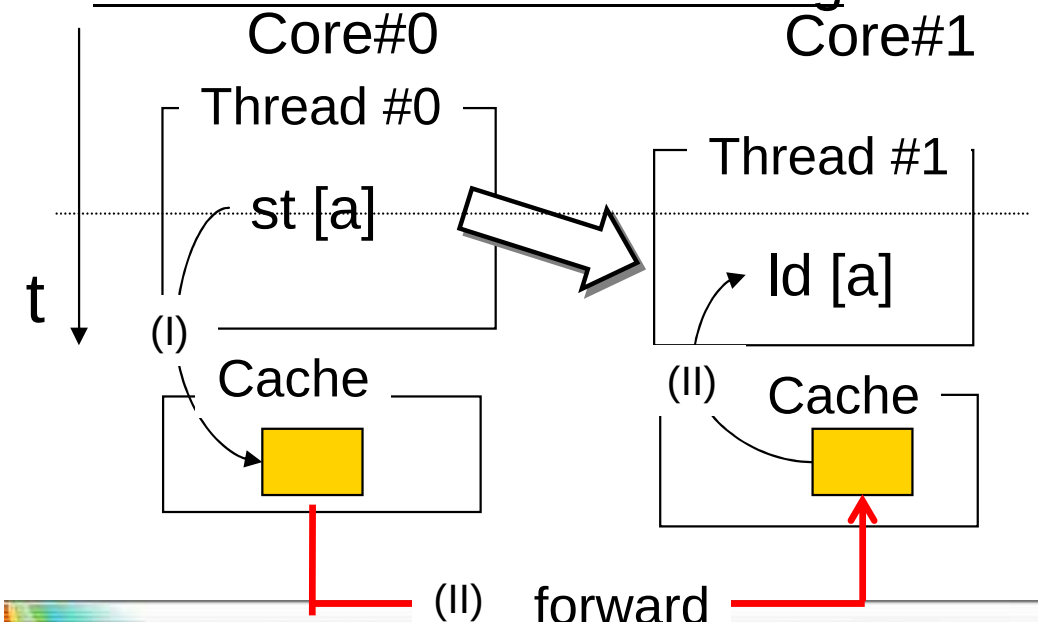
Sequential processing



Scenario 3: violation detection



- Scenario 2: data forwarding



- Re-execution of thread #1
- Execute threads speculatively prepared for re-execution
- Speculative Multithreading

プログラムに内在する並列性　(制御並列処理)

Wallの研究: 非数値計算でのSuperScalarで引き出せる性能向上限界が、4.1~7.4倍

↓

M.S.Lam and R.P.Wilson 理論研究：

制御依存のないコードを確定的あるいは投機的に実行することで性能向上する可能性を示した。(CPU Arch. 研究者の心の支え?)

(全命令の実行時間を1クロックと仮定…

→ load latencyが苦労の元凶なのに……)

	BASE	CD	CD-MF	SP	SP-CD	SP-CD-MF	ORACLE
アプリケーション	分岐を逐次	非依存BB実行	複数Branch	分岐投機			神様
awk	2.85	3.24	5.32	9.22	12.89	41.88	242.77
ccom	2.13	2.51	5.61	6.92	9.83	18.05	46.80
equntott	1.98	2.05	5.21	6.40	18.09	225.90	3282.91
espresso	1.51	1.54	7.49	4.16	19.55	402.85	742.30
gcc(cc1)	2.10	2.55	14.63	7.76	13.18	66.29	174.50
irsim	2.31	2.66	11.89	8.40	15.82	45.86	265.42
latex	2.71	3.17	6.18	7.60	9.72	18.65	131.26
Harmonic Mean	2.14	2.39	6.96	6.80	13.27	39.62	158.26
matrix	293.00	432.00	68324.00	36192.00	108575.00	180632.00	188470.00
spice2g6	2.14	2.29	16.80	8.11	25.28	196.76	843.60
tomcatv	22.23	42.77	3237.00	124.00	1881.00	3918.00	3918.00

出典：“Limits of Control Flow on Parallelism,”M.S.Lam et. al., ISCA’92, pp. 46-57, 1992.

Pinot新聞発表: 日刊工業新聞 2006年01月19日 朝刊◇ 27面

◎日刊工業新聞 2006年01月19日 朝刊◇ 27面 (20060119900196)

4カ月がたった3分に

NECが並列ソフト自動作成技術

〝投機的処理〟で塊を並列化

「賭けが外れたらやり直す」発想で

CPUの稼働率アップも

並列処理による性能改善

ベンチマークソフト

- MediaBench : **メディア処理系のアプリを集めたフリーのベンチマーク**
- Blowfish:**対称ブロック暗号**
- Mad: Minimum Absolute Dierence – **動きベクトル探索**
- Susan: **スムージング**

Pinotでのハードウェア改造量