社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－ARC－177 (8)
2008－HPC－114 (8)
2008／3／5

性能歩留まり改善を目的とする演算器カスケーディングの提案

渡辺 慎吾[†] 橋本 昌宜[‡] 佐藤 寿倫[††,§]

[†] 九州工業大学大学院 情報工学研究科 情報科学専攻
[‡] 大阪大学大学院 情報科学研究科 情報システム工学専攻
[††] 九州大学 システムＬＳＩ研究センター
[§] 独立行政法人科学技術振興機構，CREST

半導体製造プロセスの微細化が進展するにつれ，製造ばらつきの拡大という深刻な問題が顕在化している．それによりトランジスタの特性ばらつきが増大し，タイミング歩留まりの悪化が懸念されている．我々は回路遅延の統計的性質に着目し，演算器をカスケーディングすることで遅延ばらつきを縮小することを検討している．本稿では，演算器の統計的遅延解析とプロセッサ性能の評価とから，カスケーディングの性能歩留まり改善に対する効果を調査する．その結果，ばらつき問題への対策にはマイクロアーキテクチャの大局的な検討が必要であるという知見が得られた．

ALU Cascading for Improving Performance Yield

SHINGO WATANABE[†] MASANORI HASHIMOTO[‡] TOSHINORI SATO[††,§]

[†]Department of Artificial Intelligence, Kyushu Institute of Technology
[‡]Department of Information Systems Engineering,Osaka University
[††]System LSI Research Center, Kyushu University
[§]Japan Science and Technology Agency, CREST

As semiconductor technologies are aggressively advanced, the problem of parameter variations is emerging. Parameter variations in transistors affect circuit delay, resulting in serious yield loss. We exploit the statistical characteristics in circuit delay, and are investigating ALU cascading for variation reduction. From the statistical timing analysis in circuit level and the performance evaluation in processor level, this paper tries to unveil how efficiently ALU cascading improves performance yield of processors. We find that innovations are required for managing parameter variations in microarchitecture level.

1. はじめに

半導体の微細化によりトランジスタ特性のばらつきが拡大するという深刻な問題が顕在化している．ばらつきは動的なばらつきと静的なばらつきに分類される[4]．動的なばらつきはトランジスタの動作温度や電源電圧など周辺環境の揺らぎを要因とするばらつきである．静的なばらつきは製造ばらつきと呼ばれ，製造時にトランジスタのゲート形状や不純物濃度などに生じる微細な揺らぎを要因とする．製造時の揺らぎは本質的に避けがたく，トランジスタサイズの縮小に伴い拡大している[3]．製造ばらつきはチップ間ばらつきとチップ内ばらつきに分類される．チップ間ばらつきは各チップの平均値が変動するばらつきであり，チップ内ばらつきはチップ内において各トランジスタの特性などが変動するばらつきである．微細化の進展にともないチップ内ばらつきが深刻化している[9]．

トランジスタの特性ばらつきにより回路遅延が変動する[3] と，製造されたチップの中にはタイミング制約を満たせないものが現れる．今後製造ばらつきが拡大すると，タイミング制約を満たせないチップの割合が増加し，性能歩留まりが悪化することが予想される．性能ばらつきを意識して歩留まりを改善しようという試みが現れ始めているが，回路あるいはマイクロアーキテクチャのレベルでばらつきそのものを縮小しようとする試みはほとんどない．本研究では，回路遅延の統計的性質に着目し，マイクアーキテクチャレベルで遅延そのものを縮小して，性能歩留まりを改善することを目標としている．

回路遅延の統計的性質を活用するために，演算器のカスケーディングを利用することを提案している[10]．カスケーディングすることで演算器の論理段数を大きくすることができるため，遅延ばらつきの縮小が期待できる．また，カスケーディングにより演算器の遅延は増加するが，同一サイクルで複数命令を実行できるためサイクル当たりの実行命令数を改善可能である．本稿では，カスケーディングされた演算器の統計的遅延解析を行い，それを採用するプロセッサの性能を考

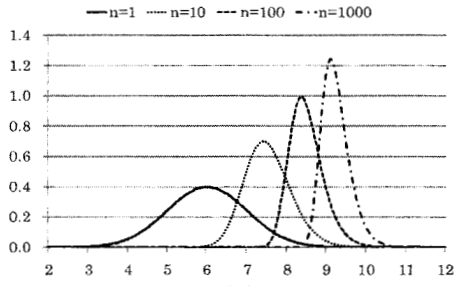


図 1 クリティカルパス数の影響

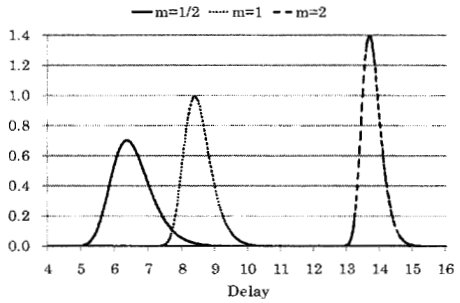


図 2 論理段数の影響

慮して，その性能歩留まりの改善度合いを評価する．

本稿の構成は以下の通りである．次節で回路遅延の統計的性質を説明する．3 節で演算器カスケーディングについて説明する．4 節で演算器の統計的遅延解析を行う．5 節では演算器カスケーディングを採用するプロセッサの性能を評価する．6 節でプロセッサの性能を考慮した性能歩留まりを評価する．7 節でまとめとする．

2. 回路遅延の統計的性質

回路遅延の統計的性質について説明する．図 1 はクリティカルパスの総数が回路遅延に与える影響を示している[5)6)]．横軸は遅延を，縦軸は出現度を示す．各パスは互いに無相関で，平均 (μ) が 6，標準偏差 (σ) が 1 である正規分布 $N\left(6,1^2\right)$ に従うものと仮定している．図 1 の n はクリティカルパスの本数であり，$n=1$ の場合が $N\left(6,1^2\right)$ の正規分布である．クリティカルパスの本数が増加するにしたがって，遅延の平均は増大する方向へ移動している．これは回路中の全てのパスの中で最も遅延の大きなパスが，その回路の遅延時間を決定するためである．この性質から，遅延の揃ったパスを多くもつ回路ほど，ばらつきによる回路遅延増大の影響を受けやすいことがわかる．一方で分布の広がり，σ はクリティカルパス数が増加するにしたがって小さくなっている．

図 2 はクリティカルパスの論理段数が回路遅延に与える影響を示している．図 1 と同様に横軸は遅延を，縦軸は出現度を示す．図 2 の m は相対的な論理段数を示す．$m=1$ が図 1 に示す $n=100$ のときの遅延分布である．$m=1/2$ は論理段数が 1/2 倍になった場合を，$m=2$ は論理段数が 2 倍になった場合を示している．論理段数が大きくなるとゲート遅延の変動が平均化されるため，遅延のばらつきは縮小する．論理段数 m が大きくなるに従って σ は $1/\sqrt{m}$ で相対的に小さくなる．このことは論理段数が大きくなると遅延は増加するがばらつきの度合いは縮小することを示している．

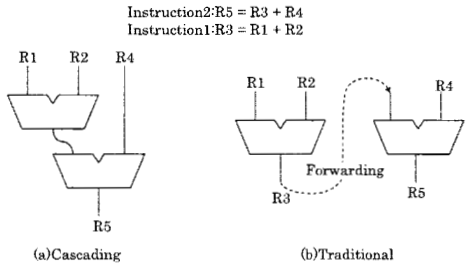


図 3 演算器カスケーディング

3. 演算器カスケーディング

回路遅延の統計的性質を活用し遅延ばらつきを縮小することを目的として，演算器カスケーディングの利用を提案している[10)]．演算器カスケーディングでは図 3(a) のように 2 つの演算器を用いて，一方の演算器の出力を他方の演算器の入力に接続する．カスケーディングされた演算器を用いて依存関係のある複数命令を 1 サイクル内で連続実行する[8)]．カスケーディングすることで論理段数が増加し，回路遅延のばらつきを縮小することが可能である．

従来のスーパスカラプロセッサは，図 3(b) のように依存関係にある命令を並列に処理できない．図の先行命令 1 と後続命令 2 には依存関係があり，命令 1 が演算を終了しないと命令 2 を実行できない．そのため少なくとも 2 サイクルを要する．演算器カスケーディングを用いると，それら 2 命令を 1 サイクルで実行できる．サイクル当たりの実行命令数を増やすことができ，IPC(Instructions per Cycle) を改善できる．しかし，カスケーディングにより演算器の遅延は増加し，動作周波数が低下する．IPC を改善できたとしても，動作周波数が大きく低下すればプロセッサ性能は低下する．本稿ではこれらを考慮した評価を行う．

4. 演算器の統計的遅延解析

4.1 対象回路

対象は 32 ビット桁上げ選択加算器 (Carry Select Adder:CSLA) である．2 入力 CSLA とそれをカスケーディングした CSLA(3 入力 CSLA と呼ぶ) を Verilog

表 1　合成条件

演算器	32bit Carry Select Adder
セルライブラリ	HITACHI 0.18μm
動作条件	TYPICAL 出力ポートの付加容量：30 出力ポートのファンアウト数：4 入力ポートのドライブ抵抗：1.5
制約条件	遅延：最小 動的電力：最小

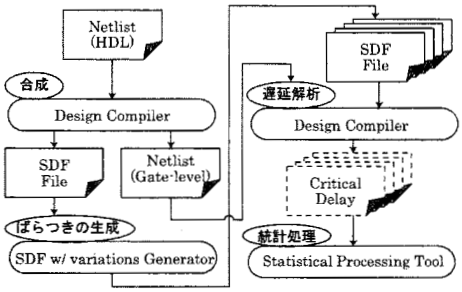


図 4　統計的評価の概要

HDL で設計する．Synopsys 社の DesignCompiler により，HITACHI 0.18um セルライブラリを用いてゲートレベル・ネットリストを合成する．表 1 に合成時の条件をまとめる．また，2 つの 2 入力 CSLA の接続は HDL レベルで行い，階層化しないでフラットに合成する．

4.2 解析方法

図 4 に示す回路の統計的遅延解析方法を説明する．図中，DesignCompiler のみが市販ツールであり，SDF w/ variations Generator と Statistical Processing Tool は自作ツールである．

① 対象回路を合成しネットリストと Standard Delay Format(SDF) ファイルを得る．

② ① の SDF ファイルを基に，各ゲート遅延がばらついた状態の SDF ファイルを作成する．

③ ② の工程を繰り返し，ばらつきの異なる多数の SDF ファイルを作成する．

④ ① で合成したネットリストと ③ で作成した SDF ファイルを 1 つ用いて，DesignCompiler で対象回路の静的遅延解析を行う．これを全ての SDF ファイルで実施する．

⑤ 全ての遅延解析結果を統計処理する．

② でゲート遅延をばらつかせる方法を説明する．SDF ファイルには，論理ゲートの各パスの立ち上り遅延と立ち下り遅延が記述されている．これらの遅延をモンテカルロシミュレーションと同様の要領で変動させる．各ゲート遅延は互いに無相関とし，それぞれに正規分布に従った変動量を与える．文献 3) によると 65nm テクノロジでのゲート遅延のばらつきは σ/μ で 0.064 であることから，ゲート遅延変動量を σ/μ を 0.064 とする．また，作成する SDF ファイルのサンプル数はそれぞれの解析で 10,000 とする．

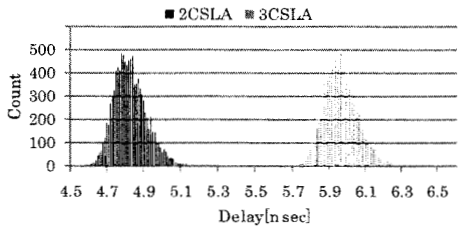


図 5　演算器遅延の解析結果

表 2　演算器遅延の平均と標準偏差

	μ	σ	σ/μ	Imp.(%)
2CSLA	4.82	0.088	0.0183	-
3CSLA	5.96	0.093	0.0155	15.1

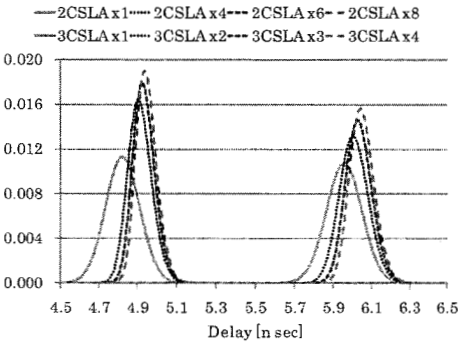


図 6　演算器数に対する回路遅延分布の変化

表 3　演算器数に対する回路遅延の平均値と標準偏差の変化

		μ	σ	σ/μ
2CSLA	x1	4.82	0.088	0.0183
	x4	4.91	0.062	0.0126
	x6	4.93	0.057	0.0115
	x8	4.95	0.054	0.0109
3CSLA	x1	5.96	0.093	0.0155
	x2	6.01	0.076	0.0127
	x3	6.04	0.069	0.0115
	x4	6.05	0.065	0.0107

4.3 解析結果

2 入力 CSLA と 3 入力 CSLA の遅延解析結果を図 5 に示す．横軸は遅延を，縦軸は度数を示す．図中の左側の一群が 2 入力 CSLA の，右側の一群が 3 入力 CSLA の分布である．表 2 に遅延の μ，σ，及び σ/μ をまとめる．表 2 の 4 列目は 2 入力 CSLA の σ/μ に対する 3 入力 CSLA の σ/μ の改善率を示している．2 入力 CSLA と比較して 3 入力 CSLA は遅延が大きいため，当然 μ と σ は大きくなる．しかし，σ/μ は小さくなっており，遅延ばらつきが小さくなっていることが分かる．言い換えると，演算器カスケーディングにより遅延ばらつきを縮小できている．

2 節で説明したクリティカルパスが回路遅延に与える影響を考慮すると，演算器数がプロセッサ全体の遅延に影響を与えることが分かる．スーパスカラプロセッ

表 4　基準プロセッサの構成

Fetch width	8 instructions
L1 I-cache	32KB, 2way, 1 cycle
Branch predictor	gshare + bimodal
	gshare:4K entries, 12 histories
	bimodal:4K entries
RUU size	128 entries
Issue width	4/6/8 instructions
Integer ALUs	4/6/8 units, 1 cycle
Integer multipliers	1 units
	MULT 3 cycles, DIV 20 cycles
Floating ALUs	4 units, 2 cycles
Floating multipliers	1 unit
	MULT 4 cycles, DVI 12 cycles,
	SQRT 24 cycles
L1 D-cache port	2 ports
L1 D-cache	32KB, 4way, 2 cycles
Unified L2 cache	4MB, 8way, 8cycles
Memory	150 cycles
Commit width	8 instructions

サは演算器を複数持つが，演算器数が増加すると実行ステージのクリティカルパス数が増加する．その影響について調査する．遅延解析で得られた遅延分布を用いて統計的 MAX 演算[2] で求める．なお，これ以後は遅延分布を正規分布に近似して扱う．

図 6 に演算器数を変えたときの遅延分布を示す．横軸は遅延を，縦軸は出現度を表す．図中の左側の一群が 2 入力 CSLA の分布を，右側の一群が 3 入力 CSLA の分布である．2 入力 CSLA は演算器数が 1，4，6，8 のときの分布を，3 入力 CSLA では演算器数が 1，2，3，4 のときの分布をそれぞれ示す．このとき，基本構成要素である 2 入力 CSLA の数では両者で同じになっている．図から演算器数が増加すると，遅延の平均は増加し，ばらつきは縮小していることが確認できる．

表 3 に遅延の μ，σ，σ/μ をまとめる．基本構成要素の 2 入力 CSLA が 1 から 8 へ増加するとき，2 入力 CSLA では μ が 2.6%増加し σ/μ が 40%縮小している．一方，3 入力 CSLA では μ が 1.6%増加し σ/μ が 31%縮小している．しかし，基本構成要素の 2 入力 CLSA の数が同じときには両者に有意差は無く，演算器数の回路全体の遅延に与える影響は同程度であることが分かった．

5. プロセッサの IPC の評価

本節では演算器カスケーディングがプロセッサの IPC に与える影響を評価する．これ以降，それを利用するプロセッサをカスケード・プロセッサと呼ぶ．

5.1 評価環境

SimpleScalar ツールセット[1] を用いて評価環境を構築した．命令セットは Alpha 命令セットを用いる．ベンチマークプログラムには SPEC2000 CINT を用い，初めの 5 億命令をスキップ後，1 億命令をシミュレーションする．カスケーディングを施さない比較基準となるプロセッサ (基準プロセッサと呼ぶ) の構成を表 4 にまとめる．基準プロセッサは 4 命令，6 命令，8 命令発行のプロセッサを用意する．これらのプロセッサは，命令発行幅と整数 ALU の数以外の構成は共通である．

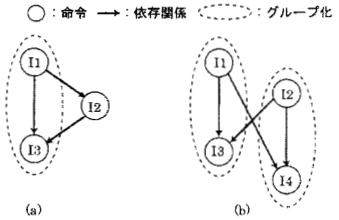


図 7　デッドロック状態のグループ化

5.2 カスケード・プロセッサの構成

カスケード・プロセッサの構成について説明する．カスケーディングする演算器は整数 ALU のみを対象とする．本稿では 4 節で求めた遅延分布を整数 ALU の遅延分布とし評価を行う．カスケード・プロセッサの実行ステージの遅延は，カスケーディングのため基準プロセッサに比べ増加する．典型値での遅延が 25%増加するので，基準プロセッサに対して 80%の動作周波数となる．以上は，実行ステージがプロセッサの動作周波数を決定するという仮定に基づいている．微細化の進展による実行ステージと他のステージにおける遅延のギャップのために，プロセッサの歩留まりは主に実行ステージにおけるタイミングエラーで決定されると言われており[7]，妥当な仮定である．また，評価した遅延分布は実行ステージについてのみであるが，動作周波数が低下することから，実行ステージ以外のステージはタイミング制約が緩和される．

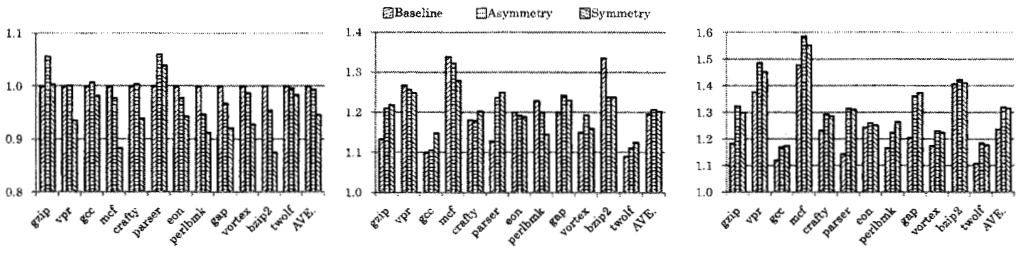
3 入力 ALU での演算カスケーディングには以下の制約を設ける．以降，カスケーディング実行される演算を特定することをグループ化と呼び，カスケーディングされない命令をグループ化不可能命令と呼ぶ．

① グループ化の対象は整数 ALU 命令のみとする．ただしロード／ストア命令はアドレス計算とメモリアクセスに分離されており，アドレス計算部はグループ化の対象とする．

② 2 命令のグループ化のみ許す．先行命令を生産者，後続命令を消費者と呼ぶ．また，どの命令も複数のグループには属さない．

③ 命令ウィンドウへのディスパッチ時にグループ化を施す．グループ化の探索範囲はウィンドウ内に限られる．生産者を見つけられない場合には，グループ化不可能命令となる．

④ ディスパッチ時に利用不可能なオペランドを 2 つ持つ命令はグループ化の消費者にはなり得ない．これは図 7 に示すようなグループ化を防ぐためである．グループ化された命令は同時に発行されなければならないので，図のグループ化は命令発行できないデッドロック状態を引き起こす．



(a) (b) (c)

図 8 IPC の評価結果：(a)4 命令発行構成，(b)6 命令発行構成，(c)8 命令発行構成

表 5 カスケード・プロセッサの構成

	Symmetry	Asymmetry
4 issue :2in ALU	–	1 unit, 1 cycle
:3in ALU	2 units, 1 cycle	1 unit, 1 cycle
6 issue :2in ALU	–	4 units, 1 cycle
:3in ALU	3 units, 1 cycle	1 unit, 1 cycle
8 issue :2in ALU	–	4 units, 1 cycle
:3in ALU	4 units, 1 cycle	2 units, 1 cycle
Common :L2 cache	4MB, 8 way, **7 cycles**	
:Memory	**120 cycles**	

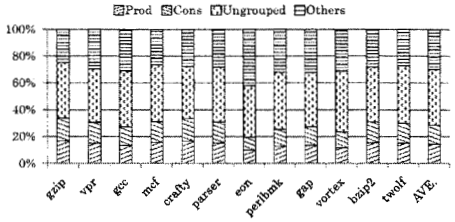


図 9 プログラム実行時の命令の内訳

カスケード・プロセッサの演算器構成について述べる．演算器構成は 2 種類考えられる．1 つは全て 3 入力 ALU とする構成であり，これを対称構成と呼ぶ．もう 1 つは 2 入力 ALU と 3 入力 ALU を混在させる構成であり，これを非対称構成と呼ぶ．前者の構成でグループ化不可能命令を実行するには何らかの偽命令，例えば 0 と 0 の加算などの命令とグループ化する必要がある．一方，後者では 3 入力 ALU 数が対称構成と比べ少なくなるため，グループ化命令のスループットが低下することになる．これらの演算器構成が IPC に与える影響も評価する．

表 5 にカスケード・プロセッサの構成をまとめる．表 4 に示した基準プロセッサの構成と異なる項目のみを示す．メモリと 2 次キャッシュのアクセスレイテンシは，動作周波数が低下するためにサイクル数では小さくなっている．

5.3 評価結果

図 8 に 4 命令，6 命令，8 命令発行のプロセッサ構成での IPC をそれぞれ示す．全ての値は 4 命令発行基準プロセッサの IPC で正規化されている．横軸はプログラムを示す．図中の Baseline は基準プロセッサを，Asymmetry は非対称構成の，Symmetry は対称構成のカスケード・プロセッサをそれぞれ示す．

はじめに，演算器カスケーディングが IPC に与える影響について考える．IPC は発行幅により異なる影響を受けている．4 命令発行ではカスケード・プロセッサの IPC は多くのプログラムで基準プロセッサよりも低い．平均では非対称構成で 0.5%，対称構成で 5.4%，それぞれ IPC が低下している．IPC が低下するのは，4 命令発行ではプロセッサは十分な命令レベル並列性を確保できるためである．基準プロセッサは並列性があればサイクル当たり 4 命令実行できる．対して，カスケード・プロセッサはグループ化不可能命令が存在すると ALU を全て利用することができない．非対称構成，対称構成ともにグループ化不可能命令はサイクル当たり 2 命令しか実行できないため，実効スループットが低下する．

6 命令発行では若干の改善が得られている．非対称構成で平均 1.0%，対称構成で平均 0.6%向上している．8 命令発行では改善が拡大している．非対称構成と対称構成での IPC の改善は，平均でそれぞれ 6.9%と 6.4%である．IPC の改善度合いが向上するのは，命令発行幅が増加するに従って命令レベル並列性の抽出が困難になるためである．基準プロセッサでは ALU の利用効率が低下する．対して，カスケード・プロセッサでは依存関係にある命令をグループ化することで ALU の利用効率が上がる．これらのことから，演算器カスケーディングはより発行幅の広いプロセッサに適用することで IPC を改善すること可能であることが分かる．

次に，演算器構成による IPC の変化について考える．全ての発行幅で，対称構成よりも非対称構成の方が大きな IPC を示している．しかし，4 命令発行に比べ，6 命令と 8 命令発行では，IPC の差は小さい．非対称構成の方が IPC をより向上できている．対称構成では演算器の利用効率が悪いためである．これはグループ化命令が不可能命令に対してそれほど多くないことを示している．次にこれを確認する．

図 9 に 4 命令非対称構成プロセッサでのプログラム実行時の命令の内訳を示す．縦軸は全実行命令数に占める割合を示している．Prod と Cons がグループ化された命令を示し，それぞれ生産者と消費者の割合を示す．Ungrouped はグループ化不可能命令を，Others

はグループ化対象外の命令を示す．グループ化命令は平均で29%，グループ化不可能命令は平均で41%存在する．その他のプロセッサ構成でも同様の傾向のため，グラフを省略する．グループ化される命令はそれほど多くないことがわかる．非対称構成の方がより高いIPCを得ることができた要因である．

6. 性能歩留まり

4節で求めた遅延分布と5節で評価したIPCとから性能歩留まりについて考察する．動作周波数とIPCとの積からプロセッサ性能を求め，その分布を用いて歩留まりについて考察する．その分布上で性能の下限を与えると，それを満たすものと満たしていないものに分けることができ，歩留まりを比較できる．

図10に性能分布を示す．横軸は性能を示す．4命令発行基準プロセッサの平均性能で正規化している．カスケード・プロセッサのIPCはその値の大きい非対称構成のものを用いている．縦軸は出現度を示す．非対称構成では2入力演算器と3入力演算器が混在するが，クリティカルパスとなるのは3入力演算器であるので，3入力演算器数のみ考慮する．

基準プロセッサとカスケード・プロセッサをそれぞれの命令発行幅で比較すると，平均性能は全て基準プロセッサの方が高いことが分かる．IPCの評価では6命令と8命令発行のときに，基準プロセッサに対してカスケード・プロセッサに改善が見られている．それにも関わらず性能が低下しているのは，動作周波数の低下による性能悪化が大きいためである．一方，それぞれの命令発行幅で，ばらつきを比較すると，カスケード・プロセッサの方がσ/μが僅かに大きい．これは演算器構成が要因である．非対称構成では3入力演算器数が少ないため，クリティカルパス数の増加によるばらつき縮小の効果が少ないためである．

演算器カスケーディングを施すと動作周波数が低下するため，同一の命令発行幅のプロセッサ構成では性能は悪化する．6命令と8命令発行のプロセッサ構成ではカスケーディングでIPCを改善しているが，周波数の低下を打ち消すほどではない．また，非対称の演算器構成では演算器数が少ないため，周波数ばらつきの改善は得られてない．

以上の考察から以下の知見が得られる．演算器単体ではばらつきを改善できたが，演算器の遅延増の影響，演算器数の影響，IPCの改善度合いから総合的に判断すると，プロセッサ性能を考慮した性能歩留まりを改善できていないと結論できる．このことは，ばらつきの問題は局所的な改善を検討するだけでは不十分で，プロセッサシステム全体を大局的に検討する必要があることを示唆している．本研究ではばらつきの問題への解決策を提示できなかったが，この知見は今後の研究において大きな意味があると言えよう．

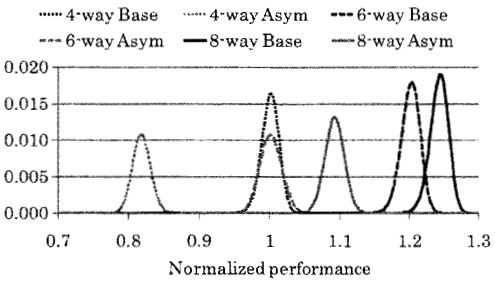


図10 プロセッサ性能の分布

7. ま と め

マイクロアーキテクチャレベルで性能歩留まりを改善するために，演算器カスケーディングの利用を提案した．評価の結果，回路レベルではばらつきを縮小できるものの，プロセッサレベルではばらつきを縮小することはできなかった．プロセッサレベルで性能歩留まりを改善するには，小手先の工夫では不十分で，抜本的なマイクロアーキテクチャの見直しが必要であるという知見が得られた．

謝辞 本研究は一部，科学研究費補助金・基盤研究A(課題番号19200004)および，科学技術振興機構CRESTプログラムの支援による．なお，東京大学VDECを通じて提供された日立製作所の0.18μmライブラリを使用している．

参 考 文 献

1) T. Austin et al., "SimpleScalar: an infrastructure for computer system modeling", IEEE Computer, 35(2), 2002.
2) M. Berkelaar, "Statistical delay calculation, a linear time method", 6th Int. Workshop on Logic Synthesis, 1997.
3) K. Bernstein et al., "High-performance CMOS variability in the 65-nm regime and beyond", IBM Jour. of Res. and Dev., 50(4/5), 2006.
4) S. Borkar et al., "Parameter variations and impact on circuits and microarchitecture", 40th Design Automation Conf., 2003.
5) K. A. Bowman et al., "Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration", IEEE Jour. of Solid-State Circuits, 37(2), 2002.
6) M. Hashimoto et al., "Increase in delay uncertainty by performance optimization", Int. Symp. on Circuits and Systems, 2001.
7) H. Li et al., "SAVS: a self-adaptive variable supply-voltage technique for process-tolerant and power-efficient multi-issue superscalar processor design", 11th Asia and South Pacific Design Automation Conf., 2006.
8) S. Vassiliadis et al., "Interlock collapsing ALU's", IEEE Trans. on Comput., 42(7), 1993.
9) 小野寺 秀俊, "ばらつきを克服する設計技術", 回路とシステム軽井沢ワークショップ, 2006.
10) 佐藤 寿倫, "命令レベル逐次プロセッサ", 情処研報, 2006-ARC-169, 2006.