計算機アーキテクチャ 126－18
（1997．10．29）

非同期式プロセッサ TITAC-2 の性能解析

小沢基一[†] 高村明裕[†]
上野洋一郎[†] 南谷崇[††]

非同期式プロセッサ TITAC-2 で用いているような非同期式パイプラインは同期式と異なり，各ステージの処理遅延が入力データにより変動する．その結果，ステージ間ラッチの読み書きの際に同期式では存在しないオーバーヘッドがかかる．そのため，このようなオーバーヘッドを無視して単純に各ステージの処理遅延のみで性能評価をするのは好ましくない．そこで，非同期式パイプラインのオーバーヘッドの考察と，オーバーヘッドを考慮した TITAC-2 の性能評価について述べる．

Performance Analysis for Asynchronous Processor TITAC-2

Motokazu Ozawa,[†] Akihiro Takamura,[†] Yoichiro Ueno[†]
and Takashi Nanya[††]

In an asynchronous pipeline system, processing delay for each stage changes by the input value. Because of this property, the asynchronous pipeline system have more latch read/write overhead than the synchronous one. If we don't give consideration to this nature, the performance analysis of the system may become inaccurate. In this paper, we describe the result of TITAC-2's performance analysis taking into account to this nature.

1. はじめに

デバイス技術の進歩によりスイッチング速度が数 ps という素子の実現が可能になりつつある．また，デザインルールが微細化する一方，機能の向上によるダイサイズの拡大も進んでいる．このような環境において VLSI システムを設計する場合には配線遅延の方が素子遅延よりも支配的になる．この結果，現行の同期式システムは，クロックスキューの制約により，素子遅延に見合う高速性を発揮できない[1)]．

このような問題を解決するために，我々は回路全体を同期させるクロック信号を用いない非同期式システム設計に関する研究を行っている．この研究の一環として，非同期式プロセッサ TITAC-2[2)3)] の試作を行った．TITAC-2 は，現行の同期式パイプラインを非同期化することで実現されている．しかし，非同期式パイプラインでは，同期式パイプラインに存在しないオーバーヘッドがかかるため同期式で用いられる性能評価手法が適用できない．そこで，本稿では，非同期式パイプラインのオーバーヘッドを考慮した TITAC-2 の性能評価について報告する．

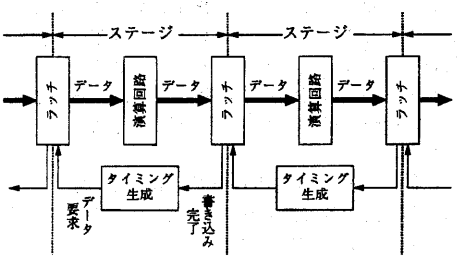

図1 非同期式パイプラインの構成

2. 非同期式パイプラインの動作

非同期式パイプラインは，一般に図1のような構成である．非同期式パイプラインの各ステージは以下のような動作を自律的に行う．

(1) 入力側のラッチにデータ出力を要求する．
(2) データの処理を行う．
(3) 出力側のラッチにデータを書き込む．
(4) データ処理と書き込みが終了するまで待つ．

非同期式パイプラインは，データ処理遅延の取扱い方により次の 2 種類に分類できる．

- データの処理遅延がある一定値より小さいと仮定する．

† 東京工業大学情報理工学研究科
Graduate School of Information Science and Engineering, Tokyo Institute of Technology
†† 東京大学先端科学技術研究センター
Research Center for Advanced Science and Technology, University of Tokyo

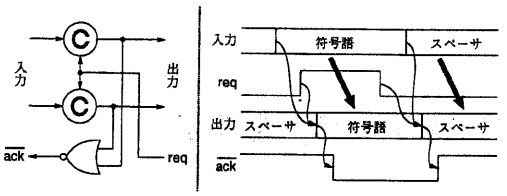

図2　C-latch の構成と動作

仮定する遅延をデータ処理にかかる最悪遅延とする必要がある．そのため，データ処理時間が回路の最悪遅延で決まる．

- データ処理の終了を何らかの手法で検出する．データ処理が終了してすぐに次の処理を行うことができる．そのため，データ処理時間が回路の平均遅延で決まる．

一般に平均遅延の方が最悪遅延より小さいため，後者の方がより高速に動作できる．そのため，TITAC-2 では後者の構成手法である C-latch を用いた非同期式パイプラインを採用している．

3. C-latch による非同期式パイプライン

図2 のような回路☆を C-latch と呼ぶ．C-latch は，1 bit のデータを 2 線符号化し，実際のデータである符号語と非符号語であるスペーサを交互に送る 2 線 2 相方式のもとで容量 1 の FIFO となる．この C-latch を複数段接続してステージ間ラッチにすると**図3** のような非同期式パイプラインを構成することができる[4]．ここで，このパイプラインの動作を直観的に理解するために次のようなモデル化を行う．このモデル化によりパイプラインが図 3 下側のように表現される．なお，区別が必要ない場合には ● ○ をまとめてトークンと呼ぶ．

- 各ステージの区切りを長い線で表す．
- C-latch の入力側が符号語，出力側がスペーサである状態を ● とする．
- C-latch の入力側がスペーサ，出力側が符号語である状態を ○ とする．
- C-latch の入力側と出力側が同じ状態の場合を 空白 とする．
- 初期状態では各ステージに ● ○ が 1 つずつ存在する．
- ● ○ は交互に存在する．
- トークンは隣接する空白に移動することが出来る．
- トークンの場所に回路がある場合，隣接する空白への移動にその回路の遅延がかかる．

このモデル化により，次のような性質が導ける．

- C-latch を n 段接続すると 1 つのステージに最大で n 個のトークンが記憶できる．

☆ 回路図中の C は C 素子である．

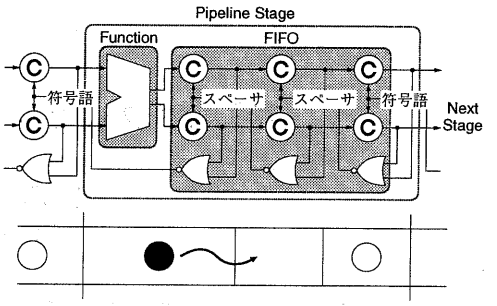

図3　FIFO によるパイプライン

- パイプラインにループがある場合，初期状態においてループ内に空白が最低 1 個ないとデータが移動できず，デッドロックを起こす．

デッドロックを起こさないためにはループ内に 1 個の空白が存在すればよい．しかし，ループ内の空白が 1 個の場合，同時に 1 個のトークンしか移動できない．パイプラインで最大の並列度を得るためには同時にステージ数と同じ数のトークンが移動できる必要がある．そのためには，各ステージが最低 1 個の空白を持つように FIFO を 3 段以上にする必要がある．

4. 非同期式パイプラインの性能

非同期式パイプラインの各ステージの平均サイクルタイム T はその動作から次のように表せる．

$$T = T_r + T_f + T_w + T_c$$

T_r ･･･FIFO 読み出し遅延の平均
T_f ･･･データ処理遅延の平均
T_w ･･･FIFO 書き込み遅延の平均
T_c ･･･読み出し要求の生成遅延の平均

サイクルタイム T のうち，実際のデータ処理時間は T_f である．残りの $T_r + T_w + T_c$ がパイプライン化によるオーバーヘッドである．

非同期式パイプラインでは，各ステージの平均サイクルタイム T が等しくなることが知られている[5]．そのため，T_f の最も大きいステージが全体の性能を決定する．

4.1 サイクルタイムの下限

各ステージの平均サイクルタイム T は FIFO の状態が次のような時に下限となる．

T_r が下限となる条件

データ要求の時，入力側の FIFO に必ず 1 個以上のトークンがある．

T_w が下限となる条件

データ書き込みの時，出力側の FIFO に必ず 1 個以上の空白がある．

4.2 FIFO 段数とサイクルタイムの関係

3 節のような非同期式パイプラインは，入力データ

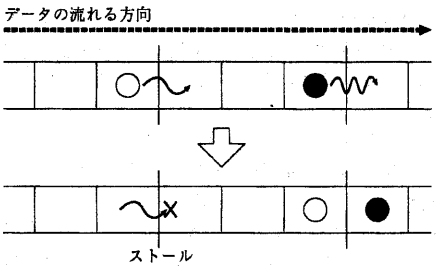

図4　FIFO が空になってしまう場合

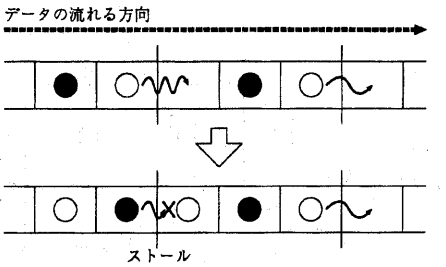

図5　FIFO が一杯になってしまう場合

によってデータ処理遅延が変動する．データ処理遅延が変動すると次のような状態が起こり，サイクルタイムの低下が起こる．

読み出し側の FIFO が空になってしまう場合

自ステージが速くなったり，先行ステージが遅くなると図4のようにデータが読み出せなくなる．この結果，FIFO 読み出し遅延の T_r が大きくなる．

書き込み側の FIFO が一杯になってしまう場合

自ステージが速くなったり，後続ステージが遅くなると図5のようにデータが書き込めなくなる．この結果，FIFO 書き込みの遅延 T_w が大きくなる．

このような状態はデータ処理遅延の変動が大きいほど起こりやすくなる．このサイクルタイムの増加を抑制するための手法として FIFO 段数の増減がある．FIFO 段数の増減により T_r, T_w が次のように変化する．

FIFO 段数を増加

FIFO が一杯になりにくくなるので T_w が減少する．しかし，データ読み出しの時にトークンの移動距離が増え，T_r が増加する．

FIFO 段数を減少

データ読み出しの時にトークンの移動距離が減るので T_r が減少する．しかし，FIFO が一杯になりやすくなり，T_w が増加する．

以上の結果から，最適な FIFO 段数の存在が予想される．しかし，最適な FIFO 段数を求めるには各ステージのデータ処理遅延の変動を考慮する必要があり難しい．TITAC-2 ではシミュレーションによる評価で FIFO 段数を決定している．

なお，初期化は各ステージに ● ○ が 1 個づつ存在するように行われる．その結果 n ステージで構成されたループでは $2n$ 段より多い FIFO は必要とならない．

4.3　データ処理遅延とサイクルタイムの関係

あるステージのデータ処理遅延を何らかの手法で小さくすると，そのステージの T_r, T_w が以下のように変化する．

T_r の変化

データ読み出しの要求間隔がより短くなる．その結果，FIFO が空になりやすくなり，T_r が増加する．

T_w の変化

書き込みデータの到着間隔がより短くなる．その結果，FIFO があふれやすくなり，T_w が増加する．

この結果，あるステージのデータ処理遅延を小さくしても，T_r, T_w が増加し，サイクルタイムがそれほど減少しないことがある．しかし，データ処理遅延を小さくしたステージの前後では，次のような変化が起こりサイクルタイムが減少することがある．

先行ステージ

FIFO があふれにくくなるため，T_w が減少する．

後続ステージ

FIFO が空になりにくくなるため，T_r が減少する．

4.4　複雑なデータパスのサイクルタイム

実際のシステムでは，1 つのステージ内に複数の演算回路が含まれたり，回路に複数の入力や書き込み先が存在することがある．この場合，データ処理遅延に関係なく正常な動作を行うためには次のような制御が必要となる．

データ処理終了の検出

データ処理終了と結果の書き込み終了を確認する．ステージに含まれる回路ごとに行う．

データ要求信号の生成

入力を共用する演算回路全てのデータ処理終了が確認されてから次のデータを要求する．

この場合のサイクルタイム T は次のように求められる．

(1)　各演算回路について「入力側 FIFO ⇒ 演算回路 ⇒ 出力側 FIFO ⇒ 制御回路 ⇒ 入力側 FIFO」というパスを全て抽出する．抽出したパスそれぞれについて $T_r + T_f + T_w + T_c$ を計算し，その最大値を各演算回路のサイクルタイムとする．

(2)　各演算回路のサイクルタイムの最大値がそのステージのサイクルタイムとなる．

入力データによりデータ処理遅延や FIFO 読み書き遅延が変動するため，各演算回路のサイクルタイムも変動してしまう．その結果，正確なサイクルタイムを

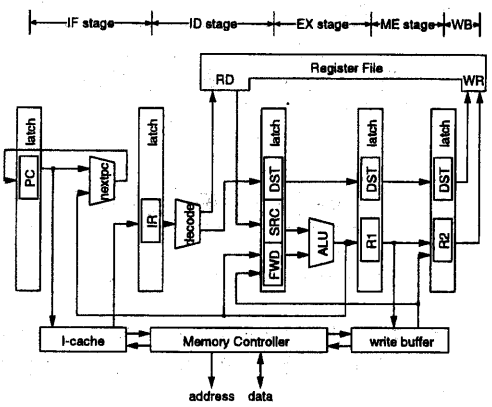

図6　TITAC–2 の構成

求めるには前述の手順を毎サイクル行う必要がある．

この制御方式を用いた場合，各ステージのサイクルタイムがステージ内の最も遅いパスにより決定される．そのため，各ステージのサイクルタイムがステージ内で同時に動作するパスの影響を受ける．その結果，5.6 節のようなステージ構成による性能変化が起こる．

なお，TITAC–2 の演算回路には一部の入力のみで出力が決まるものがある．TITAC–2 ではこのような演算回路についても正常な動作を行うために次のような待ち合わせを行っている．

(1)　演算回路の出力側 FIFO 全ての書き込み完了
(2)　演算回路の入力側 FIFO 全ての読み出し完了

この結果，$T_f + T_w$ が (2) の生成にかかる時間より小さくなることはできない．TITAC–2 の場合，(2) の生成にかかる遅延は約 2.2 ns である．

5. TITAC–2 の性能解析

4 節で述べた性能変化に着目して TITAC–2 の性能評価を行う．

5.1 TITAC–2 の構成

TITAC–2 は，非同期式による実用レベルのプロセッサの実現を目的に設計，試作されたものである．TITAC–2 の構成は図 6 である．また，その概要は次のようになっている．なお，実際の試作では FIFO 4 段によるパイプラインを採用している．

- 32bit のアドレスバス / データバス
- MIPS R2000 準拠の命令セット（FPU なし）
- IF, ID, EX, ME, WB の 5 段パイプライン
- 8KB のオンチップ命令キャッシュ
- 例外処理 / 外部割り込みの提供
- 3.3 V で約 54 VAX MIPS（実測）

5.2 性能評価の条件

性能評価の対象には TITAC–2 のゲートレベル記述を用いた．この記述ではレイアウト面積の予測から計算した遅延を用いている☆．この場合の T_r の下限は 0.42 ns, T_w の下限は 0.82 ns である．したがって，TITAC–2 におけるサイクルタイムの下限は $1.24 + T_f + T_c$ ns となる．

評価は論理シミュレーションにより T_r, T_w, T_f, T_c, T を求めて行った．評価に使用するプログラムとしては dhrystone と noploop を用いた．

dhrystone：　dhrystone version 2.1 である．命令キャッシュを有効にした状態で 10 回実行した．総実行命令数は 5678 である．多くの種類の命令を実行するため，データ処理遅延にばらつきが大きい．

noploop：　nop 命令 16 個を命令キャッシュを有効にしてループさせたものである．総実行命令数は 777 である．実行する命令の種類が少ないため，データ処理遅延のばらつきが小さい．

4.4 節より正確なサイクルタイムを求めるためには各命令ごとのサイクルタイムを計算する必要がある．しかし，今回の評価では「ステージ内で最大の平均遅延を取るパスが常に最大のサイクルタイムを取る」と仮定した．この場合，各ステージのサイクルタイムが実際のものよりやや小さくなる．そこで，プログラムの実行終了時間から計算したサイクルタイムも求めた．なお，TITAC–2 のパイプラインは 2 線 2 相方式であるため，1 命令の平均実行時間は求めたサイクルタイムの 2 倍となる．

5.3 パイプラインのオーバーヘッド

実際の TITAC–2 のシミュレーション結果を 図 7, 図 8 に示す．図 7, 図 8 の太線は各ステージのサイクルタイム決定パスを示す．4 節で述べたように，非同期式パイプラインのサイクルタイムには $T_r + T_w + T_c$ というオーバーヘッドが含まれる．そこで，各ステージでの T_f とオーバーヘッドを求めた結果を 表 1 に示す．

この結果を見ると TITAC–2 の持つオーバーヘッドがかなり大きいことが分かる．この原因は T_f が最

表1　TITAC–2 のオーバーヘッド

dhrystone

ステージ	T(ns)	T_f	オーバーヘッド (%)
IF	8.98	4.35	51.6
ID/WB	8.27	3.70	55.3
EX	8.67	3.68	57.6
ME	8.82	4.93	44.1

noploop

ステージ	T(ns)	T_f	オーバーヘッド (%)
IF	6.93	4.24	38.8
ID/WB	6.84	3.51	48.9
EX	6.62	3.59	45.8
ME	6.87	1.04	84.9

☆ この場合の性能は 62.5 VAX MIPS である．

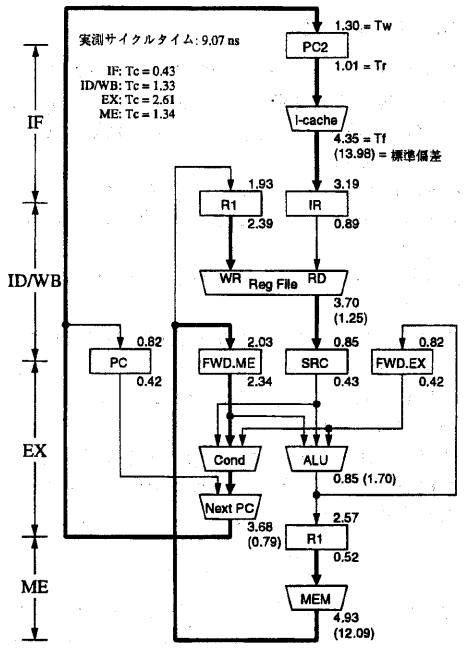

図7 dhrystone の評価結果

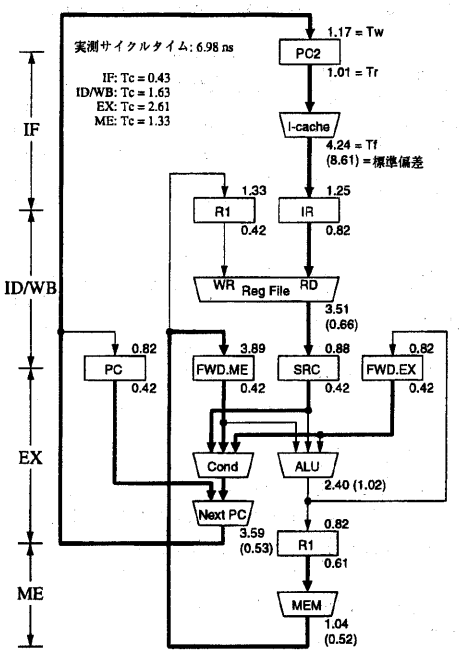

図8 noploop の評価結果

も大きいステージ（dhrystone では ME, noploop では IF）の遅延変動が大きいことである．その結果，そのステージの T_r, T_w が大きくなり全体の性能を制約する．また noploop と dhrystone を比較すると ME ステージを除いて noploop のオーバーヘッドの方が小さい．これは noploop が実行する命令の種類が少なく，各ステージの遅延変動が小さいためである．なお，noploop の ME ステージが持つオーバーヘッドが大きいのは EX ステージよりも大幅に T_f が小さく，FWD.ME の FIFO が常にあふれるためである．

5.4 FIFO 段数による性能変化

4.2 節で述べた FIFO 段数と性能の関係を調べるために，各ステージの FIFO 段数を 3, 4, 5 と変化させた．その時のサイクルタイム，T_f の最も大きいステージ（dhrystone では ME, noploop では IF）でのオーバーヘッドを表 **2** に示す．

この結果より FIFO 段数を増やすと T_w が減少，T_r が増加すること，FIFO 段数 4 のオーバーヘッドが最も少ないことが分かる．FIFO 段数 4 が最適な理由として，TITAC-2 が最大でも 3 ステージの小さなループの組合せにより構成されていることが考えられる．また，T_r, T_w の FIFO 段数 3 から 4 での変化と 4 から 5 での変化に差があるのは ME ステージのサイクルタイムを決めるパスが R1 → MEM → FWD.ME から R1 → MEM → R1 に移動するためである．

5.5 各ステージの遅延減少の効果

4.3 節で述べたような各ステージの処理遅延の変化が性能に与える影響を調べるために各ステージに含まれる制御回路の遅延（T_c）を小さくした．その結果のうち ΔT_c, ΔT_c と ΔT の比，変化させたステージの $\Delta(T_r + T_w)$ を表 **3** に示す．

この結果より，あるステージのデータ処理のみを高速化しても T_r, T_w の増加によって性能向上が制限されることが分かる．しかし，全てのステージをほぼ等しく高速化すれば全体のサイクルタイムも各ステージで減少しただけ小さくなる．

表 **2** FIFO 段数と性能の関係

dhrystone

段数	T(ns)	T_r(ns)	T_w(ns)	オーバーヘッド（%）
3	9.74	0.42	3.03	50.2
4	9.07	0.52	2.03	44.1
5	9.08	1.32	1.47	46.6

noploop

段数	T(ns)	T_r(ns)	T_w(ns)	オーバーヘッド（%）
3	7.44	0.42	2.28	42.1
4	6.98	1.01	1.25	38.8
5	6.97	1.33	0.96	39.2

5.6 ステージ構成の違いによる性能変化

TITAC-2 の icache はその遅延が数 100 ns から 1 ns の範囲で変動する．icache の入力である PC2 は図 **9** のようにすると削除できる．この構成でも同じ動作をするが，EX ステージの制御手法が次のように変化する．

表3　データ処理遅延の変化による性能の変化

dhrystone

対象	ΔT_c(ns)	$\Delta T/\Delta T_c$(%)	$\Delta(T_r+T_w)$ (ns)
IF	-0.43	0.19	0.44
ID/WB	-1.63	0.02	1.13
EX	-1.18	0.08	0.77
ME	-1.34	0.11	0.49
全部	—	$\Delta T=-0.96$	-0.69

noploop

対象	ΔT_c(ns)	$\Delta T/\Delta T_c$(%)	$\Delta(T_r+T_w)$ (ns)
IF	-0.43	0.21	0.28
ID/WB	-1.63	0.00	0.98
EX	-1.18	0.08	0
ME	-1.13	0.01	0.25
全部	—	$\Delta T=-1.12$	-0.74

PC2 がない場合

ALU, nextpc, icache の処理終了と PC, IRPC の書き込み終了を待ち合わせて PC を読み出す．そのため，EX ステージの遅延変動が icache の影響で大きくなり，icache の遅延の下限が nextpc, ALU の遅延で制約される．

PC2 がある場合

EX ステージ, nextpc の処理終了と PC, PC2 の書き込み終了を待ち合わせて PC を読み出す．そのため，EX ステージの遅延変動が icache の影響を受けず，icache が遅延の下限で動作できる．

PC2 の有無による性能の変化を表 4 に示す．この結果では dhrystone, noploop ともに PC2 ありの方が高性能になっている．この理由として，dhrystone では EX ステージの遅延変動減少により ME ステージの T_w が減少したこと，noploop では ID ステージの T_r が減少していることから icache がその遅延の下限で動作可能になり，IR を生成するステージの T_f が減少したことが考えられる．

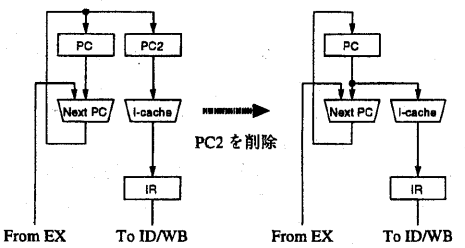

図9　PC2 の削除

表4　ステージ構成による性能の違い (ns)

dhrystone			noploop		
PC2	T	ME T_r	PC2	T	ID T_r
なし	9.42	2.80	なし	7.91	1.56
あり	9.07	2.03	あり	6.98	0.82

6. ま と め

非同期式パイプラインの性能変化をそのオーバーヘッドに着目して考察し，オーバーヘッドを考慮した TITAC-2 の性能評価を行った．その結果，次のようなことが分かった．

- FIFO 段数には最適値が存在すること．
- 特定ステージのみの高速化では性能向上の効果が小さいこと．
- 動作が同じでもステージ構成により性能が変化すること．

今後の課題として，FIFO それぞれの段数を最適化する手法や高性能を得ることができるステージ構成手法の検討，T_r, T_w と演算遅延のばらつきの関係の調査などがあげられる。

謝辞　本研究の一部は科研費補助金基盤研究（B）09480049，及び（株）半導体理工学研究センターとの共同研究によるものである．

参 考 文 献

1) 南谷崇: 非同期式プロセッサ — 超高速 VLSI システムを目指して —, 情報処理, Vol. 34, No. 1, pp. 72–80 (1993).
2) Takamura, A., Kuwako, M., Imai, M., Fujii, T., Ozawa, M., Fukasaku, I., Ueno, Y. and Nanya, T.: TITAC-2 : An asynchronous 32-bit microprocessor based on Scalable-Delay-Insensitive model, *Proc. International Conf. Computer Design (ICCD'97)* (1997).
3) Nanya, T., Takamura, A., Kuwako, M., Imai, M., Fujii, T., Ozawa, M., Fukasaku, I., Ueno, Y., Okamoto, F., Fujimoto, H., Fujita, O., Yamashina, M. and Fukuma, M.: TITAC-2 : A 32-bit Scalable-Delay-Insensitive Microprocessor, *HOT CHIPS IX*, Stanford, pp. 19–32 (1997).
4) Sparsø, J., Staunstrup, J. and Dantzer-Sørensen, M.: Design of delay insensitive circuits using multi-ring structures, *Proc. European Design Automation Conference (EURO-DAC)*, Hamburg, Germany, IEEE Computer Society Press, pp. 15–20 (1992).
5) Kearney, D. and Bermann, N. W.: Performance Evaluation of Asynchronous Logic Pipelines with Data Dependant Processing Delays, *Asynchronous Design Methodologies*, IEEE Computer Society Press, pp.4–13 (1995).
6) 小沢基一, 高村明裕, 上野洋一郎, 南谷崇: 非同期式パイプラインの動作解析, 情報処理学会第 54 回全国大会 (1997).