非同期式パイプラインプロセッサの高性能化手法について

3N－2

小沢基一　高村明裕　上野洋一郎　中村宏†　南谷崇†
東京工業大学 情報理工学研究科　†東京大学 先端科学技術研究センター

1　はじめに

デバイス技術の発達により、非常に高速な素子が実用化されつつある。このような素子を用いて論理回路を設計する場合、従来の同期式では配線遅延の増加によりクロック分配が困難になると予測されている。そこで、クロック分配が不要な非同期式論理回路が注目されている。

非同期式でパイプラインを構成する場合、要求応答制御のオーバーヘッドや入力による演算遅延の変動により性能が低下する [1]。そこで、要求応答制御のオーバーヘッドによる性能低下をステージが持つデータ数の最適化で抑制することが提案されている [2]。本稿では、この手法が演算遅延の変動による性能低下も抑制することを示す。また、非同期式プロセッサ TITAC-2[3] にこの手法を適用した場合の効果の予測を行う。

2　非同期式パイプラインの構成

非同期式パイプラインは、図 1 のように構成される。この構成では以下のような動作が繰り返し行われる。

動作 1 (演算パス)：演算を実行 → 結果をラッチに書く

動作 2 (要求応答パス)：書き込み終了 → データを要求

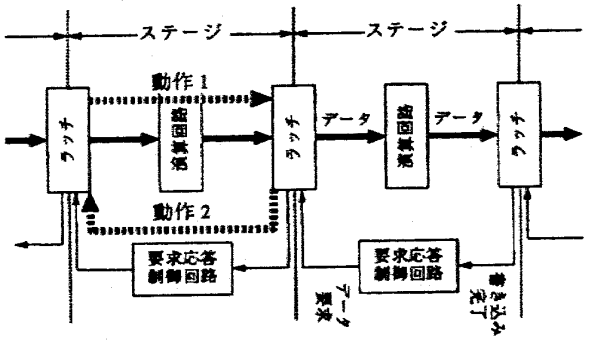


図 1: 非同期式パイプラインの構成

本稿では、実際の演算を行う稼働相と回路を初期化する休止相を交互に実行する 2 線 2 相方式のパイプラインについて考える。2 線 2 相方式で非同期式パイプラインを構成する手法として C ラッチを用いるもの [4] がある。C ラッチとは図 2 中の点線で囲まれた回路で、次のような入出力の同期動作を行う。

- req が 1(0) の時に入力が符号語 (スペーサ) なら、その入力が記憶される。
- 入力が記憶されると ack が遷移する。

この C ラッチを図 2 のように m 段直列に接続すると、稼働相と休止相が合わせて $m-1$ 個入る FIFO を構成できる。この FIFO をステージ間ラッチとすることで非同期式パイプラインが構成できる [3]。

A method for performance improvement of asynchronous pipeline processor
Motokazu Ozawa, Akihiro Takamura, Yoichiro Ueno
Tokyo Institute of Technology, Graduate School of Information Science and Engineering
Hiroshi Nakamura, Takashi Nanya
University of Tokyo, Research Center for Advanced Science and Technology

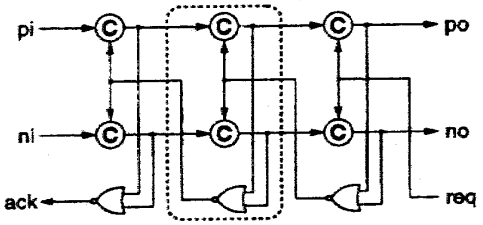


図 2: C ラッチによる FIFO

この構成では、稼働相と休止相という 2 種類の演算が交互に行われる。しかし、どちらの相でもパイプラインの動作に差はない。そこで、今後の議論では稼働相と休止相を区別せずにデータと記述する。なお、実際の処理にかかる時間は 1 データを処理する時間の 2 倍となる。

3　1 ステージの持つデータ数と性能の関係

2 節のパイプラインでは、初期化時に 1 ステージが持つデータ数を $m-1$ 個以下で自由に設定できる。そこで、各ステージが平均的に持つデータ数 n (全体のデータ数/ステージ数) と性能の関係について以下のような条件でシミュレーションを行った。その結果を表 1, 2 に示す。この表中の太字で示した部分が最適な n を示す。

- パイプラインの構造は、全体で平均 20 ns, 10 ns, 5 ns の演算遅延がかかるループとする。
- ステージ間ラッチは図 2 のような FIFO とする。FIFO の遅延は TITAC-2 と同じにする。
- パイプライン全体でデータを 2 個 (稼働相と休止相を 1 個ずつ) 置く。
- 動作 2 にかかる遅延は一定 (2.65 ns) とする。
- 演算を分割することで n を変化させる。分割法として、均等な分割と 4:1 の不均等な分割を用いる。
- 演算の分割は容量 1 の FIFO の挿入で行う。なお、分割数が 1 の場合の動作を補償するため、分割数と無関係に容量 3 の FIFO が 1 個挿入される。
- 分割後の各演算には $\pm k$ % の一様な遅延変動を与える。
- サイクルタイムは 1 ステージがデータ 1 個を処理する時間とする。

表 1, 2 では $n=1$ を境にして傾向が変化している。そこで、n の変化に伴う動作の変化からこの結果を考察する。

3.1　1 ステージ内にデータが 1 個以上ある場合

この場合、全ステージが同時にデータを処理しており、各ステージの演算 (動作 1) と要求応答制御の動作 (動作 2) が交互に行われる。そのため、$n=1$ である 1 ステージのループ構造パイプラインでは図 3(i) のような動作をする。このときのサイクルタイムは以下のようになる。

動作 1 の遅延 + 動作 2 の遅延

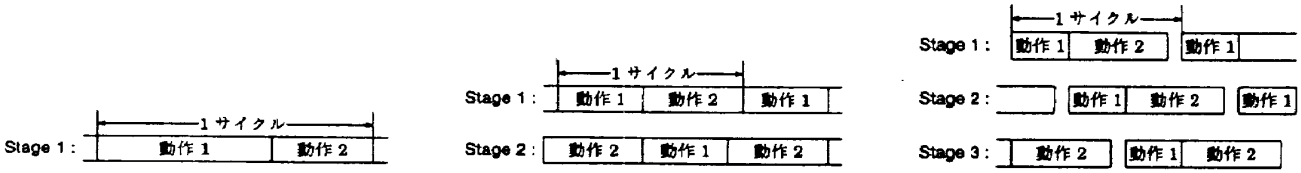
ステージの遅延が変動すると、ラッチがあふれたり、空になったりする。その結果、ラッチを読み書きできない時間 (図 3(i) では動作 1 と動作 2 の間に現れる) が生じ、性能が低下する。このラッチを読み書きできない時間はステージの遅延変動が大きいほど長くなる。この時、全体のデータ数を越えない範囲でラッチの容量を増やすとラッチの書き込みできない時間が短くなり性能が向上する。ラッチ容量が大きく書き込みが必ず可能ならば、読み出せない状態

表 1: 1 ステージあたりのデータ数 n, 遅延変動率 k (%) とサイクルタイム (ns) の関係 (均等分割)

n	全体の遅延 ··· 20 ns				全体の遅延 ··· 10 ns				全体の遅延 ··· 5 ns			
	$k=10$	$k=30$	$k=50$	$k=70$	$k=10$	$k=30$	$k=50$	$k=70$	$k=10$	$k=30$	$k=50$	$k=70$
2	23.09	23.13	23.09	23.06	13.08	13.12	13.08	13.14	8.08	8.08	8.09	8.09
1	13.44	14.13	14.78	15.47	8.27	8.60	8.95	9.28	5.69	5.86	6.02	6.19
2/3	**11.22**	11.57	12.11	12.67	6.60	6.91	7.23	7.57	4.86	5.01	5.16	5.31
1/2	11.45	**11.56**	**11.84**	12.20	**6.46**	**6.56**	**6.77**	7.03	4.45	4.58	4.71	4.84
2/5	11.70	11.73	11.88	**12.15**	6.70	6.72	6.82	**6.99**	**4.22**	**4.33**	**4.46**	4.59
1/3	11.93	11.93	12.06	12.23	6.92	6.92	6.97	7.07	4.43	4.43	4.49	**4.58**

表 2: 1 ステージあたりのデータ数 n, 遅延変動率 k (%) とサイクルタイム (ns) の関係 (4:1 に分割)

n	全体の遅延 ··· 20 ns				全体の遅延 ··· 10 ns				全体の遅延 ··· 5 ns			
	$k=10$	$k=30$	$k=50$	$k=70$	$k=10$	$k=30$	$k=50$	$k=70$	$k=10$	$k=30$	$k=50$	$k=70$
2	23.09	23.13	23.09	23.06	13.08	13.12	13.08	13.14	8.08	8.08	8.09	8.09
1	19.08	19.08	19.05	19.09	11.07	11.07	11.06	11.05	7.07	7.07	7.06	7.06
1/2	**11.45**	**11.56**	**11.84**	**12.20**	**6.46**	**6.56**	**6.77**	**7.03**	**4.45**	**4.58**	**4.71**	4.84
1/3	11.92	11.99	12.22	12.54	6.93	6.96	7.11	7.31	4.49	4.59	4.71	**4.82**



(i)$n = 1$ での動作　(ii)$n = 1/2$ での動作　(iii)$n = 1/3$ での動作

図 3: n による動作の違い

の起こる確率は n が大きいほど小さくなるため、遅延変動による性能低下が小さくなる。しかし、n に比例して 動作 1 の遅延が増加してしまうため、n が大きいほど性能が高くなるとは限らない。

各ステージの演算遅延が不均一な場合、全ステージが同時に動作することから、最も動作 1 の遅延が大きいステージで性能が決まる。このステージの状態は次のようになる。

入力側 : データが多くなり、入力側のラッチが空になる確率が小さくなる。

出力側 : データが少なくなり、出力側のラッチがあふれる確率が小さくなる。

この結果、ラッチの読み書きできない確率が小さくなり、遅延変動による性能低下が抑制される。しかし、動作 1 の遅延が増加するため、性能を高くできるとは限らない。

3.2 ステージの持つデータ数が 1 個未満の場合

この場合、全ステージがデータ処理を同時に行うことはない。そのため、あるステージの動作 1 の後、そのステージの 動作 2 と後続ステージの 動作 1 が同時に動作できる。その結果、図 3(ii),(iii) のように 動作 2 の遅延を隠蔽できる。この場合、隠蔽できる遅延量とサイクルタイムが以下のように表せる。

サイクルタイム ＝ 動作 1 の遅延/n + 隠蔽できない動作 2 の遅延

隠蔽できる遅延は n を小さくするほど図 3 のように増加する。しかし、現実には処理の分割用にラッチが挿入される。その結果、ラッチの通過遅延が 動作 1 の遅延に加わるため、n にある最適点が存在する。最適な n は 動作 2 の遅延が一定ならば、演算遅延全体が小さい (大きい) ほど小さく (大きく) なる。

この場合、ステージの遅延が変動してもデータが平均 $1/n$ ステージおきにしか存在しないため、ラッチを読み書きできない確率が $n = 1$ と比べて小さくなる。その結果、n が小さくなるほど遅延変動による性能低下が抑制される。

各ステージの演算遅延が不均一な場合、全体の性能が $1/n$ ステージ分の 動作 1 の遅延の和で決まるため、性能の変化が小さい。

4 TITAC-2 に適用した場合の性能予測

TITAC-2 では $n = 2$ のパイプラインを採用している。しかし、3節の結果から n を小さくすることで性能を向上させることができると考えられる。dhrystone benchmark を動作させている TITAC-2 では 動作 1 の平均遅延が約 5 ns、動作 2 の平均遅延が約 3 ns である [1]。そこで、動作 1 と 動作 2 の遅延がほぼ等しい表 1 の結果を見ると、最適な n は 2/5 である。この時、サイクルタイムは 4.3 から 4.5 ns となっている。TITAC-2 が 2 相方式であることから、実際の命令処理サイクルは約 8.6 から 9 ns と予想される。$n = 2$ の TITAC-2 では、命令処理サイクルが約 18.2 ns [1] なので、約 2 倍の性能向上が見込まれる。

5 まとめ

各ステージが持つデータ数と性能の関係をシミュレーションにより評価し、その結果を考察した。その結果、各ステージの持つデータ数を小さくすることで要求応答制御のオーバーヘッドのみでなく、遅延変動による性能低下も抑制できることを示した。また、TITAC-2 にこの手法を適用することで約 2 倍の性能が得られることを予測した。

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] 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 の性能解析. 情処研報, 第 97 巻, pp. 103–108, October 1997.

[2] A.Martin, A.Lines, R.Manohar, M.Nystroem, P.Penzes, R.Southworth, U.Cummings. The design of an asynchronous MIPS R3000 microprocessor. In *Advanced Research in VLSI*, September 1997.

[3] A.Takamura, M.Kuwako, M.Imai, T.Fujii, M.Ozawa, I.Fukasaku, Y.Ueno, T.Nanya. TITAC-2: An asynchronous 32-bit microprocessor based on scalable-delay-insensitive model. In *Proc. ICCD*, pp. 288–294, October 1997.

[4] J.Sparsø, J.Staunstrup, M.Dantzer-Sørensen. Design of delay insensitive circuits using multi-ring structures. In *Proc. EURO-DAC*, pp. 15–20, September 1992.