非同期式パイプライン高速化のためのステージ間ラッチの構成

3N－3

桑子 雅史　小沢 基一†　中村 宏　南谷 崇

東京大学 先端科学技術研究センター　†東京工業大学 情報理工学研究科

1 はじめに

近年、高速なスイッチングデバイスの素子性能を十分に享受する一手法としてクロック信号を用いない非同期式論理設計が注目されている [1]。プロセッサのパイプライン化は、同期式と同様、非同期式設計においても高速化に有効である。高速に動作する非同期式パイプラインを実現する手法として、パイプラインステージを非常に細かく分割する Deep-pipeline と呼ばれる手法が提案されている。この手法を採用するとき、各ステージの休止相を自律的に実行する機構をステージ間ラッチに組み込むことで、休止相に要する時間を隠蔽しパイプラインを高速化する回路構成を提案する。

2 2線2相式データ転送

クロックのない非同期式回路でデータ転送を実現するためには、データの到着を知るためにデータに時間情報を付加する必要がある。そのようなデータ転送方式の一つが2線2相式 [1] である。

この方式では、1 ビットのデータ Xは 2 本の信号線対 $(x,\bar{x})$ を用いて伝送される。この信号線対は有効符号語、スペーサと呼ぶ 2 種の状態をとる。有効符号語はデータを 2 線符号化して表わしたものであり、$(x,\bar{x}) = (0,1)$ は $X = 0$ を、$(x,\bar{x}) = (1,0)$ は $X = 1$ を表わす。スペーサは $(x,\bar{x}) = (0,0)$ の状態である。

回路はその環境から有効符号語を入力されると、演算などの処理を行なって、結果の有効符号語を出力する。これが稼働相である。そして次にスペーサが入力されるとスペーサを出力する。これが休止相である。この稼働相と休止相を交互に実行して演算を行なうことから、2線2相式と呼ばれる。

3 非同期式パイプライン

2線2相式のデータ表現による非同期式パイプラインを容易に構成する手法としてCラッチを用いる方式が知られている [2][3]。図 1の点線で囲んだ回路が 1 ビットのCラッチである。図中で©で表わされている素子は、Muller のC素子 [1] と呼ばれるものである。これは、全ての入力が 1 になると出力も 1 になり、全ての入力が 0 になると出力も 0 になるような記憶を持つ素子である。

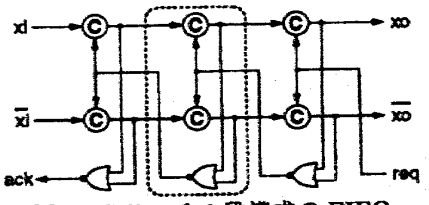

図 1: Cラッチ 3 段構成の FIFO

以下に示すように、Cラッチは入力側と出力側の同期動作を行なう。

1. req が 1 となり、かつ入力が有効符号語となると、その入力が記憶される。
2. 入力が記憶されると ack が 0 に遷移する。
3. req が 0 となり、かつ入力がスペーサとなると、スペーサが記憶される。
4. スペーサが記憶されると ack が 1 に遷移する。

このCラッチを図1のように n 段直列に接続すると、有効符号語とスペーサを合わせて n 個記憶できる FIFO を構成できる。32 ビット非同期式プロセッサ TITAC-2 [2][3] では、$n = 3$ または $n = 4$ の FIFO をステージ間ラッチとして、自律的に動作する 5 段の非同期式パイプラインを採用していた。

4 Deep-pipeline 方式と休止相の問題

非同期式パイプラインでは、その中に入れることのできるデータ数の上限値が存在する [2]。この上限値はパイプライン段数とステージ間 FIFO 段数によって決定される。TITAC-2 のパイプラインは、上限値に近い個数のデータを入れて動作するように設計されている。この方式を用いて 1 段の単純なループ構造のパイプラインを実現した場合、その動作は図 2に示すように演算回路の稼働相実行、ステージ間ラッチへの書き込み、要求-応答信号の伝播 (図中の R/A)、演算回路の休止相実行が逐次的に発生する。

これに対して、パイプライン段数を非常に多くして、上限値よりもはるかに少ない数のデータを入れて動作させる Deep-pipeline 方式が提案されている [4]。この方式を用いて図 2のパイプラインステージを 3 段に分割し同様の単純なループ構造を実現した場合、その動作は図 3に示すようになる。演算回路の動作と要求-応答信号の伝播が並行して発生するため、要求-応答信号の伝播遅延がある程度隠蔽されて高速となる [5]。

図 3のパイプライン動作では、各ステージの休止相の実行は前ステージからスペーサが伝播することで開始される。このため休止相の実行にも稼働相と同等の時間が必要となっている。

しかし、休止相は演算回路の初期化とも見なせる動作であり、前ステージからスペーサが伝播するのを必ずしも待つ必要はない。しかも、図 3の方式ではパイプラインに入っているデータが疎であることから、ステージ内が稼働相に安定した後、スペーサが入力されるまでには比較的長い時間的余裕がある。

従って Deep-pipeline 方式では、たとえ要求応答の遅延が増加したとしても、ステージが稼働相に安定したら入力によらず自律的に休止相を実行する機能をステージ間ラッチに持たせると、休止相の実行時間が他の処理の実行時間に隠蔽されて高速化できると考えられる。

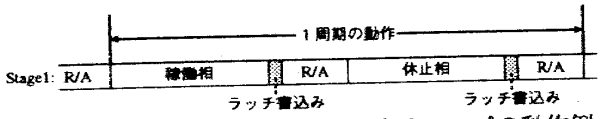

図 2: 1 ステージにデータが 1 個あるループの動作例

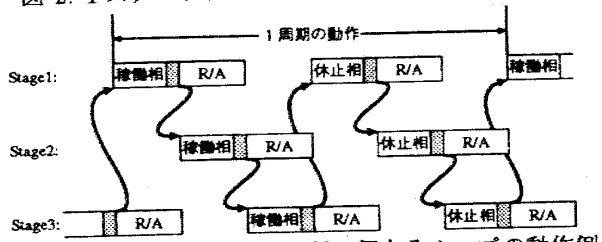

図 3: 3 ステージにデータが 1 個あるループの動作例

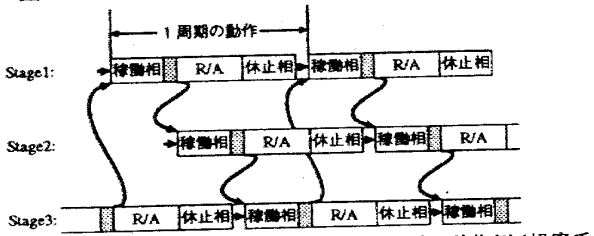

図 4: 3 ステージにデータが 1 個あるループの動作例 (提案手法)

A design of Stage-latch for Asynchronous Pipeline
Masashi Kuwako, Motokazu Ozawa†, Hiroshi Nakamura, Takashi Nanya
Research Center for Advanced Science and Technology, University of Tokyo　†Graduate School of Information Science and Engineering, Tokyo Institute of Technology

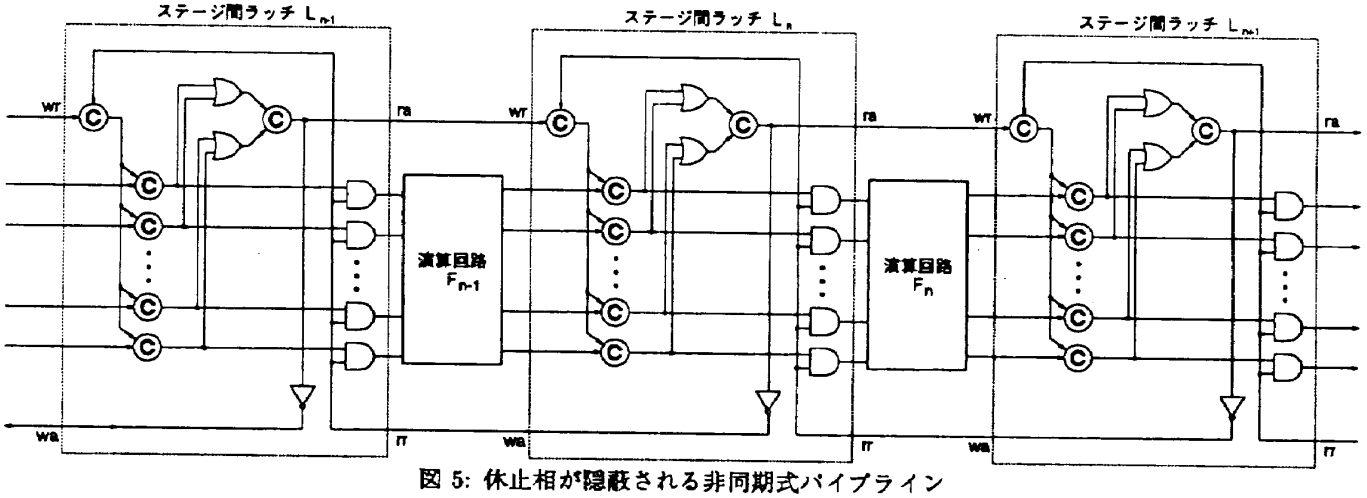

図 5: 休止相が隠蔽される非同期式パイプライン

5 休止相を隠蔽できるステージ間ラッチ

自律的に休止相を実行する機能を持ったステージ間ラッチとして図 5の構成を提案する。ステージ間ラッチ内の C ラッチはデータの記憶を行なう部分であり、AND ゲートは C ラッチ内のデータにかかわらず後段の休止相を実行する部分である。OR ゲートと C 素子で構成された回路部分は、C ラッチへの書き込み完了を確認するためのものである。

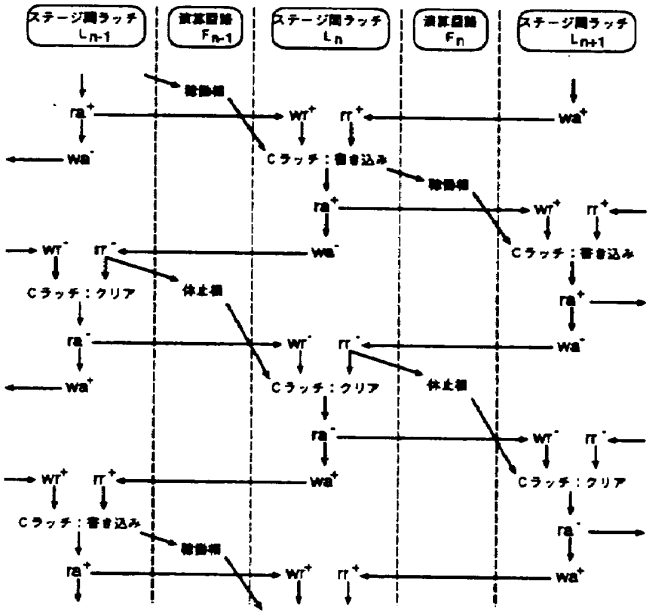

図 6: 図 5の動作

図 5の動作は図 6に示す。この図では変数 x の 0→1 遷移を x^+、1→0 遷移を x^- で表記している。

演算回路 F_{n-1} の稼働相の後、ステージ間ラッチ L_n 内の C ラッチへの書き込みが完了すると wa_n=0 となる。L_{n-1} は rr_{n-1}=0 を受けて、L_{n-1} 内の C ラッチの状態にかかわらず F_{n-1} の休止相を開始する。L_{n-1} 内の C ラッチがスペーサを記憶すると ra_{n-1}=0 となる。rr_n=0 となって、かつ L_{n+1} への有効符号語の書き込みが完了すると、L_n は C ラッチへスペーサを読み込む。そして、wa_n=1 を出力して、次の有効符号語を受け取る準備ができたことを前段に知らせる。

図 5の簡略化した動作は図 4のようになる。図 2、図 3に比較して動作の並列性が高くなることがわかる。

また、図 5に示したのは単純な直線構造のパイプラインであるが、分岐や合流のあるパイプラインであっても図 7、図 8に示すように容易に実現できる。

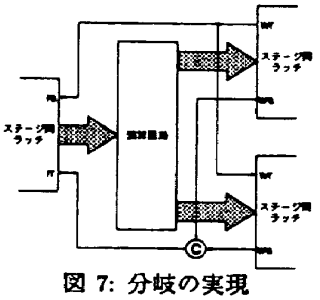

図 7: 分岐の実現

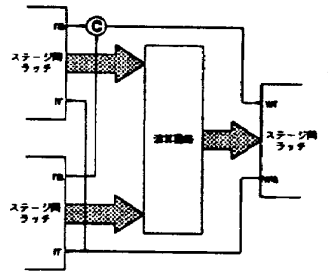

図 8: 合流の実現

6 まとめ

2 線 2 相式の Deep-pipeline を高速化する手法として、各ステージの休止相を自律的に実行することで休止相に要する時間を他の処理時間で隠蔽することのできるステージ間ラッチの一構成を提案した。

また、今回はステージ間ラッチをゲートレベルで構成したが、ステージ内の演算回路が MOS トランジスタレベルで設計されている場合、演算回路とステージ間ラッチを融合して最適化することでより高速な回路となる可能性がある。このような回路構成の検討は今後の課題である。

なお、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, January 1993.

[2] 高村明裕, 桑子雅史, 南谷崇. 非同期式プロセッサ TITAC-2 の論理設計における高速化手法. 信学論 (D–I), Vol. J80-D-I, No. 3, pp. 189–196, March 1997.

[3] Akihiro Takamura, Masashi Kuwako, Masashi Imai, Taro Fujii, Motokazu Ozawa, Izumi Fukasaku, Yoichiro Ueno, and Takashi Nanya. TITAC-2 : An asynchronous 32-bit microprocessor based on Scalable-Delay-Insensitive model. In *Proc. International Conf. Computer Design (ICCD'97)*, October 1997.

[4] A. J. Martin, A. Lines, R. Manohar, M. Nystrom, P. Penzes, R. Southworth, U. Cum mings, and T. K. Lee. The design of an asynchronous MIPS R3000 microprocessor. In *Proc. of Advanced Research in VLSI'97*, September 1997.

[5] 小沢基一, 高村明裕, 上野洋一郎, 中村宏, 南谷崇. 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. 情報処理学会第 56 回全国大会, March 1998.