1G－6

非同期式プロセッサ TITAC-2 の ALU の構成

藤井太郎　今井雅　池田吉朗　石田伯仁　西川慎哉　上野洋一郎　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

素子遅延に対する配線遅延の相対的な増加により、チップ全体にクロック信号を分配する同期式回路設計の限界が指摘されている。その解決策としてクロックを用いない非同期式回路が注目されている。

我々は、現在広く使用されている同期式プロセッサと同等の機能を持ち、高速動作することを目標に 32 ビット非同期式プロセッサ TITAC-2 を設計した。本稿では ALU の回路構成の工夫について述べる。

2　ALU の構成と高速化

TITAC-2 の ALU は、加算、減算、論理演算、シフト演算、LHI[1]、乗算、除算、素通しの 8 種類の回路が並列しており、制御信号によって選択された 1 つの演算回路のみが動作する。 設計方針として、使用頻度の高い加算器と減算器を中心に高速化を行った。

非同期式回路は事象の因果関係に基づいて動作する。そのため、ALU の速度向上には演算時間の短縮とともに、演算終了の検出を速くすることが重要となる。そこで、演算回路の速度向上として 2 線 2 相式 [2] の休止相を短縮する回路を追加、演算終了の検出速度向上として、出力データの到着とほぼ同時にその動作完了信号を出力する方式を実装した。これらを、加算器を例に用いて示す。また、DCVSL[1] を用いて設計した乗算器の構成を示す。

3　休止相の短縮

2 線 2 相式では演算を行なう稼働相と初期状態に戻す休止相が交互に存在し、両方が全体の動作時間に影響する。しかし入力されたデータに応じて演算を行う稼働相に対し、初期状態に戻す休止相は、対象となる回路内全体で一斉に行うことができる。

TITAC-2 の加算器と減算器は桁上げ先見方式 (Carry Look Ahead, CLA) を採用し、 4bit CLA 2 段で実現している (図 1)。稼働相における信号の伝播は図 1 の場合、全ての入力が 1bit 加算器から最上段の CLA へと順に伝わり、それにより Carry が下段へと伝わる。休止相における初期化も同様に下段から上段へと順に行われ、稼働相と同様の時間がかかってしまう。そこで休止相開始時に CLA へ直接初期化信号を入力し、上段の CLA を下段と並行して初期化するよう改善した。

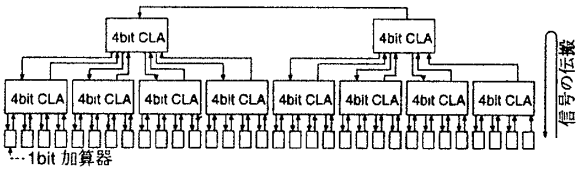

図 1: 32bit 加算器

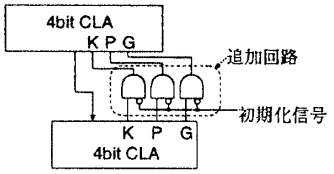

図 2: 休止相短縮回路

具体的には、CLA の 1 段目から 2 段目への信号線全てに初期化を開始するための回路を挿入した。この信号線は、桁上げの生成 (Generate)、伝播 (Propagate)、消滅 (Kill) の 3 本 1 組から成り、いずれか 1 本のみが稼働相において 1 に遷移する。ここに図 2 のような回路を追加することで、初期化信号の入力により直ちに休止相が開始される。初期化信号は演算選択信号の論理反転で実現できる。この方式は全体としてゲートの段数が最大 1 段増加するため、稼働相の処理時間が増加するが、休止相において CLA 1 段分以上短縮する。よって全体の処理時間は表 1 のように短縮する。

	稼働相	休止相	計
短縮前	3.62	3.62	7.24
短縮後	4.04	2.72	6.76

表 1: 休止相の短縮による平均処理時間の変化 (ns)

4　演算終了検出の高速化

桁上げ信号が伝播する加算器や減算器は、ビット毎に演算終了時刻が異なり、その順序も特定できない。このような場合、全出力信号の到着を監視することにより演算終了の検出を行うが、検出自体に時間がかか

ALU Design of Asynchronous Processor TITAC-2
Taro Fujii, Masashi Imai, Yoshiro Ikeda, Norihito Ishida, Shinya Nishikawa, Yoichiro Ueno
Graduate School of Information Science and Engineering, Tokyo Institute of Technology
Takashi Nanya
Research Center for Advanced Science and Technology, University of Tokyo

[1]Load HIgh。2 つの入力の下位 16bit を順に並べる演算

るため、演算終了時刻と検出終了時刻に大きな差が生じてしまう (図 3.(a))。そこで、演算回路の内部信号を用いた検出を行い、検出時間の大部分が演算時間に隠れるようにした (図 3.(b))。

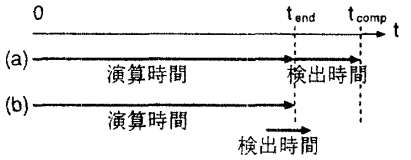

図 3: 演算終了の検出時間

具体的には、図 1 の加算器の場合、上段の CLA から下段の各 CLA へは桁上げ信号の肯定線 (Cp) と否定線 (Cn) の 2 本 1 組の線があり、どちらか一方が演算時に 1 に遷移する。これは桁上げの決定を表す信号であり、その遷移は演算結果がほぼ決まったことを示す。よって、この信号の到着を全て監視する (図 4) ことで、演算がほぼ終了したと確認できる。また監視位置の変更により、最終出力の位置では 32 個所の監視が必要であったのに対し、本手法では 7 個所と少なく、検出時間自体が短くなる。また、この位置の監視で、初期化終了の検出も行うことができる。

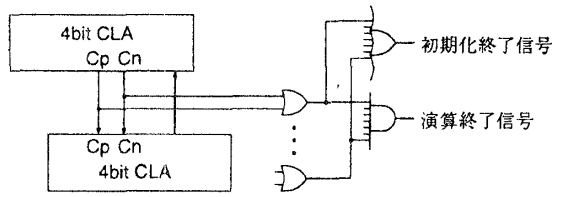

図 4: 演算終了の検出回路

ただしこのような内部信号による検出では、出力位置まで全ての信号が到着し、演算が終了したことを直接示してはいない。そこで、全ビットの演算終了時刻を t_{end}、演算終了信号の遷移時刻を t_{comp} とした時、仮定する遅延変動が生じても $t_{comp} - t_{end} > 0$ が常に成立することを示す必要がある。

先程の加算器の場合、監視位置から演算結果出力までのゲート段数は、下段の CLA 内で最大 2 段、1bit 加算器内で 2 段であるが、ALU として TITAC-2 に組み込む際、各演算回路の出力をまとめるため更に一定段数 (m 段) 必要となる。ゆえに、演算結果は監視位置から最大 4+m 段のゲートを通過し、ALU から出力される。一方、検出信号は加算器内で 2 段、さらに組み込む際に n 段必要とし、計 2+n 段の通過で出力される。TITAC-2 の設計では、この 2 つ段数の間に、仮定する遅延変動が生じてもタイミングに余裕があることを確認し、演算終了の検出信号としている。

5 乗算器の構成

高速な乗算方式として、桁上げ保存加算器 (Carry Save Adder, CSA) を使用する 3 種類の構成を比較検討するため、ランダム値 (符号付き) の入力による平均計算時間をシミュレーションにて計測した。

1. アレイ型 (CSA30 段) : 20.67ns
2. ツリー型 (Dadda、CSA8 段) : 8.50ns
3. ツリー型 (Wallace、CSA7 段) : 8.11ns

レイアウトの容易さを重視すればアレイ型となるが、回路量に大差はなく、演算速度を最も重視する方針から Wallace ツリーを採用した。また、使用する全加算器、半加算器を DCVSL により構成し、高速化を計った。

さらに高速化するため、ツリー最終段にある 32bit 加算器を、非同期式に適した構成にする検討をした。しかし静的な解析では、最終段の加算器へ入る信号のタイミングを特定することが非常に困難であるため、桁上げ伝播加算器 (Ripple Carry Adder, RCA) と CLA 方式を組み合わせた回路をいくつか用意し、シミュレーションによる速度評価を行った。

その結果、4bit CLA を並べた方式 (図 5) が最も高速であるため採用した。これにより、入力に 32 ビットのランダム値を与えた際、RCA を使った時より 0.42ns、CLA 加算器を使った時より 0.49ns 高速化した。

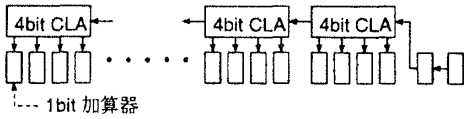

図 5: 乗算器最終段の加算器の構成

6 まとめ

TITAC-2 の ALU の回路構成に関して、非同期式特有の、演算以外にかかる時間を短縮する方式とその実装方法を示した。また乗算器に関して、設計時に行った速度評価と採用した構成を示した。

なお、本研究の一部は、新エネルギー・産業技術総合開発機構 (NEDO) 提案公募型・最先端分野研究開発事業受託研究 C-026、並びに科学研究費補助金 (試験研究 B)07558036 によって行われたものである。

参考文献

[1] K.Chu, D.Pulfrey, "Design Procedures for Differential Cascode Voltage Switch Circuits" *IEEE Journal of Solid-State Circuits*, Vol.SC-21, NO.6, Dec 1986

[2] 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, Jan 1993