システムＬＳＩ設計技術　109－10
(2003. 3. 11)

プログラマブルコントローラ向けアーキテクチャの検討と評価

山口　大介†　　松永　裕介††

† 九州大学大学院システム情報科学府
†† 九州大学大学院システム情報科学研究院
E-mail: †{daisukey,matsunaga}@c.csce.kyushu-u.ac.jp

あらまし　FAの分野では，プログラマブルコントローラ（PLC）という制御機器が普及している．PLCは，主にプロセッサ部とメモリ部から構成されるが，メモリのコストがPLC全体のコストの大部分を占めている．よって，メモリのコストを減らす本稿では，PLCにおけるプログラムの性質を用いて内部の処理を効率的に行うことで，メモリアクセス回数を隠蔽し，低速な汎用メモリを使用しても性能の低下を最小限度に抑えることのできるPLC専用アーキテクチャを検討し，その性能を評価した．
キーワード　プログラマブルコントローラ，ラダー図，論理演算，専用アーキテクチャ

Consideration and Evaluation on Dedicated Architecture for a Programable Controller

Daisuke YAMAGUCHI† and Yusuke MATSUNAGA††

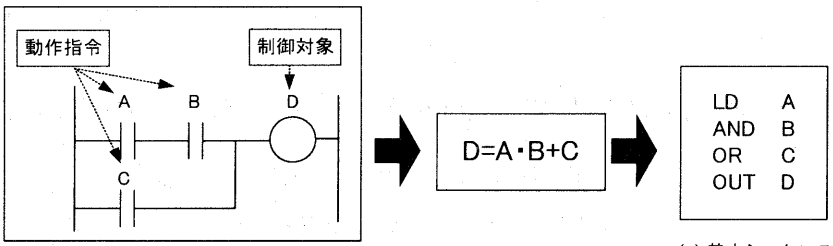
† Graduate School of Information Science and Electrical Engineering Kyushu University
†† Graduate School of Information Science and Electrical Engineering Kyushu University
E-mail: †{daisukey,matsunaga}@c.csce.kyushu-u.ac.jp

Abstract　A Programable Controller(PLC) is used in the field of FA(Factory Automation). PLC equips the processor and the general-purpose memories. The cost of the memories accounts for a large portion of the whole cost of the PLC, so it is required to cut costs of the memories. However, the performance of the PLC is determined by that of the memories, so there is a trade-off between the performance and the cost. In this paper, we consider the dedicated architectures which can minimize the effects of the memory access latency ,and evaluate the performance.
Key words　Programable Controller, Ladder, Logic Evaluation, Dedicated Architecture

1. はじめに

産業分野ではFA(Factory Automation)技術が用いられているが，このFA技術の中心的技術として自動制御技術が重要な役割を果たしている．以前は，自動制御実現のためにリレー回路によって構成されたシーケンス制御回路，いわゆるリレーシーケンスが使用されていたが，近年の制御内容の複雑化および高度化の要請により，マイクロプロセッサ技術を応用したプログラマブルコントローラ(以下PLC)と呼ばれるシーケンサが広く使用されるようになった[1]．PLCは主に，プロセッサ部，メモリ部から構成されており，PC(Personal Computer)などで書かれたプログラムを実行することにによって機器を制御する．このため，以前のリレーシーケンスによる制御では，一度完成した装置の機能を変更させるには配線のやり直しなどの手間と時間が必要だったのに対し，PLCではソフトウェアによって簡単に機能の変更ができる．

PLCのプログラミング言語として，図1 (a)のような有接点リレー回路に似た表現形式のラダー図が最も多く使用されている[2]．ラダー図では，押ボタンスイッチやリレー，リミットスイッチやモータなのど規格化されたシンボルと記号を用いて操作や動作順序などの制御上の機能が描かれている．図1の場合，接点A,B,Cはスイッチの ON/OFF 状態を表しており，接点Dは出力リレーとなっている．この図が表わす意味は，「接点AおよびBがON状態であるか，または接点CがON状態であるならば，接点DをON状態にせよ」ということである．つまり，これは，ON状態を“1”，OFF状態を“0”とすれば $D = A \cdot B + C$ という論理式に変換できる(図1 (b)参照)．このようにラダー図で描かれたプログラムは，接点を変数として任意の論理式に変換可能という特徴を持つ．PLC上のプロセッサで上記のラダー図を実行する場合，基本シーケンス命令と呼



(a) ラダー図　(b) 論理式　(c) 基本シーケンス命令

図1　PLCの概要

ばれるPLC独特の命令に変換される．この基本シーケンス命令は，基本的に論理演算に必要な命令で構成されており，例えば前述した論理式 $D = A \cdot B + C$ は図1 (c) のような5つの命令に変換される．基本シーケンス命令の他の命令として応用命令というものもある．なお，本稿では，基本シーケンス命令に該当する命令を便宜上，論理演算命令と呼ぶことにする．

現在，PLC全体のコストのうち大部分を占めるのがメモリ部のコストである．そのため，安価で性能が低いメモリを用いても，既存のPLCと同等の性能が得られるプロセッサの開発が要求されている．しかし，PLCのパフォーマンスはメモリの性能に依存しているため，性能が低いメモリ，つまりメモリレイテンシが大きいメモリを使用する場合，PLC全体のパフォーマンスも低下してしまう．そこで本稿では，PLCのプロセッサ部において，内部で効率よくデータ処理することで，PLC全体のパフォーマンスの低下を抑えるアーキテクチャの検討を行なっている．

本稿では，2章でプログラマブルコントローラの概要を説明し，3章で現在検討しているアーキテクチャの説明を行なう．4章では，検討中のアーキテクチャに関して性能の見積もりを行ない，5章では本稿をまとめと今後の課題を述べる．

2. 準　　備

本章では，比較対象として，現在想定している既存プロセッサの概要および内部構成，プログラムの説明を行なう．なお，本稿ではPLC性能の尺度として，クロックサイクル数を用いる．

2.1 プログラマブルコントローラ概要

現在PLCは，図2のような制御フローを常に繰り返しているものと想定している．PLCは以下のフローを繰り返すことで対象物の制御している．フローの各部の詳細は以下のとおりである．

1. 初期化処理：各種初期化およびI/Oリフレッシュを行なう
2. 評価処理：ラダー図の評価を行う
3. 各種サービス：通信処理などを行う

初期化部分では，各種初期化処理を行ない接点情報のデータの内容をリフレッシュする．リフレッシュとは，各制御対象の最新状態を取り込んだり，逆に現在の状態を格納されている情報を各制御スイッチに反映させることである．次に，評価処理部では，プロセッサが前章で説明した命令を実行する．最後に各種サービスでは，他のPLCとの通信を行ったり，アプリケーションへのサービスを行なう．ここでは図2に示す処理を1通り行なうことを「PLCの1サイクル」と定義する．また前提として，格納されている各接点情報（入力情報）は，1度リフレッシュされてから次にリフレッシュされるまで外部から変化が加わることはないものとする．

PLCは図3のように，プロセッサ部，プログラムを格納するメモリ (命令メモリ，以下IM)，機器の接点情報を格納するメモリ (データメモリ，以下DM) から構成されているものとする．今回想定してる各メモリへのデータ幅は，プロセッサ部とIMは32ビットであり，プロセッサ部とDMは16ビットである．なお，IMとDMは汎用の1ポートメモリを用いているものとし，1ワード単位でしかアクセス出来ない．また，プロセッサ内部には特にキャッシュ機能は持っていないものとする．なお，IMおよびDMは既存プロセッサの性能を向上させるため，1クロックサイクルでアクセスできるものとする．

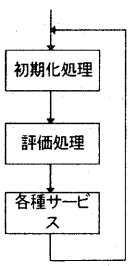


図2　制御フロー

2.2 既存プロセッサの内部構成

既存のプロセッサは，ロード/ストア型のアーキテクチャ[3]であり，内部にアキュムレータを持っている構成になっている．プロセッサの内部は，図4のように，Feステージ，Deステージ，Exステージの3段のパイプラインステージで構成されている．Feステージでは，IM上の命令の読み込みを行なわれ．次のDeステージに命令を受け渡す．Deステージでは，Feステージで読み込まれた命令のデコードを行ない，次のExステージで行なわれる指令を出す．Exステージでは，Deステージでデコードされた命令を基に，データの読み込み，書き込み，および演算を行なう．演算は，ALU(Arithmetic Logic Unit) を用

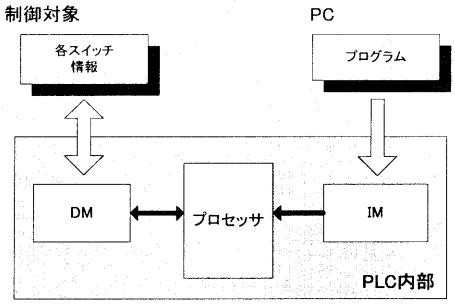


図3　PLCの概要

いて行われ，結果はExステージ上のアキュムレータに書き込まれる．なお，論理演算で必要な変数のビット幅は1ビットであるが，DMには汎用メモリを使用しているため16ビット単位で読み込みおよび書き込みが実行されると仮定する．よって，DMへ1ビットのデータを書き込むためには，1度DMから該当する変数をプロセッサ側に読み込んだ後，必要データを書き込み，メモリに書き戻す，という手順を踏むので，メモリアクセスが2回必要になる．

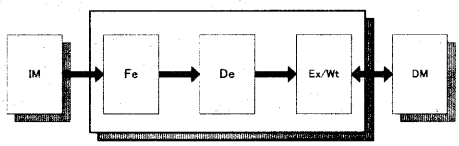


図4　プロセッサの内部構造

2.3　プログラムの特徴

前章で述べたように，PLCによる制御を行なう場合，プログラミング言語としてラダー図が広く使われている．このラダー図はPLCのプロセッサが用意した命令に変換されて実行されるが，この命令には大きく3つの特徴があると仮定する．まず1つ目の特徴として，命令全体の約9割が論理演算命令であることが挙げられる．つまり，言い換えればPLCの命令は，ほとんど論理式に変換可能であるということであり，その度にDMへのメモリアクセスが発生する．2つ目の特徴として，PLCの命令はシーケンシャルに実行されという性質がある．ラダー図のプログラムは有接点リレー回路に由来するため，プログラム中に分岐やループが存在することはない．3つ目の特徴として，変数が複数回アクセスされることがある，という点である．変数は，DM上のアドレスと1対1で対応しているため，つまり同じアドレスへのアクセスが複数回存在しうるということになる．また，論理演算命令は32ビット長で1命令で表現されている．

2.4　性能の低いメモリを使用した場合の問題点と改善策

前章でも述べたとおり，メモリのコストを下げることはPLC全体のコストを下げる上で非常に有意義である．しかし，実際，メモリのコストを下げることは，メモリの性能を低下させることになり，メモリの性能に依存しているPLC全体のパフォーマンスの低下にもつながる．例えば，1クロックサイクルでアクセスできたメモリがあったとする．これを2クロックサイクルかかるメモリに変更した場合，毎命令DMへのアクセスを行うので，プロセッサ側では1クロックのデータ待ち（ここでは，ストールと呼ぶ）が発生する．これにより，PLCは後続の命令実行が1クロック遅れてしまうため，プログラム全体を処理するために必要なクロックサイクル数も大きくなり，PLC全体のパフォーマンスは低下する．このように，既存プロセッサでは，メモリの性能を低下させた場合，それに比例してPLCの性能も低下させてしまう．そこで，この性能低下を抑える解決策として，PLCの内部の処理を効率的に行い，メモリアクセス回数を削減すればよいと考えた．まず1つ目の効率化として，DMへのアクセス回数の効率化を考えた．つまり，前節で述べた，同じ変数に複数回アクセスする変数が存在するプログラムの特徴を用いて，この変数を内部記憶機構に保持することでDMへのメモリアクセス回数の削減が期待できると考えた．また2つ目の効率化として，IMへのアクセス回数の効率化を考えた．つまり，PLCのプログラムの9割が論理演算命令であることを考え，論理式評価を効率的に行なう機構を用いることでIMへのアクセス回数を削減することができると考えた．そこで，次章ではこの2つの効率化を用いたアーキテクチャの提案を行い，検討する．

3.　メモリアクセス回数削減のためのアーキテクチャ

前章で述べたとおり，既存プロセッサのまま性能の低下したメモリを使用した場合，毎命令メモリアクセスが発生するため，PLC全体の性能が低下してしまう．そこで，本稿ではPLCのプログラムの特徴である

- 論理演算が9割占める．
- 毎命令メモリアクセスを行う．
- 命令は逐次実行される．
- 同じ変数に数回アクセスすることがある．

ということに着目し，内部で効率よくデータを処理することでIMおよびDMへのメモリアクセス回数を削減するアーキテクチャを提案する．

3.1　概　　要

既存プロセッサをベースにして，2つのモジュールを追加するアーキテクチャを提案する．

今回提案するアーキテクチャで追加するモジュールは下記の機能をもつ．

1. 論理演算のみを行なう専用モジュール
2. 内部に高速で容量の小さな記憶機構をもつモジュール．

前者に該当するモジュールとして，LEU(Logic Evaluation Unit)を提案する．LEUは，1度にnリテラルの論理式を評価可能なモジュールであり，1ビットの制御情報によってAND/ORを切り替えることができるユニット(Logic Unit，以下LU)を基本素子とし，この素子を木状にすることで構成される．図5にLEUの構成例，およびLUの構成を示す．図5の例は，5つのLUを木状に並べた構成になっており，この構成で任意の4リテラルの論理式を評価可能である．図5の場合，

C1,C2,C3,C4 に任意の値 (0 or1) を入力することで，LEU は任意の論理式を評価できることになる．また，同時に命令部も変更され，n リテラル以下の論理式を 1 命令で表現できるようにする．

後者は，バッファ記憶モジュールを提案する．キャッシュとの相違点は，バッファ内のデータをプロセッサ側からアドレス指定して呼び出す点である．通常キャッシュは，プロセッサから指定されたメモリアドレスに応じて，キャッシュ内を検索して存在すればキャッシュ内のデータを返し，存在しなければメモリからデータを検索する，という方式を取っている．しかし，PLC の特徴としてシーケンシャルに実行されることを考えると，コンパイラ上でデータがある場所を解析することができるので，任意の場所に直接アクセスできるバッファを用いた方が効率がよく処理できる．これによって DM へのメモリアクセス回数を削減する．ただし，バッファを用いた DM へのアクセス回数削減はコンパイラの性能にも依存することになる．

LEU とバッファは，前章で述べた図 4 の Ex ステージに図 6 のような構成で接続することを検討している．図 6 は ALU の論理演算以外の応用命令を受け持つユニットである．全てのデータはバッファを介して行われ，一時的にバッファに蓄えられるものとする．

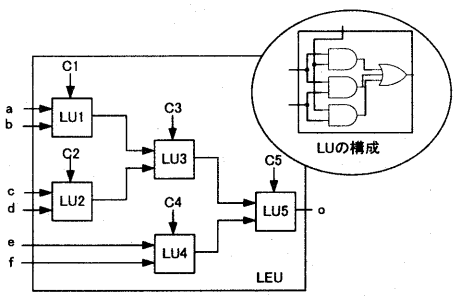


図 5　LEU の構成

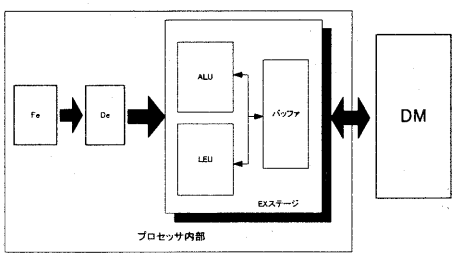


図 6　アーキテクチャの概要

3.2　LEU による IM へのアクセス回数削減法

前節で述べたように，LEU モジュールを用いた場合 1 命令で n リテラル以下の論理式を表現できるようにすると，論理式を評価するために必要な IM アクセス回数が減ができる．例えば，論理式 D=A· B+C を評価したいと仮定したとき，既存プロセッサにおける命令（ここでは，通常命令と呼ぶ）の場合，1 命令 32 ビットあり，図 7のように 4 命令で表現されるので，全部で 4 回の IM アクセスが必要である．一方，LEU を用いた命令（ここでは LEU 命令と呼ぶ）の場合，ヘッダ部 16 ビット，オペランド 16 ビットで指定できるとすれば，1 命令のビット長は全部で 16+4 ×16=80 ビットなので，IM のワード長が 32 ビットであることから $\lceil 80/32 \rceil = 3$ 回の IM アクセスで十分になる．

ただし，1 論理式だけで考えた場合，DM での処理も考えると通常命令で処理を行なった方が LEU 命令よりもスループットの方が良いことが多い．例えば図 7 の場合で考えると，通常命令においては図 8 (a) のようにパイプライン処理されるため，11 クロックサイクルで論理式の評価が終了する．しかし，LEU 命令の場合，命令語長は削減できても，DM アクセスに時間がかかるため，図 8 (b) のように 15 サイクル必要となり，4 サイクルも余分にクロックサイクルがかかってしまう．

この問題は，本来 PLC のプログラムはいくつもの論理演算から成り立っているので，通常命令と同様に i 番目命令が Ex ステージで処理を行なっている最中に，$i+1$ 番目命令を読み込む処理を行なっていれば，$i+1$ 番目の Fe ステージにおけるクロックサイクル数が隠蔽されると考える．また，本アーキテクチャでは，バッファによって DM アクセス回数を削減するので，Ex ステージのクロックサイクル数も削減でき，プログラム全体の全体の性能の低下は抑えることができるものと考える．なお，LEU が処理できる入力数のことを「LEU の入力数」と呼ぶ．

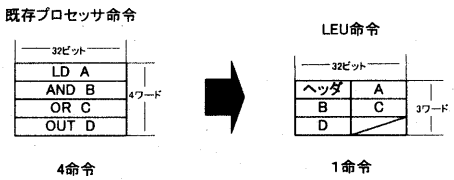


図 7　命令の変換例

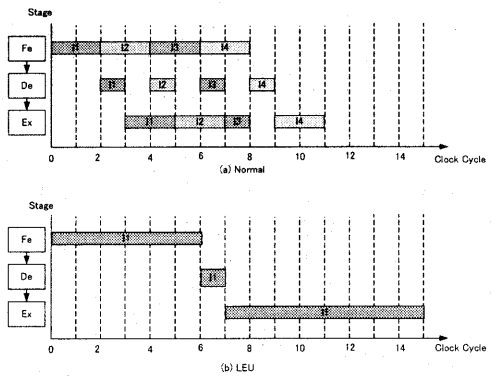


図 8　$D = A * B + C$の処理にかかるクロックサイクル数

4.　実験および考察

本章では，前章で述べたアーキテクチャの性能を評価を示すため，4 つのモデルにおいて性能を見積もる実験を行なった．

4.1 アーキテクチャの分類

今回提案するアーキテクチャの特徴として，データアクセス方式としてバッファ，および論理演算方式として LEU を用いていることが挙げられる．そこで，PLC 専用アーキテクチャのモデルとして，データアクセス方式と論理演算の方式によって分類し，4 つのモデルを考えた．表 1 に 4 つのモデルを示す．

論理評価方式による分類とは，論理式を評価するユニットによって分類したものであり，ALU 方式と LEU 方式に大別される．ALU 方式の場合，ALU を用いて論理式を評価するので，1 命令で評価できるデータ数は 1 つになる．それに対し，LEU 方式の場合，1 度に複数リテラルの論理式を評価できるので，1 命令で n 個のデータを処理出来ることになる．ただし，ALU 方式の場合，基本的に論理評価命令に必要命令語長は 1 ワード（つまり，IM へのアクセスが 1 回）であるのに対し，LEU 方式の場合，命令ビット長 l ビットにおいて $\lceil l/32 \rceil$ 回のメモリアクセスが必要になる．

メモリアクセス方式による分類とは，命令を実行する際のデータ読み込み方式によって分類したものであり，メモリ方式とバッファ方式に大別される．メモリ方式とは，データの読み込みおよび書き込みが，DM を直接アクセスする方式のことである．つまりこの方式では，データは DM から読み込まれ，DM に書き戻される．内部にはバッファ（もしくは，キャッシュ）を持たないので，1 命令実行する度に DM へのアクセスが必要になる．それに対しバッファ方式では，内部に記憶できるバッファを持つため，DM へのメモリアクセス回数が削減される．

つまり，既存のプロセッサモデルとしたは M-A 型が該当し，今回提案するプロセッサのモデルとしては B-L 型が該当する．

表 1　PLC プロセッサモデル

論理評価方式	アクセス方式	
	メモリ方式	バッファ方式
ALU 方式	M-A 型	B-A 型
LEU 方式	M-L 型	B-L 型

4.2 実験方法

前節で述べた 4 種類のプロセッサモデルに関して性能をプログラムによってシュミレーションし，比較検討した．サンプルプログラムとしては，異なる 4 種類のプログラム (Program A, Program B, Program C, Program D) を用意した．実験手順としては，まずサンプルプログラムを各モデルにあった命令に変換し，次にその変換された各命令のクロックサイクル数を見積もり，最後にその値からプログラム全体にかかるクロックサイクル数を計算し，それを各モデルの性能として評価した．

各命令におけるクロックサイクル数の見積もりは，以下の式で算出した．

$$C_{\mathrm{Fe}} = N_{\mathrm{I}} \times C_{\mathrm{IM}} \tag{1}$$

$$C_{\mathrm{De}} = 1 \tag{2}$$

$$C_{\mathrm{Ex}} = N_{\mathrm{MEM}} \times C_{\mathrm{DM}} + N_{\mathrm{BUF}} \times C_{\mathrm{BUF}} \tag{3}$$

C_{Fe}　Fe ステージでかかるクロックサイクル数
C_{De}　De ステージでかかるクロックサイクル数
C_{Ex}　Ex ステージでかかるクロックサイクル数
N_{I}　命令ワード数
N_{MEM}　1 命令あたりの DM アクセス回数
N_{BUF}　1 命令あたりのバッファアクセス回数
C_{IM}　IM にアクセスする時に必要なクロックサイクル数
C_{DM}　DM にアクセスする時に必要なクロックサイクル数
C_{BUF}　バッファにアクセスする時に必要なクロックサイクル

ここで命令ワード数とは，1 命令当りのワード数のことと定義し，通常命令の場合は，1 命令 1 ワードとして計算した．LEU 命令の場合はヘッダ部 16 ビット，メモリアドレス 16 ビット，バッファアドレス 8 ビットで計算し，LEU の入力数以上の論理式になった場合は，命令を増やすことで対処した．なお，バッファサイズは 256 ビットを想定しており，バッファに記憶される変数は，使用頻度の高い変数から上位 256 個を選ぶ手法を選んだ．

4.3 実験結果および考察

表 2 , 表 3 , 表 4 に実験結果を示す．表の中の値は，プログラム全体のクロックサイクル数を示している．また，括弧内の数字は，M-A 型 (既存プロセッサ) を基準とした各モデルのクロックサイクル削減率を示している．$C_{\mathrm{IM}}, C_{\mathrm{DM}}, C_{\mathrm{BUF}}$ の各パラメータは，表 3,4の実験では，$C_{\mathrm{IM}} = 5, C_{\mathrm{DM}} = 5, C_{\mathrm{BUF}} = 1$ と固定した．

表 2は，メモリアクセスのレイテンシを増加させたときに Program A を実行するのに必要なクロックサイクルの見積もりを，M-A 型と B-L 型において見積もった結果である．現在の PLC のメモリアクセスのレイテンシを基準として，つまり $C_{\mathrm{IM}} = 1, C_{\mathrm{DM}} = 1$ と，各メモリにアクセスするために必要なクロックサイクル数 $C_{\mathrm{IM}}, C_{\mathrm{DM}}$ を可変にした．また，表 3 の実験結果は，LEU の入力数を 4 として各サンプルプログラムにおける各モデルの性能を見積もった．また表 4 の実験では，LEU の入力数を 5 とした場合の各モデルの性能を見積もった．

表 2 の結果より，PLC はメモリの性能に比例していることが分かる．ただし，提案プロセッサの方がメモリアクセスのレイテンシ増加に伴うクロックサイクル数の増加の割合は低いので，既存プロセッサのモデルに比べ，提案プロセッサの方が PLC 全体のパフォーマンスの低下は抑えれられている．つまり，提案プロセッサモデルはメモリの性能の影響を受けにくいことが分かった．

また，表 3 4 から LEU の入力数が 4,5 のどちらでも，M-A 型に比べ，B-L 型の方が平均で 30%のクロックサイクル数を削減できているという実験結果が得られた．また，B-A 型や，M-L 型に比べ，B-L 型の方がクロックサイクル数が削減できているという結果がでており，このことからバッファ，LEU のどちらか一方を用いたアーキテクチャでは全体のクロックサイクル数の削減にあまり効果がないことが分かった．つまり，提案プロセッサモデルは，バッファおよび LEU の両方があることで，メモリの性能の低下による影響を抑えている．

これはおそらく，IM もしくは DM へのアクセスに時間がかかってしまい，ストール状態になるためと思われる．そのため，バッファのみもしくは LEU のみで用いるよりは，両方を組み合わせて用いたほうがこのストールを解消でき，効率よく処理できることが分かった．また，LEU の入力数 4 の場合のクロックサイクル数と，LEU の入力数が 5 の場合で比較した場合，全体のクロックサイクル数は削減できていない．このことより，あまりリテラル数の大きな論理式は存在せず，リテラル数が 4,5 くらいの論理式が多いため LEU の入力数を増やしても効果はあまり得られないことが分かった．

表 2　メモリレイテンシの増加による各モデルの影響比較

レイテンシ	既存プロセッサ	提案プロセッサ
1	6646	6524
2	11774	9913
3	17156	13303
4	22814	16693
5	28515	20083
6	34216	23473
7	39917	26863
8	45618	30253
9	51319	33643
10	57020	37033

表 3　LEU=4,Buffer=256

	M-A 型	B-A 型	M-L 型	B-L 型
Program A	28515	26687 (6.41%)	28461 (0.19%)	20538 (27.97%)
Program B	85443	82056 (3.91%)	81281 (4.87%)	61240 (28.33%)
Program C	66328	59974 (9.581%)	55368 (16.52%)	49938 (24.71%)
Program D	131957	121399 (8.00%)	110142 (16.53%)	78831 (39.63%)

表 4　LEU=5,Buffer=256

	M-A 型	B-A 型	M-L 型	B-L 型
Program A	28515	26687 (6.41%)	28461 (0.19%)	20538 (28.66%)
Program B	85443	82056 (3.96%)	81281 (4.87%)	61240 (28.33%)
Program C	66328	59974 (9.58%)	55368 (16.52%)	49938 (24.71%)
Program D	131957	121399 (8.00%)	110142 (16.53%)	78831 (40.26%)

5. おわりに

本稿では，PLC 向けアーキテクチャの検討および評価を行った．PLC のプログラムは，9 割の論理評価命令を持ち，同じ変数が複数回アクセスされることがあるという性質がある．そこで，今回検討中のアーキテクチャには，論理式評価を専用に行う LEU(Logic Evaluation Unit) とプログラム中で頻繁に出現する変数を蓄えておくバッファを搭載させ，メモリへのアクセス回数を削減させた．LEU およびバッファを用いることによる性能評価を行うため，M-A 型モデル，M-L 型モデル，B-A 型モデル，B-L 型モデルの 4 種類のプロセッサモデルを考え，それぞれにサンプルプログラムを用いて評価した．これによって，LEU およびバッファを同時に搭載するモデルが，メモリの性能の影響を受けにくいことが分かった．ただし，提案プロセッサにおいても安価なメモリを用いたことによる PLC の性能低下は避けられないので，今後より効率よく処理してメモリの性能を受けにくいアーキテクチャを検討していくと共に，PLC 専用アーキテクチャの構成とコンパイラの手法を関連付けながら，今後の研究進めて行く必要があると考える．

今後の予定としては，プログラム中にストールの原因になっている部分をより詳しく調査し，より効率良く処理する仕組みの検討を行う予定である．また，現在バッファに置く変数のアロケーションは固定されているため，バッファ内のアロケーションが時間的に変化するアルゴリズムを検討し，よりメモリアクセス回数を削減する手法を調査する予定である．

謝　辞

本研究は，平成 14 年度学術創成研究によって行われたものである．サンプルプログラムを提供して頂いたオムロン株式会社の方々に深く感謝の意を表します．

文　献

[1] 技術評論社，望月傳，“図解でわかるシーケンス制御の基本”，ISBN4-7741-0681-X C3054

[2] 制御機器の基礎知識 - プログラマブルコントローラ (PLC) 編 - 社団法人 日本電気制御機器工業会

[3] David A. Patterson, John L. Hennessy, “コンピュータ・アーキテクチャ ー設計・実現・評価の定量的アプローチー”，富田眞治 村上和彰 新貫治男訳,ISBN4-822-7152-8, 1993