平成７年度
石田 (實) 記念財団研究助成
研究成果報告書

高速情報圧縮復元ハードウェアアルゴリズムに関する研究

１９９６年１１月

阿曽弘具

東北大学工学部通信工学科

高速情報圧縮復元ハードウェアアルゴリズムに関する研究

あらまし　コンピュータネットワークの発展により，情報通信の利用が盛んとなり，情報をいかに効率的に伝送，蓄積するかが重要な課題となっている．そのため様々なデータ圧縮技術が開発されているが，ソフトウェアによる実現が主なものであり，現状では，大量の情報を扱うための速度的要求を満足することが困難になってきている．この困難の解決法として，VLSI/ULSI 技術を用いた専用ハードウェアによる高速なデータ圧縮が考えられている．本研究では，様々な情報に対して有効な符号化法の一つである LZ77 符号化法を対象に，それによる符号化・復号化のためのハードウェアアルゴリズムを考案し，その性能の評価のため，高位論理合成システム PARTHENON を用いて論理設計等を行なった．ハードウェアアルゴリズムは，並列処理アーキテクチャ - PAHL-C／PAHL-D により実現した．PAHL-C は情報圧縮のための符号化アーキテクチャで，符号化で最も計算量を要する最長一致記号系列探索での冗長な計算を削減し，スルー・レートを大幅に向上するものである．その性能評価により，約 20〜25 MByte/sec というスルー・レートが得られることを確かめた．PAHL-D は符号化された情報を復元するための復号化アーキテクチャで，約 30〜40 MByte/sec というスルー・レートの性能が得られることを確かめた．

1　はじめに

さまざまな分野へのコンピュータの普及およびネットワークの普及に伴い，多種多様且つ大量の情報をコンピュータ上で扱う機会が増えてきている．そのため情報をより効率的に伝送，蓄積することが要求され，そのための有効な技術として情報圧縮技術が重要となっている．情報の圧縮／復元すなわち，符号化／復号化する既存の手法は，利用者の明示的な指示によるオフライン的実行であり，今後要求される大量の情報やオンライン的情報を扱うことに不都合が生じることが予想される．その解決策として，コンピュータと通信線・外部記憶装置の間で自動的に圧縮／復元を実現することが考えられる．これにより既存コンピュータの応用ソフトウェアに変更加えることなく，より大量の情報やオンライン的情報を効率的に処理できることになる．この実現のため，VLSI/ULSI 技術を応用したデータ符号化／復号化ハードウェアの構築が有用である．

現存するデータ符号化／復号化技術には様々なものがあり，データ符号化／復号化ハードウェアにおいても様々なものが提案されている [1, 2, 5, 6]．本研究では，多様な形式の情報において高い圧縮率を実現するデータ符号化／復号化手法の一つである，Lempel-Ziv 符号化法（以下，LZ 符号化法）について，その LZ 符号化法の一つである LZ77 符号化／復号化法のハードウェアアルゴリズムを考案し，並列処理アーキテクチャとして実現できることを示した．

LZ77 符号化法において最も計算時間を要する処理は最長一致記号系列の探索である．そこで，この探索の高速化に焦点をあてた．まず，LZ77 符号化による生成符号の長さなどに関して統計的特徴を調査し，それがほとんどの場合，ある一定値以下になるという結果を得た．

本研究で考案した LZ77 符号化アーキテクチャは，この統計的特徴を有効利用できるもので，高速なデータ符号化を可能にする並列処理アーキテクチャ - PAHL-C(**P**arallel **A**rchitecture for **H**igh-speed **L**empel-ziv data **C**oding) である．LZ77 符号化法のハードウェアに関しては，1 次元シストリックアレイによるアーキテクチャ[1] や，Broadcast/Reduce によるアーキテクチャ[2] など，並列処理技術を応用したものがいくつか提案されている．本研究で考案した LZ77 符号化アーキテクチャ PAHL-C では，冗長な比較演算をできるだけ削減している．これにより，実際のデータに対して，高い符号化スルー・レートが達成できることを示す．

PAHL-C では，最長一致記号系列探索部を各位置からの一致長を判定する部分と，各位置での一致長から最長一致系列を決定する部分とに分割し，それぞれを並列に同期動作させ，より高いスルー・レートの符号化処理を実現する．文献 [2] で提案されているアーキテクチャは同様の考えで構成されていると考えられるが，最長一致系列を決定する部分に長さ情報の比較器があり，大量の比較素子を必要とする．PAHL-C では，それを削減し，実現可能なものにしている．実際に PAHL-C について，高位論理合成システム PARTHENON （開発: NTT 情報通信網研究所）を用いて論理設計，動作シミュレーション，論理合成等を行ない，PAHL-C の実現性を検証し，性能を定量的に評価した．

復号化法については，復号化アーキテクチャ PAHL-D を考案し，PARTHENON を用いて論理設計を行い，実現可能性を検証した．さらに，PAHL-C ／ PAHL-D を効率的に統合して情報圧縮／復元ハードウェアアルゴリズムを構成した．

2 LZ 符号化法

LZ 符号化法は，J. Ziv と A. Lempel により提案された符号化法である．大別すると，1977 年に提案された符号化法 [3]（一般には LZ77，または LZ-1 符号化法と呼ばれる）と，1978 年に提案された符号化法 [4]（一般には LZ78，または LZ-2 符号化法と呼ばれる）との二種類がある．どちらの手法も，符号化に用いる辞書 (符号表) をあらかじめ用意せず，符号化する記号系列を読み込みながら順次辞書を編集，符号化していく符号化法であり，同一のアルゴリズムで広範囲の情報源に対し有効な圧縮率を実現できるユニバーサル符号化法の一つである．LZ77 符号化法，LZ78 符号化法は，現在コンピュータにおいて主要なデータ圧縮技術の一つとして使用されている．本論文ではハードウェア化の対象とする符号化法として LZ77 符号化法を採用した．

LZ77 符号化法には様々な改善手法が存在し，その代表的なものとして LZSS 法がある．LZSS 法は現在ある主要な圧縮アプリケーション (lha, gzip など) に用いられている．この手法では，LZ77 法における符号内の冗長性を削減することでより高い圧縮率を実現するものであるが，基本的な符号化処理は LZ77 法と同様である．LZSS 法のハードウェア化は，高速な LZ77 符号化アーキテクチャを基に付加的な処理を組み込むことで実現できる．

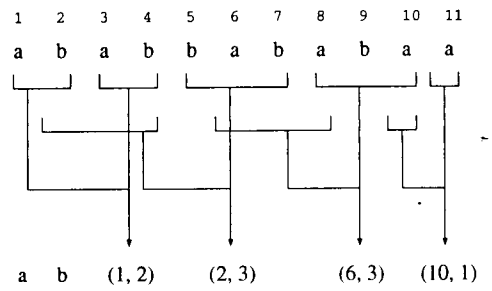

図 1: LZ77 符号における辞書参照法

2.1 LZ77 符号化法

LZ77 符号化法では，スライド窓と呼ばれるバッファを設定し，既に符号化された記号系列 (辞書)，次に符号化すべき記号系列をスライド窓上の参照部，符号化部に置く．このバッファは符号化が進むにつれて記号系列上をスライドしていく．

実際の符号化過程における辞書の参照は以下のとおりである．既に符号化の完了した記号系列（参照部内の記号系列：辞書）内で，次に符号化されるべき記号系列（符号化部内の記号系列）の先頭位置から始まる記号系列との最長一致記号系列を探索することを基本としている．辞書内の位置を示すポインタをPとし，位置Pからの長さLの記号系列 $S[P, P+L-1]$ を (P, L) と表現することとすると，図1のような記号系列 "ababbababaa" は "a b (1,2) (2,3) (6,3) (10,1)" と符号化される．また，逆に "a b (1,2) (2,3) (6,3) (10,1)" が与えられると，各符号 (P, L) から元の記号列を復元することができる．

LZ77 法の圧縮率 (圧縮後の大きさ／圧縮前の大きさ) に関しては，データの種類に大きく依存し，データによってはほとんど圧縮できないものもあるが，だいたいLZ77法のみでも約2/3〜1/2程度，LZSS 法ではそれ以上の圧縮が実現可能である [8].

3 生成符号の統計的特徴

3.1 一符号当りの比較回数

LZ77 符号化法での主要な処理は最長一致記号系列探索であり，その処理には並列処理可能な部分が多く存在する．この並列処理を実現するものの一つとして一次元シストリックアレイを用いたものがあるが，参照部バッファ長をNとすると，一符号当りの最長一致記号系列探索にはO(N)の比較回数を要する (ちなみに逐次的手法ではO(MN)である．但しMは符号化部バッファ長である)．実際に存在する最長一致長をLとすると，最長一致記号系列探索には，L+1回の比較で十分なはずである．最長一致記号系列探索をL回程度の比較に抑えることができれば，LがNに比べて十分に小さい場合，LZ77 符号化法でのスルー・レートを大幅に向上することが可能となる．

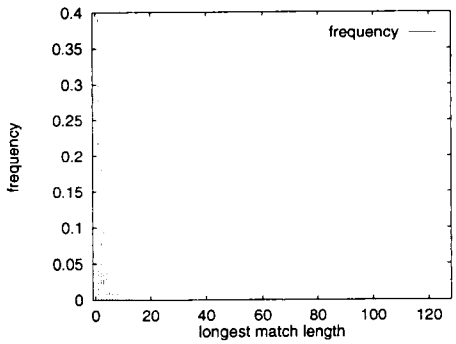

図 2: 毎符号に対する最長一致記号長の頻度分布

3.2 参照部バッファ長と最長一致記号系列長

現実の圧縮処理対象データに関して，その符号化における実際の最長一致記号系列長がどのような長さであるのかを調査した．様々な形式のデータファイルについて，1データファイル当りの最長一致記号系列長の出現頻度を求めた．様々な形式のデータファイルについて，符号毎に求められる一致長の頻度分布はデータの形式によらずほぼ同様のものであった．その一例を図2に示す．なお，記号（文字）系列の長さは 1 byte 単位で考えている．

図 2では参照部バッファ長 N=128[byte]，符号化部バッファ長 M=128[byte]，縦軸が頻度，横軸が最長一致長である．N を様々な値に変化させて同様な頻度分布を調査（N = M とする）したところ，N を大きくすると頻度分布の幅は若干の広がりをみせたが，頻度分布としては図 2 とほぼ同様の分布を示した．この頻度分布から，探索される最長一致長のほとんどが 0〜12 程度に集中していることがわかる．ときおり最長一致長が 100 程度，またはそれ以上の場合もあるが，全体からみると非常にまれであり，統計的には，一符号当りの比較回数は大部分が 0〜12 程度であるものとみなせることがわかった．

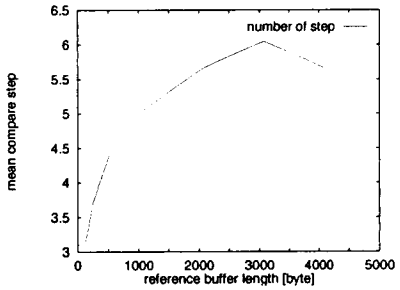

図 3: 一符号当りの平均比較回数

この調査データから N を変化させた場合の一符号当りの平均比較回数を求めると図 3のようになった．この結果は，一符号当りの平均比較回数が N が増加してもほぼ 6 程度で収まることを示している．このことから，最長一致記号系列が見つかった時点で比較をやめることができれば，一つのデータファイルを符号化するために，一符号当りの平均比較回数をほぼ 6 程度に収めることが可能である．そのような最長一致記号系列探索手法により，符号化に要する比較回数を参照部バッファ長 N に依存せずに (ほとんどの場合) 一定値に抑えることができ，符号化におけるスルー・レートを大幅に向上することが可能となる．本研究で考案した並列処理アーキテクチャはこの探索法を実現するものである．

比較処理の計算量を，1 次元シストリックアレイの場合と本提案アーキテクチャの場合とを比較すると，それぞれ

・符号個数×(2N−1)　(一次元)

・符号個数×平均比較回数　(本提案)

となる．実際にある C ソースプログラム (198506[byte]) についての計算量を概算したところ (N = 128)，全符号数 ： 88106 [個]，平均比較回数 ： 2.253 [step] より，1 次元シストリックアレイでは約 2250 万 [step]，比較回数を L 程度で抑えると約 20 万 [step] となり，全処理回数を約 1/100 程度に削減することが可能となることがわかる．

4 高速 LZ77 符号化アーキテクチャ - PAHL-C

4.1 PAHL-C の構成

本研究で考案したハードウェアアルゴリズムを実現する並列処理アーキテクチャ - PAHL-C(Parallel Architecture for High-speed Lempel-ziv data Coding) を図 4(参照部バッファ長 8 の場合) に示す．なお，PAHL-C で実現している処理機能は基本的な LZ77 符号化であり，圧縮率は LZ77 符号化で決

まるものに一致する.

PAHL-C では図 4に示すように，参照部バッファの各要素 (一文字分) を出発点とした記号系列との一致判定部（cell1 部）と，各 cell1 より得られた一致長決定信号を入力とする最長一致記号系列判定部（2 分木構造に結線された cell2 部），最長一致記号系列長計数用カウンタ (counter)，一致長計数および全 cell1 の一致判定を監視する部分 (all-match-cell) とスライド窓バッファ及び各種バッファから構成されている．また入力記号ビット幅は 8bit，出力符号ビット幅は参照部バッファ長 Nから定まり $\lceil \log_2 N + \log_2 N + 8 \rceil$ [bit] となる.

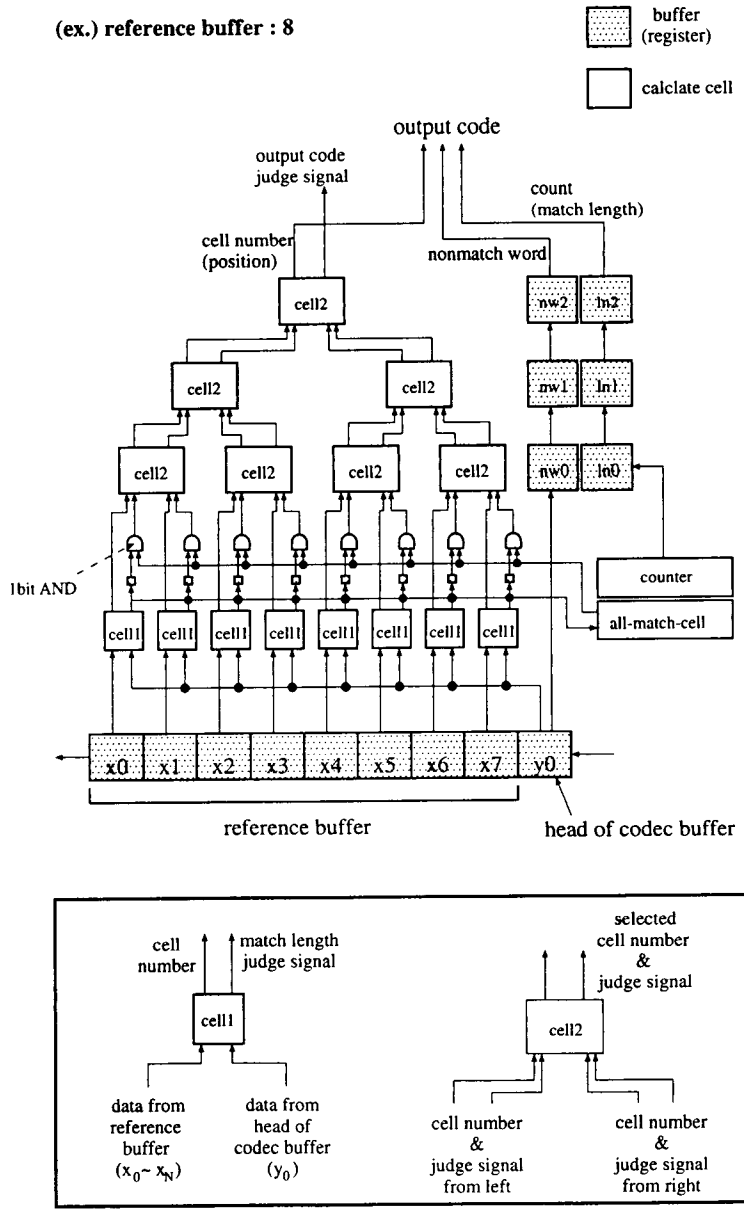

図 4: 提案アーキテクチャ - PAHL-C の概略図

4.2 PAHL-C での処理

PAHL-C のアーキテクチャに基づいて，ハードウェアアルゴリズムの概要を示す.

1．スライド窓バッファ内データ，出力符号用バッファの初期化
2．各セル内レジスタ，カウンタの初期化
3．各 cell1 にスライド窓バッファ内データの転送
4．各 cell1 での一致判定
   ・ if 一致 ： 一致長決定信号として 0 を出力
   ・ if 不一致 ： 一致長決定信号として 1 を出力．以後出力 1 の継続
5．all-match-cell で全 cell1 で一致長決定が終了したか否かの判定：
   各 cell1 からの一致長決定信号の AND
6．スライド窓バッファ内データを 1 シフト
7．各 cell1 の結果を cell2 に転送
8．各 cell2 で条件に合うデータを出力：
   一致長決定信号 (最下部では一致長決定信号の NOT と all-match-cell からの出力値の AND) が 1 のもののデータを選択し出力．双方が 0 または 1 の場合，左側 (位置の値が小さいもの) を出力
9．最終段階の cell2 から所望符号の出力
10．カウンタ部の動作及び出力：
   カウンタ部は一致長をカウントし，ステップ 5 の判定で 1 になったとき，カウンタの値を上部のシフトレジスタに送り，カウンタをリセットする．

このアルゴリズムは，一つの符号化で上記の一連の処理を並列に実行する．すなわち，ステップ 3 から 10 までは全ステップを並列に実行し，ステップ 5 で一致長決定信号がすべて 1 となる判定がでる度に全 cell1 およびカウンタを初期化 (ステップ 2 の動作) し，ついで次の符号化処理を実行する．

5 PAHL-C の論理設計

高位論理合成システム PARTHENON を用いて PAHL-C の論理設計，動作検証，論理合成を行った．また PARTHENON による論理合成時の条件データとしては，demo 社のデモ用ライブラリ（条件データ 1.5 μ CMOS テクノロジの実ライブラリ）を使用した．

PARTHENON による論理設計，及び動作検証により，PAHL-C が実際に実現可能であることを確認した．また，様々な N での論理合成結果から消費電力，実装面積，ゲート数，最大遅延時間を求めた．

理論的には，参照部バッファ長が N のとき，セル数は O(N) の数だけ必要となるので，消費電力，実装面積，ゲート数は N に比例し，cell2 が二分木上に結線されているため，回路全体での遅延時間は $\log_2 N$ に比例することになる．消費電力，実装面積，ゲート数は定性的には同様の結果を示す．以下では，様々なバッファ長での論理合成により求められたゲート数，最大遅延時間について考察する．

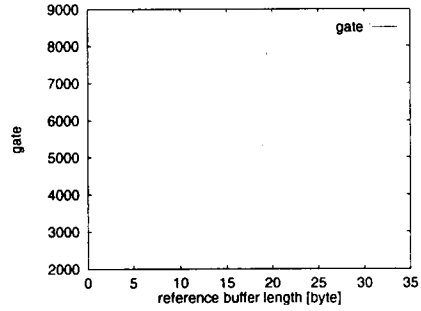

図 5: 各参照部バッファ長でのゲート数

5.1 PAHL-C のゲート数

本アーキテクチャの回路規模は参照バッファ長 N に比例する．実用的な符号化アーキテクチャの実現のために N をどの程度の値にするべきかが大きな問題となる．一般には N がより大きいほど圧縮率は向上すると言われるが，それでもある程度の長さ以上になると圧縮率は低下する．圧縮率の点から見た場合，実用的な値としては N=8192 (かつ，符号化部バッファ長 M=16 ないし 32) 程度で比較的良い結果が得られるという報告がある [8]．実際，現在一般で使用されている LZ77 符号化法では N=8192 (M=32) 程度の値が用いられている．しかし，情報圧縮／復元をメモリとハードディスク間でのみ利用するなど，ある用途へ特化したものとして使用する場合は，N を小さくしても実用的価値がある．最大圧縮率の 8 割程度のデータ圧縮が達成されればよいものと仮定すると，様々な形式のデータを用いて調査した結果，データの形式によってばらつきはあるものの，N=1024 または 2048 (M=32) 程度という結果を得た．そのような特定用途のため PAHL-C を ASIC により実装することを考えた場合，N=1024 以上でゲート数が実現可能な規模であるかを調べる必要がある．

参照部バッファ長 N を変化させて得られた論理合成結果のゲート数を図 5に示す．図 5の直線の傾きからわかる比例定数をもとにいろいろな N の場合のおおよそのゲート数が予測できる．N=4096 では，ゲート数は約 100 万程度は必要になるものと予測される．N=1024 または 2048 では，必要なゲート数としては約 20 万〜30 万ゲート程度と予測され，容易に実現できるサイズとなる．

5.2 各セルの遅延時間

PAHL-C では各セルが全てシストリック的に並列動作し，パイプライン的にデータ処理を行う．従って，実際の動作速度を議論する場合には，回路全体としての遅延時間ではなく，全体を構成している個々のセル単位での遅延時間について考える必要がある．遅延時間は遅延パス終点への立ち上がりパルスの到着時間とする．各セルにおける遅延時間は各々のセルが一回の動作に最低必要な時間である．回路全体としての一回の並列動作の時間は，各々のセルのうちで最も動作速度の遅いセルが基準となる．

様々な N についての各セルの遅延時間の結果を図 6に示す．これに示されているように，cell1 では N の大きさにかかわらず個々のセルの構成は変化しないので遅延時間は N に関係なく一定である．また cell2 および counter に関しては，それぞれ入出力ビット幅が $\log_2 N$，$\log_2 M$ となるため N, M に依存することになるが，実用上の大きさに関してはほとんど一定であるものと見てよい．all-match-cell

については，その構成が 2 入力 1 出力の AND の二分木配置であるため，一回の動作に段数 ($\log_2 N$) 分の 1bit AND 処理を実行するが，1bit AND 自体の遅延時間が小さい (本使用データでは AND 部のみの固定分で約 1.3[ns]) ため，他のセルの処理と比較してそれほど遅延時間を要するものになっていない．理論的には，N=4096 で遅延時間は 1.3 × 12=15.6[ns] となるが，図 6では約 25[ns] となっている．all-match-cell で実現している処理は，実際には wired-or(and) の手法等により，全ての AND(OR) 素子を単一線に並列に接続する形で実現でき，さらに短縮が可能と考えられる．

PAHL-C では，cell1，cell2，counter，all-match-cell，スライド窓バッファが全てシストリックに同期動作することを理想としていた．しかし現時点では完全な並列動作は実現されておらず，全 cell1 で 1 ステップ動作した後に all-match-cell に信号が送られ，all-match-cell が動作するようになっている．従って動作時間という観点で考えた場合は cell1 と all-match-cell は一つのものとして議論しなければならない．図 6には cell1 と all-match-cell の遅延の加算されたものを同時に示してある．

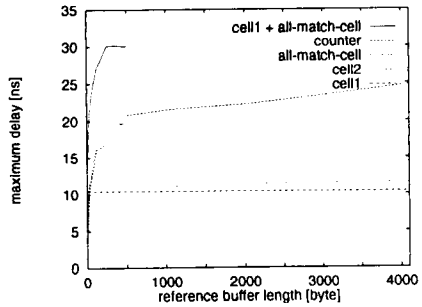

図 6: 各セルの遅延時間

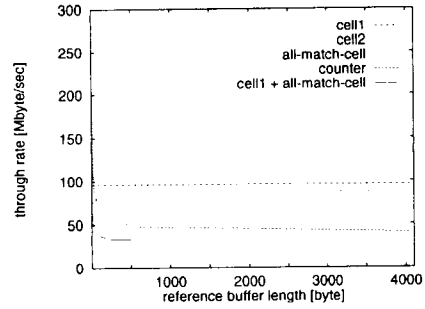

図 7: 各セルのスルー・レート

5.3 各セルのスルー・レート

PARTHENON による論理合成で得られた各セルでの最大遅延時間（図 6）より，符号化回路の最大スルー・レートを計算することができる．図 6より得られるスルー・レートを図 7に示す．

図 7より，今回用いた条件データでの論理合成による結果から得られた性能としては，実用的な参照バッファ長では約 20〜25Mbyte/sec 程度の最大スルー・レートを実現でき，1 文字（1 記号）を 1byte と仮定すると，約 2000 万〜2500 万文字/秒程度のスルー・レートと予測される．

表 1に，今回の条件データでの予測される性能の大枠を示す.

表 1: 予測される PAHL-C の性能

参照部バッファ長 [byte]	4k	1k
消費電力 [W/MHz]	3	1
実装面積 [mm^2]	900	200
ゲート数	100 万	20 万〜30 万
最大遅延時間 [ns]	40〜50	40〜50
スルー・レート [Mbyte/s]	20〜25	20〜25

6 LZ77 復号化アーキテクチャ - PAHL-D

6.1 PAHL-D の構成

LZ77 復号化処理において，所望の復元記号列の出力のための基本処理は，与えられた符号 (位置 P, 長さ L, 未一致記号 W) から，参照部バッファ上の P から始まる長さ L の記号列と未一致記号を復元記号として取り出す処理である.

この LZ77 復号化について，図 8に示す高速処理可能な LZ77 復号化並列処理アーキテクチャ - PAHL-D を考案した．本アーキテクチャは，参照部バッファをなす部分と，一符号に相当する記号列を出力するためのカウンタとから構成している．符号が与えられると，符号から P, L, W が抽出され，一回の動作で参照部バッファの位置 P 内の記号の選択とバッファ内記号のシフトを並列に実行する．このとき選択された記号はバッファ内データのシフトと同時に参照部バッファの最後の部分に追加される．これを L 回繰り返すことで，参照部バッファ上の位置 P に相当する部分から出力される記号により長さ L の所望記号列を取り出すことができる．その後未一致記号 W を付加し，復元記号列が得られることになる．PAHL-D では，一回の動作で一記号を出力し，長さ L+1 の復元記号列を得るのに L+1 回の処理を実行する.

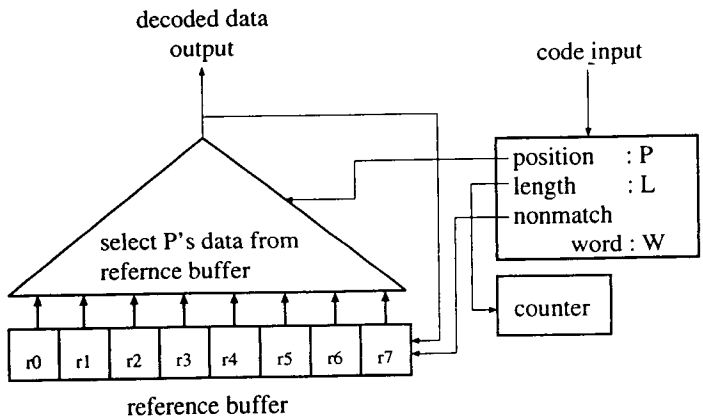

図 8: PAHL-D の構成

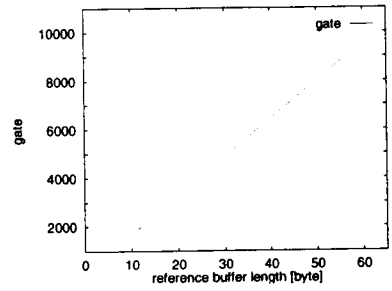

図 9: PAHL-D のゲート数

6.2 PAHL-D のゲート数と動作速度

PAHL-D についても PAHRTHENON により論理設計，論理合成を行い，実現可能であることを確認した．

PAHL-D のゲート数を参照部バッファ長 N を変化させた場合について求めた結果を図 9に示す．

このアーキテクチャでの主な処理は，シフトレジスタ内でのデータ (記号) 転送と，出力記号数を制御するためのカウンタ動作である．各々は並列動作するので，符号化の場合と同様に，復号化処理において一回の動作に要する時間は，各々の一回の動作に要する時間 (=遅延時間) から決定できる．PAHL-D の各々の処理部分での遅延時間を参照部バッファ長 N を変化させて求めた結果を図 10に示す．さらに遅延時間からスルー・レートを求めた (図 11)．

図 10からも確認できるように，シストレジスタは，常に 8bit データをその前後のレジスタとのみ扱うので，動作における遅延時間は N には依存しない．また，PAHL-C における議論から最長一致長は 32 以下として十分であり，カウンタは 5bit 程度の固定長として考えることができる．参照部バッファ上の所望のデータ選択については，本提案アーキテクチャでは 1bit の AND 処理が $\lceil \log_2 N \rceil$ 段必要となる．よってこの部分において N に依存した処理時間となるが．1bit AND 自体の処理は非常に小さいので，データ選択部の処理が他に比べて格段に大きくなることはない．

表 2に PAHL-D の予測される性能を示す．

表 2: 予測される PAHL-D の性能

参照部バッファ長 [byte]	4k	1k
消費電力 [W/MHz]	2	0.5
実装面積 [mm^2]	500	80
ゲート数	60 万〜70 万	15 万〜20 万
最大遅延時間 [ns]	25〜30	25〜30
スルー・レート [Mbyte/s]	30〜40	30〜40

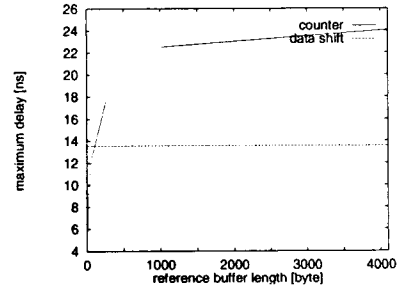

図 10: PAHL-D 内処理部の遅延時間

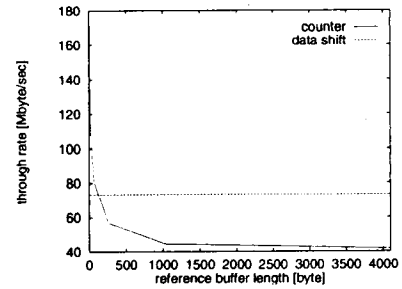

図 11: PAHL-D 内各処理部のスルー・レート

6.3 PAHL-C ／ PAHL-D の統合

PAHL-C 及び PAHL-D はどちらも主要な構成要素として参照部バッファとカウンタを含む．すなわち PAHL-D に存在する構成要素は全て PAHL-C 内に存在することとなる．このことを利用して，符号化／復号化を統合した高速 LZ77 符号化／復号化アーキテクチャ - PAHL を設計した．符号化と復号化を統合するにあたり，入出力ポートは符号化／復号化で共用として，端子の増加を抑えた．アーキテクチャの基本構成は PAHL-C とほぼ同じであり，その性能ではゲート数の面で 500〜600 程度の増加があったが，PAHL-C のときとほぼ同様の性能を実現できた．

7 むすび

本研究では，多様な情報源に対し良好な圧縮率を得ることのできる，データ符号化／復号化法の一つである LZ77 符号化／復号化法のハードウェアアルゴリズムを考案し，並列アーキテクチャ - PAHL-C ／ PAHL-D を設計し，性能を評価した．

PAHL-C では，符号化過程で最も計算量を要する最長一致記号系列探索部において，その処理過程での比較回数の冗長性をできるだけ削減することにより，符号化時におけるスルー・レートを大幅に向上させている．また各部分の並列度を増すことによって，より高速な符号化を実現している．PAHL-C の具体的な動作，性能を確認するため，PARTHENON を用いて論理設計，論理合成等を行ない，本研究での条件データについて，約 20〜25Mbyte/sec （2000 万〜2500 万文字/秒）の最大スルー・レートが得られることを示した．PAHL-D についても，論理設計，論理合成を行い，最大スルー・レート等の性能として PAHL-C とほぼ同様の結果が得られることを示した．本研究で提案し

た符号化／復号化アーキテクチャを統合した形でのLZ77符号化／復号化アーキテクチャ - PAHLが設計でき，その性能が同程度であることを述べた．

現在普及しているLZ77符号化法は，基本手法に様々な改良が加えられているもの(LZSS法など)が一般的であり，それらが様々なデータ圧縮アプリケーション等に利用されている．本研究で考案したアーキテクチャは，LZ77符号化／復号化の基本手法を並列処理として実現するものであるが，その高スルー・レートを保ちつつLZSS法などより一般的な手法に対応した，より汎用性のあるアーキテクチャに拡張することは今後の課題である．

本研究は東北大学大学院学生 藤岡豊太君に負うところが大きい．記して感謝します．この研究を行うにあたり，ハードウェア設計・支援システムであるPARTHENONを使用させて頂いたNTT情報通信網研究所の関係各位に感謝します．

最後に、本研究に対してご援助頂きました石田(實)記念財団に深く感謝いたします。

参考文献

[1] N. Ranganathan and Selwyn Henriques, "High-Speed VLSI Designs for Lempel-Ziv-Based Data Compression," IEEE Trans. Circuit & Syst. , vol.40, no.2, pp.96–106, February 1993.

[2] R. Zito-Wolf, "A Broadcast/Reduce Architecture for High-Speed Data Compression," Proc. IEEE Int. Symp, on Parallel and Distributed Processing, Dallas, TX, Dec. 1990.

[3] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," IEEE Trans. Inform. Theory, vol. IT-23, no.3, pp.337–343, May 1977.

[4] J. Ziv and A. Lempel, "Compression of Individual Sequences via Variable-Rate Coding," IEEE Trans. Inform. Theory, vol. IT-24, No.5, pp.530–536, Sep. 1978.

[5] James A. Storer and John H. Reif, "A Parallel Architecture for High-Speed Data Compression," Journal of Parallel and Distributed Computing 13, pp.222–227, 1991.

[6] Amar Mukherjee and N. Ranganathan, "MARVEL: A VLSI Chip for Data Compression Using Tree-Based Codes," IEEE Trans. VLSI Sys., vol.1, no.2, June 1993.

[7] 山本博資, "ユニバーサルデータ圧縮アルゴリズム：原理と手法," 情報処理, vol.35, no.7, pp.600–608, July 1994.

[8] 植松友彦, "LZ77 符号," 文書データ圧縮アルゴリズム, pp134–141, ＣＱ出版株式会社, 東京都, 1994.