Vol.2012-ARC-201 No.17
2012/8/2

サービス指向型ルータにおけるコンテキストスイッチを利用したGZIP復号処理機構の実装

穂川 大伍[†1,a)]　石田 慎一[†1,b)]　西 宏章[†1,c)]

概要：複数のデータコンテキストを管理するGZIP復号処理機構をハードウェアで実装する．我々が提案しているサービス指向型ルータはコンテンツベースのルーティングを目指しており，このルータ上で取得したパケットのペイロードを解析する必要がある．HTTP1.1においてパケットはGZIP圧縮されているためGZIP復号処理が必要である．さらに，インターネット上のコンテンツは元々大きなサイズのデータをEthernetを通して送信するために，いくつかのパケットに分割され，さらにそれらのパケットは順不同で相手先に届けられる．提案するGZIP復号処理機構はワイヤレートでかつ複数のデータを並行して処理できるコンテキストスイッチを用いた機構となるように設計した．

キーワード：サービス指向型ルータ，GZIP復号処理機構，コンテキストスイッチ

1. 導入

インターネットによる情報通信手段は世界中で広く使われるようになった．今や情報収集にインターネットを用いることは常識となり，さらには自らを発信源とし情報を送信することも多くなった．人々はiPhoneやAndroidといった端末を用いFacebookやTwitterなどのサービスを利用して，日々たくさんの情報をやり取りしている．こうしてインターネット上を流れる情報は益々増えて行く．

このような膨大な情報を扱うために，新しいタイプのルータが求められている．従来ルータはネットワークの中心に位置し，いかにコンテンツを目的地まで正しく速く届けるかにのみ力が注がれて来た．そのためTCP/IPのヘッダ情報だけを解析することで，コンテンツの送信元，受信先などを取得し，インターネット上の経路を決定していた．しかし，人々はインターネット上で享受できるサービスについてはよりユーザリッチなものを求めるようになって来ている．例えば，アマゾンのオンラインストアにおいてはユーザの購入履歴や検索履歴などの情報をもとに，よりユーザが求めそうなものを提供する，レコメンデーションサービスを提供している．このようなサービスをあらゆるデータを扱うことができるルータによって提供できれば，よりリッチな環境が実現できると言える．

我々は新しいタイプのルータである，サービス指向型ルータ(SoR)[2], [3]を提案している．このルータはパケットのペイロードを解析することでコンテンツベースのルーティングを可能にしている．コンテンツがインターネットを流れる際にはTCP/IPヘッダやその他のプロトコルのヘッダがともなう．例えば普段使用しているサービス,Googleの検索エンジンを利用しているときもインターネットプロトコルを利用しており，ヘッダが存在する．その後，サーバはコンテンツをヘッダ情報とともに送信する．コンテンツをリクエストしたクライアントコンピュータにおいてそのヘッダ情報を解析し，コンテンツを利用することが出来る．

しかしながら，ネットワーク上を流れるストリーミングはGZIPという圧縮アルゴリズムで圧縮されることがある．さらにリンク層であるEthernetにおいてそれらのデータはより小さなパケットに分割されることがある．多くのユーザはこれらの仕組みについて普段意識することはないが，ルータにおけるコンテンツ解析では意識して処理しなければならない．こうしたGZIPで圧縮され分割されたデータは分割され順不同にネットワーク上を流れる．それで,SoR上でパケットを解析するためにはもともと一つのストリームであった複数のパケットをGZIP復号処理する必要がある．分割されたパケット毎の処理をコンテキストスイッチにより効率よく処理できると共に必要メモリ量の削減が行えると考えられる．以上のことが提案したコンテキストスイッチを利用したGZIP復号処理機構の研究を始めた動機

[†1] 現在，慶應義塾大学
Presently with Keio University
[a)] hogawa@west.sd.keio.ac.jp
[b)] sin@west.sd.keio.ac.jp
[c)] west@sd.keio.ac.jp

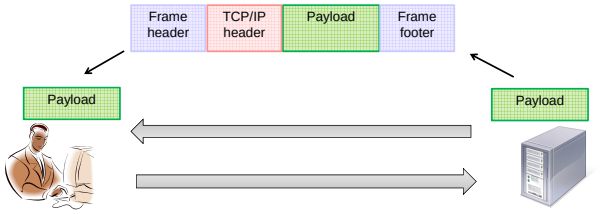

図 1 データフォーマット
Fig. 1 a Data Format.

である．このようなコンテキストスイッチを利用することで, 複数の分割されたパケットを一つのストリームと同じように復号することが出来る．

この論文は以下のような構成である．第 2 章ではネットワークとサービス指向型ルータの説明を行う．第 3 章では我々が提案した GZIP 復号処理機構を説明する．第 4 章で我々が実装した GZIP 復号処理機構について評価を行った．最後に結論を第 5 章で述べる．

2. 背景

2.1 HTTP1.1, TCP/IP, ETHERNET

インターネット接続のほとんどは HTTP に基づいている．最も広く利用されている通信のプロトコルは TCP/IP と呼ばれ, ネットワークトラフィックの約 8 割を占めている．TCP/IP パケットには TCP/IP ヘッダが付け加えられ, 宛先 IP アドレスに送られる．大きなデータを送信する際は, フレームという単位に分割され, それぞれ TCP/IP ヘッダとともにフレームヘッダ, フレームフッタが付け加えられる．**図 1** は簡単なインターネット接続について表している．普段なにかウェブページをブラウズする際にもメールを送受信する際にもインターネットプロトコル (IP) が使用されている．送信元ホストから宛先ホストへ, いくつかの IP ネットワークを利用して, ホストのアドレス指定からデータのルーティングが行われる．

HTTP1.1 ではデータ部は GZIP 圧縮アルゴリズム [6] によって圧縮される場合がある．その後ネットワークで転送される．ネットワークトラフィックは増え続けているが, インターネットを使って一度に転送する量はネットワーク通信バンド幅の制限を受ける．今後さらにトラフィックは増えるものと予想されるが, データを多く転送するために HTTP1.1 ではデータを圧縮する仕組みが提供されている．圧縮には一般的に GZIP アルゴリズムが用いられている．このアルゴリズムは圧縮率がおよそ 30〜40%となり, 圧縮速度との関連からもバランスのとれた圧縮アルゴリズムである．これによりデータ転送の前に圧縮を行うことで, より多くのデータ転送を可能にしている．GZIP アルゴリズムは誰でも使うことの出来るフリーのアルゴリズムであり, 古くから,UNIX システムなどで用いられて来た．

我々の研究室のネットワークを解析したところ, これら

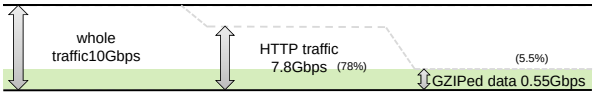

図 2 利用したトラフィックの傾向
Fig. 2 a Trend of the Used Traffic.

のネットワークトラフィックの中には HTTP トラフィックが含まれており, その割合は 78%である．同様に**図 2** に示すようにもし 10Gbps のネットワークを想定すれば, 全トラフィック中では GZIP 圧縮されたデータは 0.55Gbps を占めることとなる．このことから GZIP 復号処理機構はおよそこのスループットを満たさなくてはならず, 高速に処理する機構が必要である．

パケットはさまざまな通信経路を通って宛先アドレスに届けられる．最も広く用いられるリンク層は Ethernet であり, これは LAN として使われている．Ethernet の最大転送ユニットはたったの 1,500 バイトである．**図 3** にこのようなフラグメンテーションの例を示す．もしパケットのサイズが 1,500 バイト以上であれば, それらのパケットは 1,500 バイト以下のいくつかのバイトに分割され, それらの分割されたパケットはフレームと呼ばれ, それぞれフレームヘッダ, フレームフッタというものが付け加えられ, その後に送信される．

こうしたフラグメンテーションによって分割されたパケットはそれぞれ IP ヘッダなどの付加情報が加えられる．パケットは分割され, 付加情報をもとに個別に順不同に転送される．別々の経路を通って来たパケットは最終的に宛先アドレスにおいて再び一つのデータになる．すなわち, あるストリームに属するパケットは, 宛先アドレスにそれらのパケットが全て到着するまでは, それぞれ独立にネットワーク上を流れている．それらの独立したパケットを解析するためには, どのストリームに所属するのかという情報が必要となる．

2.2 サービス指向型ルータ (SoR)

近年ではインターネット技術は著しく発達し, 人々はその技術や手法を日常的に使うようになった．最近ではビジ

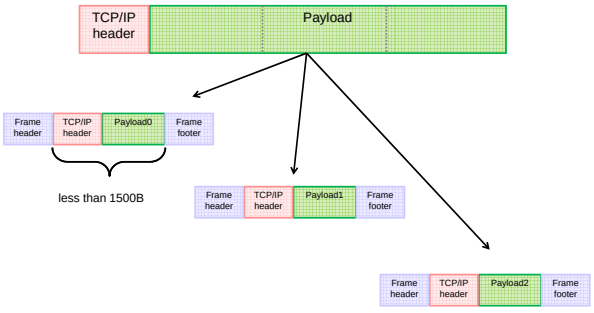

図 3 Ethernet におけるフラグメンテーション
Fig. 3 a Fragmentation at the Ethernet.

ネスに置いても学術分野においても知識や情報を共有するためにインターネットを用いている．インターネットは情報の送信や情報の受信にとても便利であり，今や多くの人々にとってそれらにインターネットを用いることは常識となっている．

インターネットは何百万人という人々が利用するデータの集まりとなっている．インターネットを利用するのに様々なサービスや機器が存在する．例えば，ほとんどの人々は自分自身の携帯やスマートフォンを持ち，いろいろな情報をインターネット上で共有している．それらのデータはインターネット上のルータを介して人々の間でやり取りされる．ルータは膨大なデータを管理し，それらのパケットの経路を決定する．

ネットワークルータはさまざまな独立したネットワークをつないで一緒にし，データを宛先アドレスに転送する．たくさんのコンテンツリッチなデータを扱うために新たなタイプのルータが必要とされている．我々は新しいタイプのルータ，サービス指向型ルータ (SoR) を提案している．このルータは従来のルータが提供することの出来なかったコンテンツベースのルーティングを実現することの出来る．SoR はルータそれ自身から API を用いてエンドユーザへサービスを提供する．このルータはデータ集合を受動的に取得することが出来るところに利点がある．一方現在のエンドツーエンドのシステムではデータを取得するためには能動的に取得する必要がある．能動的なデータ収集においてはエンドホストが必要なデータを取得するにはウェブクローラなどを用い，他のホストにアクセスするしかない．この能動的な情報取得には多大な時間がかかり，さらに取得できる情報は限られている．リアルタイムのインターネットの情報を取得するために，頻繁なウェブクローリングは時にネットワークの混雑を引き起こす．サービス指向型ルータにおける受動的なデータの収集はリアルタイムな情報の取得を可能にし，現在のインターネットの情報を正確に反映できる．

SoR においてデータを解析するためには GZIP 復号処理機構が必要である．パケットはエンドホストサーバにおいて GZIP アルゴリズムで圧縮されることがある．その圧縮されたデータは従来，クライアントのコンピュータで復号されていた．加えて，インターネット上にはさまざまな種類のデータが存在し，それらのデータは個別にネットワーク上で転送される．従って我々はコンテキスト情報を管理することなしには完全に復号することは出来ない．さらに前章で述べたように 10Gbps のネットワークに対応する場合,GZIP 圧縮されたデータのスループットは 0.55Gbps である．したがってサービス指向型ルータで GZIP 復号処理を実現するためにはこのようなスループットを満たし，また様々なデータを並行に復号できるような機構が必要である．

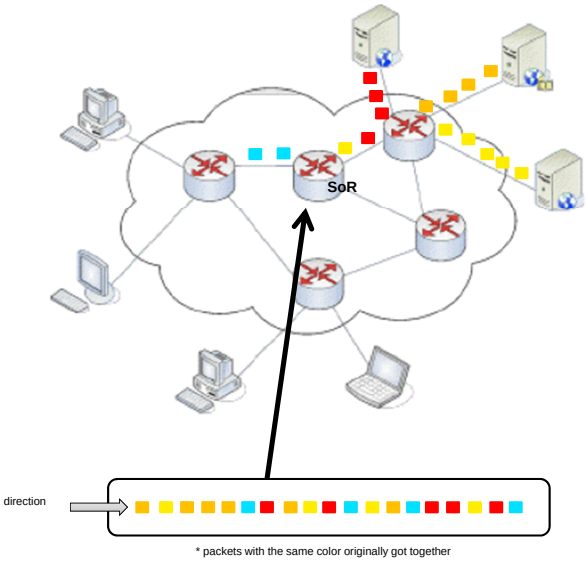

図 4 サービス指向型ルータ
Fig. 4 Service-oriented Router.

3. GZIP 復号処理機構

我々は高速かつ並行に GZIP 圧縮されたデータを復号処理するハードウェア実装を提案する．上記で述べたようにこの復号処理機構はワイヤレートを満たすような速度で高速に処理できる必要がある．そのためにこの提案した復号処理機構は高速に処理する必要があり並行処理できるようにハードウェア実装する．さらに複数のユーザの GZIP 圧縮されたデータを扱うために，コンテキスト情報を管理している．

3.1 並行処理

処理時間のかかるプロセスの一つにハフマンアルゴリズムがある．ハフマン木を構成する際にソートプロセスが存在する．ソフトウェアでのハフマン復号プロセスでは，ソーティングにかかる時間は少なくとも $n \log n$ かかる．しかしながら，ハードウェアでは奇遇転地ソート [5] という単純なアルゴリズムを用いることで，より高速にソーティングすることが出来る．

図 5 はソートプロセスの例を示している．この例では 3 回でソートが完了するなど，ソーティングのオーダーは n となる．

奇遇転地ソートとは単純なアルゴリズムで，バブルソートに似ている．バブルソートでは最初のフェイズは隣同士の値を比較し，交換する．その後，一番大きな値が列の最後に出現する．このプロセスを繰り返すことでソートが完了する．しかし，このバブルソートはシーケンシャルにソートの各ステップを踏む必要がある．奇遇転地ソートはパイプラインをもちいてバブルソートを最大限並行に行えるようにしたものである．一回のフェイズで行う入れ替えを最

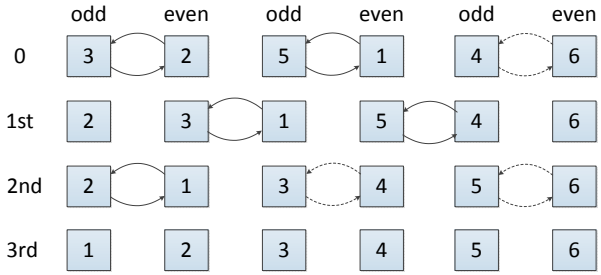

図 5 ハードウェア奇遇転置ソートの例
Fig. 5 Odd-Even Transposition Sort.

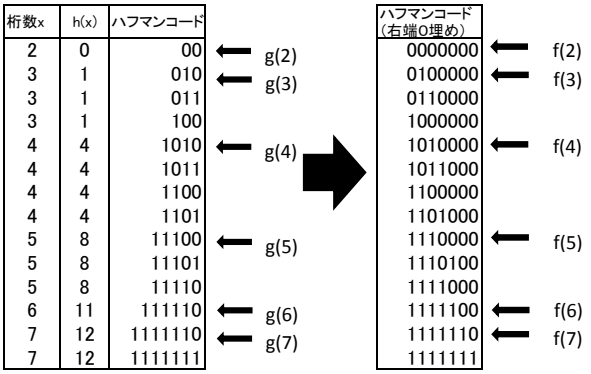

図 6 パラレルデコーディングの改良アルゴリズム
Fig. 6 an Improved Algorithm of Parallel Decoding.

大限並行に行うことでソーティングの速度を高めることが出来る．ハードウェアで実装することで, この並行プロセスを同時に行い, より効率的にソートを行うことが出来る．

3.2 パラレルデコーディングの改良アルゴリズム

先行する GZIP 復号処理 [4] の手法において, パラレルデコーディングと呼ばれる手法が提案されている．以下ではこの改良アルゴリズムを示したい．まず, 復号しようとするビット列（15 ビット）を b_{max}, ハフマンコード長 x における最小のハフマンコードを $g(x)$, $g(x)$ の右端を 0 埋めしたものを $f(x)$ とする．$f(x)$ の具体例を**図 6** に示した．また $g(x)$ のハフマンテーブルの先頭からの距離を $h(x)$ で表す．

このとき, i を整数として,

$$f(i) \leq b_{max} < f(i+1) \tag{1}$$

が成り立つならば, b_{max} の先頭 i ビットがハフマンコードのテーブルに存在すると言える．ここで, b_{max} の先頭 i ビットを $b(i)$ と表すことにする．

例えば先ほどの図 6 において,

$$b_{max} = 1011101 \tag{2}$$

とすると,

$$f(4) \leq b_{max} < f(5) \tag{3}$$

であり, b_{max} は先頭 4 ビットつまり $b(4) = 1011$ はハフマンコードのテーブルに存在している．

このとき

$$y = b(i) - g(i) + h(i) \tag{4}$$

と新たにマッピングすると, y の値は,

- 辞書を用いてリテラル／長さを復号する場合は 0〜285 の値
- 辞書を用いて距離を復号する場合は 0〜29 の値

をただ一つ持つことになる．すなわち GZIP 圧縮されたデータを復号するのに必要な辞書は 0〜315 のアドレスを持つメモリを用意すれば十分である．ハードウェアで実装しているため, 上記のマッピングはパラレルに処理され 1clock で完了する．

3.3 コンテキストスイッチを利用した GZIP 復号処理

提案する GZIP 復号処理機構ではコンテキストスイッチを用いている. ここでコンテキストスイッチとは処理しようとするパケットのコンテキスト情報を管理するシステムである．コンテキスト情報とは, そのパケットがどのストリームに所属しているのか, またそのパケットを GZIP アルゴリズムによって復号する際に用いる辞書のことである．ネットワーク上には膨大な数のストリームが存在し, それぞれのストリームは分割され, 個々のパケットとなり, サービス指向型ルータ上を転送されて行く．その際, パケットはどのストリームに属するのかという情報がヘッダ部に記載されている．しかし, 個々のパケットはもともと GZIP 圧縮されているものが存在し, それらの復号処理に用いる辞書は共有されている．別々に転送されているこれらのパケットを正確に GZIP アルゴリズムで復号するために, それぞれの辞書をメモリに格納し, また前回までの処理経過について保存しておく必要がある．

こうしたコンテキストスイッチは単純な仕組みで実現できる．GZIP 復号処理が処理中のパケットを復号し終わったとき, それらに利用していた辞書と, 処理途中のパケットをメモリに保存する．一つのストリームに対して, コンテキストメモリに対するポインタを 2 つ保持することで, さまざまなデータに対応することが出来る．その二つのポインタとはコンテキストメモリ全体の中でそのコンテキスト情報がどの位置に存在するかを示すポインタと, そのポインタからどの位置を取得すればいいかのポインタである．それぞれのストリームに応じてそのポインタを選択することで簡単にロードすべきコンテキストを切り替えることが出来る．**図 7** にコンテキストスイッチの概念図を示した．

それぞれの辞書データは 316×16 ビット必要であり, さらにそれ以外のメモリアドレス計算用のデータと前回処理したパケットの復号しきれなかった残りバイトを合わせて 66×16 ビットを保存する必要がある．すなわち一つの GZIP ストリームに対して保存する必要があるコンテキス

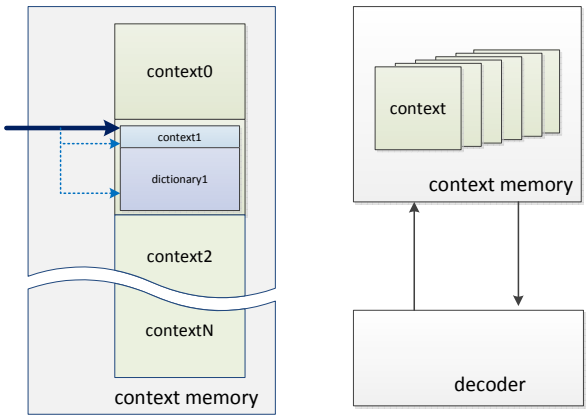

図 7 コンテキストスイッチ
Fig. 7 Context Switch.

表 1 任意のタイミングにおける平均 GZIP ストリーム数と必要なコンテキストメモリサイズ
Table 1 an Average number of GZIP Stream and Necessary Context Memory Size at Some Point.

timeout(s)	GZIP ストリーム数	メモリサイズ (MB)
600	1.40×10^5	103.07
300	7.00×10^4	51.54
60	2.63×10^4	19.36
10	5.25×10^3	3.87

ト情報のサイズは

$$316 \times 16bit + 66 \times 16bit \ = \ 772B \tag{5}$$

となる. ただ大量のメモリ資源を用意し,GZIP 復号処理を行って行くのではなく, こうしたコンテキストスイッチを利用することで, 省資源でインターネット上を流れるデータを復号処理することが出来る.

さらに実際にこのようなコンテキストをメモリにいくら保持すればいいのか評価するため, 通信が終了したと見なす時間 (timeout) を変化させ, ある任意のタイミングにトラフィック中に存在する平均の GZIP ストリーム数を解析した. その結果を**表 1** に必要なコンテキストメモリサイズも合わせて示した.

4. 実装と評価

4.1 環境

提案した GZIP 復号処理機構をシミュレーション, 合成評価する環境を**表 2** に示した. 今回用いた我々の研究室内の 2011 年 12 月 5 日〜12 日のトラフィックデータに関して**表 3** に示した.

4.2 シミュレーション結果

4.2.1 Simvision による波形表示

今回 Google.com にアクセスした際のパケットを用い, 実際にテスト用データを作成し, シミュレーションを行った.

表 2 シミュレーション・合成環境
Table 2 an Environment of Simulation and Synthesis.

記述言語	Verilog-HDL
論理シミュレーション	Cadence NC-Verilog LDV5.7
波形表示ツール	Cadence Simvision
ASIC 合成ツール	Synopsys Design Compiler X-2005.09
ASIC 合成用ライブラリ	FreePDK OSU Library[1] (NAND2 ゲート面積: 0.798 μm^2)

表 4 波形より読み取ったレイテンシ
Table 4 Latency from Waveform.

辞書作成におけるレイテンシ	1,279clock
メモリへの辞書転送におけるレイテンシ	68clock
メモリからの辞書転送におけるレイテンシ	48clock

表 5 合成結果（全体）
Table 5 Synthesis Result(Whole).

回路規模 (メモリ部位除く)	$0.14mm^2$
動作遅延	2.99ns
最大動作周波数	334.45MHz
ゲート数（NAND2 ゲート換算）	174,901

図 8 に復号された波形の様子を示した.

また,GZIP 復号処理におけるレイテンシを**図 9** から見積もり, **表 4** に示した. ここでいうレイテンシとは復号された文字が出力される以前の辞書作成, 辞書転送, また文字出力後の辞書転送にかかるクロック数のことである.

4.3 合成結果

Verilog-HDL ハードウェア記述言語を用いて記述したモジュールを Synopsys Design Compiler X-2005 を用いて ASIC 合成を行った. モジュール全体を合成した結果を以下の**表 5** に示し, 個々のモジュールについて**表 6** に示した.

ここでモジュール全体のゲート数に関して,FreePDK ライブラリにおいて NAND2 ゲートの面積が $0.798\mu m^2$ であることから, NAND2 ゲートを基準に計算を行った.

また以下の**図 10** に全体のアーキテクチャとそれぞれのモジュール単体でのゲート数を示した. このアーキテクチャでは主に 10 のブロックが存在し, それぞれ独立に平行にパイプライン処理が行われる. 図の上部にある 3 つあるブロックはメモリであり, それぞれ入力用メモリ, コンテキスト用メモリ, 出力用メモリである.

4.4 スループット

以上のデータから計算されるスループットを以下に示す. スループットは, 処理バイト数/処理時間で求められる. 復号処理の際,1byte 処理するのに 1clock かかることも考えると以下のようにスループットが導出される.

(1) 辞書作成からコンテキスト保持までのスループット t_1

表 3　評価用トラフィックデータ
Table 3　Traffic Data for Evaluation.

全体トラフィック中,HTTP ストリーム合計バイトが占める割合	78 %
全 HTTP ストリーム中,GZIP 圧縮されたストリーム合計バイトが占める割合	7.05 %
全体トラフィック中,GZIP 圧縮されたストリーム合計バイトが占める割合	5.50 %
平均パケット長	1,221.54byte
平均 GZIP 圧縮率	32.33 %
平均 GZIP 展開後パケット長	3,778.43byte
1GZIP ストリームを構成する平均パケット数	5packet

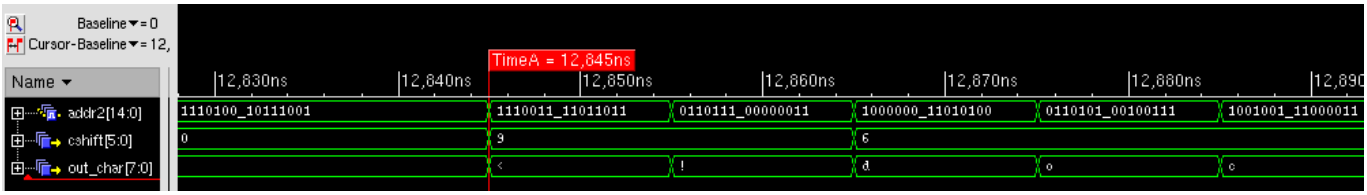

図 8　復号に関する波形の様子
Fig. 8　Waveform of the Decoding.

表 6　合成結果（個々）
Table 6　Synthesis Result(individual).

adpt.v	回路規模	$6.83\times10^3\mu m^2$
	動作遅延	1.98ns
	ゲート数	8,563
chosei.v	回路規模	$2.14\times10^2\mu m^2$
	動作遅延	0.47ns
	ゲート数	268
ctable.v	回路規模	$3.68\times10^3\mu m^2$
	動作遅延	1.93ns
	ゲート数	4,614
dhead.v	回路規模	$1.08\times10^2\mu m^2$
	動作遅延	0.21ns
	ゲート数	135
ghead.v	回路規模	$1.30\times10^2\mu m^2$
	動作遅延	0.21ns
	ゲート数	163
mtable.v	回路規模	$1.19\times10^5\mu m^2$
	動作遅延	1.59ns
	ゲート数	148,573
nlz.v	回路規模	$1.18\times10^3\mu m^2$
	動作遅延	1.92ns
	ゲート数	1,483
shifter.v	回路規模	$3.58\times10^3\mu m^2$
	動作遅延	1.98ns
	ゲート数	4,488
tsume.v	回路規模	$1.10\times10^3\mu m^2$
	動作遅延	1.78ns
	ゲート数	1,373

1 パケット (1,221.54B) の GZIP 復号処理を行うときに辞書作成も合わせて行う場合, 処理時間は, 辞書作成にかかる 1,279clock と, 復号処理に 3,778.43B×1$clock/B$ と, 復号処理が終わった後コンテキストメモリに辞書を転送する際にかかる 68clock であり,

$$\begin{aligned} t_1 &= 1,221.54B/((1,279+3,778.43+68) \\ &\quad \times 2.99ns) \\ &\approx 0.59Gbps \end{aligned} \tag{6}$$

となる.

(2) コンテキスト読み込みからコンテキスト保持までのスループット t_2
すでにコンテキストメモリに辞書が存在し,1 パケットを復号する場合, 処理時間は, コンテキストメモリから辞書を転送する際にかかる 48clock と, 復号処理に 3,778.43B×1$clock/B$ と, 復号処理が終わった後コンテキストメモリに辞書を転送する際にかかる 68clock であり,

$$\begin{aligned} t_2 &= 1,221.54B/((48+3,778.43+68) \\ &\quad \times 2.99ns) \\ &\approx 0.78Gbps \end{aligned} \tag{7}$$

となる.

(3) 平均スループット t_3
1 ストリームを復号する場合,1 ストリームは平均 5 パケットで構成されるから, 先頭のパケットのみ辞書作成プロセスが存在し, 残りのパケットではコンテキストメモリ転送の処理が存在する. したがって

$$\begin{aligned} t_3 &= (1,221.54\times5)B/((3,778.43\times5) \\ &\quad +1,279+((68+48)\times4)+68) \\ &\quad \times 2.99ns \\ &\approx 0.74Gbps \end{aligned} \tag{8}$$

となる.

これらのスループットの値はいずれも要求仕様である 0.55Gbps を満たしている.

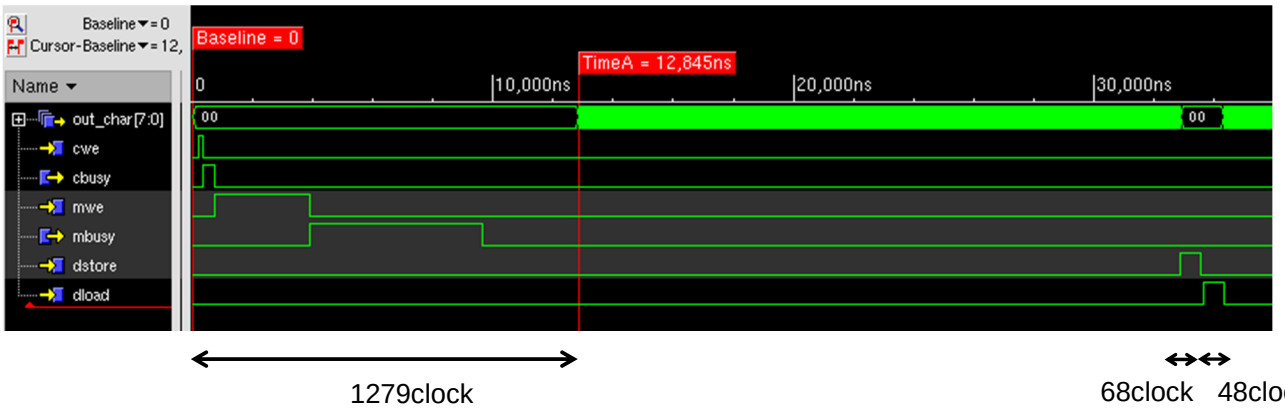

図 9 復号におけるレイテンシ
Fig. 9 Latency of the Decoding.

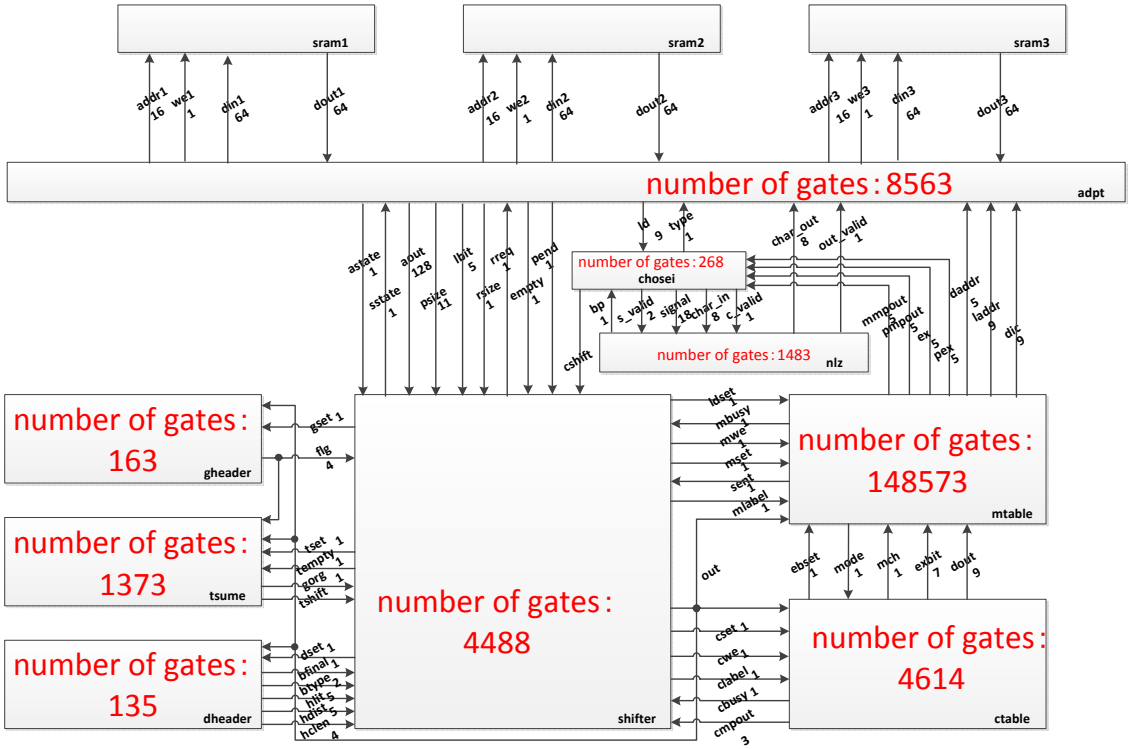

図 10 全体のアーキテクチャとゲート数
Fig. 10 the Whole Architecture and the Number of Gates.

4.5 コンテキストスイッチの有無による比較

コンテキストスイッチがある場合の GZIP 復号処理とない場合を比較を行うにあたって, 必要なメモリ量とスループットをそれぞれ導出する. ある任意のタイミングに存在する GZIP ストリーム数を x とする.

コンテキストスイッチ有の場合はパケット毎に処理を行うため, Ethernet の最大転送ユニットである 1,500B を保持できるインプットメモリを必要であると仮定した. その他にコンテキスト情報を保持するメモリがストリーム数分必要である.

一方, コンテキストスイッチ無の場合はストリーム全体をインプットメモリに保持する必要が有り, 平均的なストリームサイズである 6,107.70B をストリーム数分保持できるインプットメモリが必要であると仮定した. ただしインプットメモリはストリームサイズの最大値を保持できる容量である必要が有るため, 実際はこの値を超えるメモリが必要になる.

上記の仮定のもとメモリサイズとスループットの比較を**表 7** に示した.

コンテキストスイッチの有無に関わらず, スループットはさほど変わらない. 一方必要なメモリサイズはネットワークに存在する GZIP ストリーム数に比例して増加する

表 7 メモリサイズとスループットの比較
Table 7 Comparing in Memory Size and Throughput.

コンテキストスイッチ	メモリサイズ (B)	スループット
有	1,500 + 772×x (確定値)	0.74Gbps
無	6,107.70 ×x (平均値)	0.75Gbps

ため, コンテキストスイッチを用いた方がより省資源の機構を実現できると言える.

5. 結論

この論文ではサービス指向型ルータで動作することを目的としたハードウェア GZIP 復号処理機構を提案した. ソフトウェアではなくハードウェアで実装することにより, この GZIP 復号処理機構は膨大な圧縮データを高速に処理することが可能である. またコンテキストスイッチを用いることで, メモリ使用量を抑え, 省資源でなおかつ実際のネットワークスループットを満足する処理を可能にした.

シミュレーションは Verilog-HDL を用いて行い, 辞書作成に伴うレイテンシは 1,279 クロック, メモリへの辞書転送におけるレイテンシは 68clock, メモリからの辞書転送におけるレイテンシは 48clock であった. さらに Synopsys Design Compiler X-2005 ツールをもちいて合成を行ったところ, 回路規模は 0.14mm^2 となり, 最大動作周波数は 334.45MHz, ゲート数は 174,901 であった.

謝辞 この研究は, 文部科学省 科学技術戦略推進費 気候変動に対応した新たな社会の創出に向けた社会システムの改革プログラム 「グリーン社会 ICT ライフインフラ」, 独立行政法人情報通信研究機構 高度通信・放送研究開発委託研究「新世代ネットワークを支えるネットワーク仮想化基盤技術の研究開発」および, 文部科学省科学技術研究費補助金基盤研究 C「安心・安全な情報提供を可能とするインターネット基盤の構築に関する研究」(22500069) の一環としてなされた。

本研究は東京大学大規模集積システム設計教育研究センターを通し, シノプシス株式会社, 日本ケイデンス株式会社の協力で行われたものである.

参考文献

[1] Freepdk. http://www.eda.ncsu.edu/wiki/FreePDK.
[2] K. Inoue, D.Akashi, M. koibuchi, and H. Nishi. Semantic router using data stream to enrich services. In *3rd International Conference on Future Internet (CFI)*, pp. 20–23, June 2008.
[3] Koichi Inoue, Dai Akashi, Michihiro Koibuchi, Hideyuki Kawashima, and Hiroaki Nish. Semantic router using data stream to enrich services. In *3rd International Conference on Future Internet CFI 2008 Seoul*, pp. 20–23, June 2008.
[4] T. Grandpierre M. Akill, L. Perroton. *FPGA-based architecture for hardware compression/decompression of wide format images*. J Real-Time Image Proc (2006). Springer-Verlag, 2006.
[5] B. Shirazi and Y.-D. Song. A parallel exchange sort algorithm. In *Computers and Communications, 1989. Conference Proceedings., Eighth Annual International Phoenix Conference on*, pp. 366 –370, mar 1989.
[6] 福島荘之介. ファイル圧縮ツール gzip のアルゴリズム, 第 28 巻, pp. 30–37. bit, 第 3 版, March 1996.