高速かつ軽量な可変データ長対応のCRC回路構成手法

片下敏宏[†1] 坂巻佳壽美[†2] 名古屋貢[†3]
寺島康典[†4] 戸田賢二[†1]

本論文では超高速ネットワークに向けた，高速かつ軽量なCRC回路の構成手法を提案する．CRCの1つであるCRC-32はイーサネットのフレームの誤り検出に用いられており，その算出処理は任意データ長への対応やワイヤスピードであることが要求されている．スループット向上のためにCRC回路の処理データ幅を拡張すると，従来手法では処理データ幅 N-bit に対し回路規模が $O(N^2)$ となり，100 Gbps など非常に高いスループットを得るには莫大な回路資源が必要であった．そこで処理データ幅に対する回路規模を $O(N)$ へ低減させるCRC回路の構成手法を提案する．提案手法と従来手法におけるCRC回路を自動的に生成するソフトウェアを開発し，提案手法と従来手法のFPGAにおける回路規模とスループットを比較，検証した．その結果，提案手法により回路規模を $O(N)$ へ大幅に削減可能であり，スケーラビリティが高くなることが示された．また，提案手法による回路は現行のデバイスFPGA Virtex-II Pro 100（44,096 Slices）において，処理データ幅8,192-bitのとき回路規模24,627 Slices（55.8%）で1.18 Tbpsのスループットを達成することが分かった．

A Scalable Light-weight Circuit for CRC Calculation

Toshihiro Katashita,[†1] Kazumi Sakamaki,[†2] Mitsugu Nagoya,[†3]
Yasunori Terashima[†4] and Kenji Toda[†1]

We propose a method of constructing a scalable light-weight circuit for CRC calculation. A CRC-32 is used in a network frame, and calculating the CRC-32 at a wire-speed is demanded. A CRC-32 generator must deal with variable length data when the CRC-32 is calculated in parallel to accelerate the processing, because tail data that contain a various byte length. In previous studies, a resource requirement of the circuit was $O(N^2)$ in the case of processing N-bit in parallel. The requirement was quite large to obtain high throughput. For such reason, we developed a method of constructing a circuit that uses $O(N)$ resource, and a tool that automatically generates the circuit. We evaluated the throughput and the resource requirement of our circuit and compared with a traditional circuit. The result demonstrated that our method reduced the resource requirement to $O(N)$, and improved the throughput in comparison with the traditional method. Furthermore, our circuit has 1.18 Tbps throughput and uses 24,627 Slices (55.8%) of Xilinx Virtex-II Pro xc2vp-100 (44,096 Slices) in the case of 8,192-bit processing.

1. はじめに

CRC（Cyclic Redundancy Check）は誤り検出方式の1つであり，バースト誤りを検出できる点や簡易なハードウェアで実現できることから，シリアル伝送路における誤り検査などに広く用いられている．イーサネットにおけるフレームの誤り検査にもCRCが利用されており，ネットワークフレームのFCS（Frame Check Sequence）では32-bit長のCRC-32が採用されている．ネットワークフレームの誤り検出におけるCRCの利用では，処理をワイヤスピードで行うことが要求されている．近年では10 Gigabit Ethernetの登場などネットワークのスピードが大幅に向上しているほか，スループット100 Gbpsの次世代インターネットが検討されるなどの背景から，CRCの演算を高速に処理するハードウェアの開発が重要となっている．

単純なCRCハードウェア構成としてあげられるLFSR（Linear Feedback Shift Register）は，1クロックサイクルごとに処理するデータ幅が1-bitであり非

†1 産業技術総合研究所情報技術研究部門
Information Technology Research Institute, National Institute of Advanced Industrial Science and Technology
†2 地方独立行政法人東京都立産業技術研究センター
Tokyo Metropolitan Industrial Technology Research Institute
†3 デュアキシズ株式会社
DUAXES Corporation
†4 株式会社ビッツ
BITS Co., Ltd.

常に低速であった[1]．イーサネットフレームはバイト単位のデータ長であることから 1 サイクルごとに 8-bit ずつのデータを処理する手法により CRC 回路の処理性能が改善されてきた．

さらなる処理性能向上の手法の 1 つとして，回路の動作周波数を 1.25 GHz まで高めることにより 10 Gbps のスループットを ASIC 上で達成した研究[2] がなされているが，100 Gbps などのより高いスループットには対応が困難である．そのため 1 クロックサイクルで処理するデータ幅を拡張することにより低い動作周波数で処理性能を向上させる手法[3] が研究されており，FPGA へ実装して 10 Gbps のスループットを得る設計[4] などもなされている．

処理データ幅を拡張する手法においては，ルータなどのネットワーク機器応用では転送されるイーサネットフレームの長さを規定することが困難であり，CRC 回路はフレームがとりうるすべてのデータ長に対応する必要がある．しかし単純に一定のデータ幅ごとに処理する CRC 回路では，イーサネットフレームのデータ長が回路の処理データ幅の倍数とならない場合に間違った値を算出してしまう．そのため任意のデータ長に対応する CRC 回路の構成手法が研究・開発されてきたが，従来の研究における回路構成手法[3),4] では処理データ幅 N に対し回路規模のオーダが $O(N^2)$ となっていた．100 Gbps など 10 Gbps を超える次世代のイーサネットへ対応するには従来手法では回路規模が莫大となる問題があった．

そこで，本論文では任意のデータ長に対応する CRC 回路をオーダ $O(N)$ の規模で構成する手法を提案する．また，提案手法と従来手法における CRC 回路を自動的に生成するソフトウェアを開発し，これを用いて生成された両回路のスループットと回路規模の比較・検証実験を行った．さらに，処理データ幅が 64-bit の CRC 回路を FPGA 上に実装し，スループットが 10 Gbps（156.25 MHz）となることを実機で確認した．

2. CRC のハードウェアによる演算

2.1 CRC のアルゴリズム

誤り検出の対象となる L-bit からなるデータの 2 進数表現を $[a_0 a_1 \ldots a_{L-2} a_{L-1}]$（MSB が a_0，LSB が a_{L-1}）とするとき，データの多項式表現は

$$A = a_0 x^{L-1} + a_1 x^{L-2} + \cdots + a_{L-2} x + a_{L-1} \tag{1}$$

となる．次数 K の生成多項式を G とするとき，CRC 値 R は式 (2) により算出される[5]．

$$R = (Ax^K) \bmod G \tag{2}$$

CRC はバースト誤りを検出できる特徴を持ち，シリアル伝送路の符号などに広く用いられている．CRC-32 は IEEE802.3 のデータリンク層のフレームにおける FCS として用いられており，その値はフレーム中のアドレス部，レングス部，データ部より算出される．フレームデータ送信時にはアドレス・レングス・データ部を生成多項式で割った剰余 R が算出され FCS として付加される．受信時には同様に R が算出され，FCS と比較されてデータの誤りが検出される．FCS における CRC 値の算出では，フレーム先頭の 32-bit のデータはビット反転される．また，CRC 値は算出後にビット反転してフレームに付加される[6]．CRC-32 の生成多項式は

$$G = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \tag{3}$$

であり，R は 32-bit 長である．

2.2 処理データ幅固定の CRC 回路

CRC の算出対象となるデータを 1-bit ごとに処理する単純な CRC 回路は**図 1** のような LFSR 構成となる．図 1 は CRC-32 の場合の回路を表している．LFSR において L-bit のデータ A の MSB から i-bit（$1 \le i \le L$）の

$$A(i) = a_0 x^{i-1} + a_1 x^{i-2} + \cdots + a_{i-2} x + a_{i-1} \tag{4}$$

で表されるデータを処理した際の途中結果となる剰余 R_i は以下のように求められる．

$$\begin{aligned} R_i &= (A(i)x^K) \bmod G \\ &= (((a_0 x^{i-2} + \cdots + a_{i-2})x + a_{i-1})x^K) \\ &\quad \bmod G \\ &= ((A(i-1)x^K)x + a_{i-1}x^K) \bmod G \end{aligned}$$

ここで，$(Ax^b + B) \bmod G = (Px^b + B) \bmod G$（$b-1$ は B の次数，$P = A \bmod G$）より[7]，

$$= (R_{i-1}x + a_{i-1}x^k) \bmod G. \tag{5}$$

$R_0 = 0$ であり，CRC 値 $R = R_L$ である．LFSR は R_{i-1} とデータ a_{i-1} より剰余 R_i を算出し，これを L サイクル処理することにより CRC 値 R を算出する回路である．

この LFSR の数サイクル分の処理を 1 サイクルで行うような組合せ回路は，Verilog-HDL の for 文によ

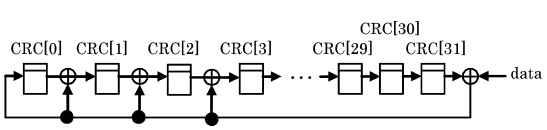

図 1 LFSR の構成
Fig. 1 Composition of LFSR.

```
function [31:0] next_crc;
 input [31:0] crc;
 input [7:0]   data;
 integer i;
 begin
    next_crc = crc;
    for(i=0;i<8;i=i+1)
     crc = {crc[30:0],1'b0} ^
     ({32{crc[31]^data[7-i]}} & 32'h04C11DB7);
 end
endfunction
```

LFSR 部

図 2 8-bit 処理の CRC 回路の Verilog-HDL ソースコード
Fig. 2 Verilog-HDL code for 8-bit processing CRC-circuit.

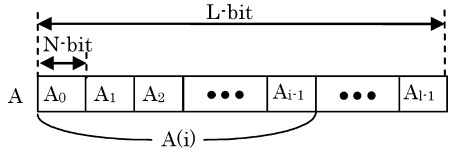

図 3 データ A の分割（$L = lN$ の場合）
Fig. 3 Division of the data A in case of $L = lN$.

り容易に構成することができる[1),8)]．**図 2** は LFSR の 8 サイクル分の処理を行う組合せ回路の Verilog-HDL ソースコードの例であり，これは処理データ幅が 8-bit の CRC 回路を示している．このような手法によって CRC 回路の 1 クロックサイクルあたりの処理データ幅を拡張することができる．

図 2 に示すように，LFSR の N サイクル分の処理を 1 サイクルで行う CRC 回路は，処理データ幅が N-bit の固定長となる．つまり，入力データが N-bit でない場合は誤った CRC 値が算出される．以後，処理データ幅が N-bit の固定である CRC 回路を CRC モジュール (N) と呼ぶ．

ここで，CRC 値の算出対象データの長さを L-bit とし，L が処理データ幅の倍数である場合（$L = lN$, $0 < l$）を考える．データ A を a_0 から順に N-bit ずつ区切り，l 個の部分データ $A_0, A_1, \ldots, A_{l-1}$ に分割する（**図 3**）．データ A_j（$0 \le j \le l-1$）を

$$A_j = a_{jN}x^{N-1} + a_{jN+1}x^{N-2} + \cdots + a_{jN+(N-1)} \tag{6}$$

と表したとき，データ A は A_j を用いて次式で表される．

$$\begin{aligned} A &= (a_0x^{N-1} + \cdots + a_{(N-1)}x^0)x^{(l-1)N} + \cdots \\ &\quad + (a_{(l-1)N}x^{N-1} + \cdots + a_{(l-1)N+(N-1)}x^0) \\ &= A_0x^{(l-1)N} + A_1x^{(l-2)N} + \cdots + A_{l-1} \end{aligned} \tag{7}$$

CRC モジュール (N) には A_0 から A_{l-1} までの部分データが順に入力され l サイクルで CRC 値 R が算出される．A_0 から i 個（$1 \le i \le l$）の部分データ

$$A(i) = A_0x^{N(i-1)} + \cdots + A_{i-2}x^N + A_{i-1} \tag{8}$$

まで処理した時点の剰余 R_i は次式で表される．

$$\begin{aligned} R_i &= (((A_0x^{N(i-2)} + \cdots + A_{i-2})x^N + A_{i-1})x^K) \\ &\quad \bmod G \\ &= (A(i-1)x^Kx^N + A_{i-1}x^K) \bmod G \\ &= (R_{i-1}x^N + A_{i-1}x^K) \bmod G \end{aligned} \tag{9}$$

$R_0 = 0$ であり，CRC 値 $R = R_l$ である．CRC モジュール (N) は R_{i-1} とデータ A_{i-1} より剰余 R_i を算出する回路であり，これを l サイクル処理させることにより CRC 値 R が算出される．図 2 中の next_crc が R_{i-1}，data が A_{i-1}，crc が R_i に対応する．

CRC モジュール (N) の回路規模はその処理データ幅に比例する[9)]．

2.3 任意バイト長のデータへの対応

CRC 値算出対象のデータが特定の長さであれば，そのデータ長に応じた CRC モジュールのみで CRC 値を算出することが可能である．たとえば，ATM（Asynchronous Transfer Mode）におけるプロトコル AAL5（ATM Adaptation Layer 5）ではフレーム長をパディングにより特定の倍数に調整されているため，処理データ幅が固定長の CRC モジュールのみで処理を行うことができる．ところが Ethernet ではフレーム長が 64〜1,518-byte の任意のバイト長をとるため，転送されるデータ長を前もって想定できない．さらに，Ethernet の機器はすでに広く普及しており，AAL5 のようにパディングによりあらかじめデータ長を調整することをすべての機器に規定することは困難である．したがって，Ethernet のルータ機器などにおける CRC 回路の応用では，CRC 回路が転送されるデータの長さが回路の処理データ幅の倍数とならない場合にも対応できる構成とする必要がある．一般にインターネットフレームのフレーム長は 64〜1,518-byte の任意のバイト長をとり，CRC 値算出の対象となるデータ長は 60〜1,514-byte の任意のバイト長をとる．

ここで CRC 値の算出対象データの長さを L-bit とし，L が CRC 回路の処理データ幅 N-bit の倍数でない場合（$L = lN + M$，$0 \le M < N$）を考える．A は l 個の N-bit 長の部分データ $A_0, A_1, \ldots, A_{l-1}$ と，M-bit 長の部分データ B に分割される（**図 4**）．このとき，lN-bit 長の部分データを

$$A' = A_0x^{(l-1)N} + \cdots + A_{l-2}x + A_{l-1} \tag{10}$$

とおくとデータ A は次式で表される．

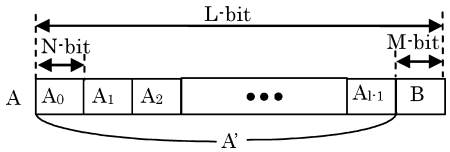

図 4 データ A の分割（$L = lN + M$ の場合）
Fig. 4 Division of the data A in case of $L = lN + M$.

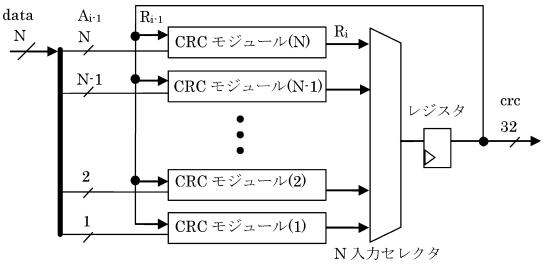

図 5 従来手法による CRC 回路の構成
Fig. 5 Composition of a traditional circuit.

$$\begin{aligned} A &= (a_0 x^{N-1} + \cdots + a_{(N-1)} x^0) x^{(l-1)N+M} + \cdots \\ &\quad + (a_{(l-1)N} x^{N-1} + \cdots + a_{(l-1)N+(N-1)} x^0) x^M \\ &\quad + (a_{lN} x^{M-1} + \cdots + a_{lN+(M-1)} x^0) \\ &= A' x^M + B \end{aligned} \tag{11}$$

ここで，

$$B = a_{lN} x^{M-1} + \cdots + a_{lN+M-1}.$$

このとき，CRC 値 R は式 (12) で表すことができる．

$$\begin{aligned} R &= (A' x^M + B) x^K \bmod G \\ &= (A' x^K x^M + B x^K) \bmod G \end{aligned}$$

ここで $R' = A' x^M \bmod G$ とおくと

$$= (R' x^M + B x^K) \bmod G \tag{12}$$

式 (9) より lN-bit 長のデータ A' を処理した時点の剰余は $R' = R_l$ であり，CRC モジュール (N) により l サイクルで算出される．一方 R の算出は B の長さ M が $0 \le M < N$ であるため，CRC モジュール (N) では正しい CRC 値が得られない（ただし $M = 0$ の場合は $R = R'$ となる）．

この M がとりうる数（データ長）に応じて処理データ幅がそれぞれ h-bit（$1 \le h < N$）の CRC モジュール (h) を $N-1$ 個用意し，M に従って CRC 値を算出する回路を切り替えることにより任意長のデータに対応する手法[3),4)] が提案されている．**図 5** に従来手法により構成された，1 サイクルの処理データ幅が N-bit であり，任意長のデータに対応する CRC 回路を示す．

従来手法により任意データ長に対応した CRC 回路は，並列に接続された CRC モジュール (h)（$1 \le h \le N$）より構成されている．それぞれの CRC モジュール (h) の出力は N 入力セレクタへ接続されており，入力データ長に応じた適切な値が選択される．

この回路では，まず CRC モジュール (N) が使用され CRC 算出対象のデータ A より N-bit ずつ区切られたデータ A_{i-1} から剰余 R' が得られる．そして CRC 値 R はデータ B の次数に応じた CRC モジュール (M) が使用され算出される（$M = 0$ の場合は $R = R'$ となる）．

従来手法による任意データ長に対応した CRC 回路の規模は，CRC モジュール (1) の回路規模を 1 とした場合，N 個の CRC モジュール (h) より構成されることから，式 (13) で算出される．

$$\sum_{h=1}^{N} h = \frac{N(N+1)}{2} = \frac{1}{2}N^2 + \frac{1}{2}N \tag{13}$$

回路規模は $O(N^2)$ となり，処理データ幅の拡張によりスループットを向上させると回路規模が非常に大きくなってしまうことが分かる．そのため，従来手法を用いて 10 Gbps を超える高いスループットを得ようとすると，莫大な回路資源が必要となってしまう問題があった．

3. 高速かつ軽量な CRC 回路構成手法

3.1 軽量な任意バイト長のデータ対応の回路

任意データ長に対応した CRC 回路が処理するデータ幅 N を $N = 2^n$ とし，CRC 値の算出対象のデータ $A = A' x^M + B$ の B を長さ $2^{n-1}, 2^{n-2}, \ldots, 2^0$ の部分データ $B_0, B_1, \ldots, B_{n-1}$ に分割することを考える．

B の 2 進数表現を $[b_0 b_1 \ldots b_{M-1}]$，B のデータ長 M の 2 進数表現を $[m_0 m_1 \ldots m_{n-1}]$ とし，

$$M = m_0 2^{n-1} + m_1 2^{n-2} + \cdots + m_{n-1} 2^0 \tag{14}$$

の初項から第 i（$0 \le i \le n-1$）までの部分和を

$$M_i = \sum_{j=0}^{i} m_j 2^{n-1-j} \tag{15}$$

$M = M_{n-1}$，$M - 2^0 = M_{n-2}$ とすると，B を次式で表すことができる（**図 6**）．

$$\begin{aligned} B &= b_0 x^{M-1} + b_1 x^{M-2} + \cdots + b_{M-1} \\ &= m_0 (b_0 x^{2^{(n-1)}-1} + \cdots + b_{2^{(n-1)}-1}) x^{M-M_0} \\ &\quad + m_1 (b_{M_0} x^{2^{(n-2)}-1} + \cdots \\ &\qquad + b_{M_0+2^{(n-2)}-1}) x^{M-M_1} + \cdots \\ &\quad + m_{n-2} (b_{M_{n-2}} x + b_{M_{n-2}+1}) x^{M-M_{n-2}} \\ &\quad + m_{n-1} (b_{M_{n-1}}) x^{M-M_{n-1}} \\ &= m_0 B_0 x^{M-M_0} + \cdots + m_{n-1} B_{n-1} \end{aligned} \tag{16}$$

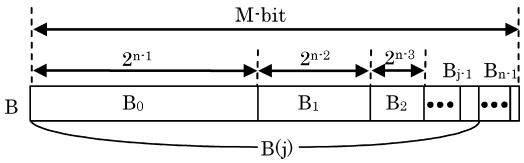

図 6 データ B の分割
Fig. 6 Division of the data B.

なお,

$$B_i = b_{M_i} x^{2^{(n-1-i)}-1} + \cdots + b_{M_i+2^{(n-1-i)}-1}.$$

たとえば, B の長さが 19-bit の場合は $19 = [10011]$ であるから, B は 2^4, 2^1, 2^0-bit の部分データ B_0, B_3, B_4 に分割される.

このときデータ B を B_0 から j 番目 ($1 \le j \le n$) の B_{j-1} までのデータ

$$B(j) = m_0 B_0 x^{M_{j-1}-M_0} + \cdots + m_{j-1} B_{j-1} \tag{17}$$

を処理した際の剰余 Q_j は次式で表される.

$$\begin{aligned} Q_j &= (R' x^M + B(i)) x^K \bmod G \\ &= (R' x^M + m_0 B_0 x^{M_{j-1}-M_0} + \cdots \\ &\quad + m_{j-2} B_{j-2} x^{M_{j-1}-M_{j-2}} \\ &\quad + m_{j-1} B_{j-1}) x^K \bmod G \\ &= (R' x^M + (m_0 B_0 x^{M_{j-2}-M_0} + \cdots \\ &\quad + m_{j-2} B_{j-2}) x^{M_{j-1}-M_{j-2}} \\ &\quad + m_{j-1} B_{j-1}) x^K \bmod G \end{aligned}$$

$M_{j-1} - M_{j-2} = m_{j-1} \cdot 2^{n-j}$ より,

$$\begin{aligned} &= (R' x^M + (m_0 B_0 x^{M_{j-2}-M_0} + \cdots \\ &\quad + m_{j-2} B_{j-2}) x^{m_{j-1} \cdot 2^{n-j}} \\ &\quad + m_{j-1} B_{j-1}) x^K \bmod G \\ &= (R' x^M + B(j-1) x^{m_{j-1} \cdot 2^{n-j}} \\ &\quad + m_{j-1} B_{j-1}) x^K \bmod G \end{aligned}$$

B_{j-1} の次数は $2^{n-j} - 1$ であるので,

$$= (Q_{j-1} x^{m_{j-1} \cdot 2^{n-j}} + m_{j-1} B_{j-1}) x^K \bmod G \tag{18}$$

$Q_0 = R'$ であり, CRC 値 $R = Q_n$ である. m_{j-1} は 0 または 1 であり, 0 のときは $Q_j = Q_{j-1}$ となる. これは $m_{j-1} = 0$ の場合は $2^{n-1-(j-1)}$ 長で分割するデータがないことを示しており, このとき剰余の算出処理は行われない. たとえばデータ B の長さが 19-bit で $n = 5$ の場合は, $19 = [10011]$ より B は 2^4, 2^1, 2^0-bit 長の部分データに分割されるが $m_1 = 0$, $m_2 = 0$ が示すように 2^3, 2^2-bit 長の部分データはない. その際には CRC モジュール (2^3) と CRC モジュール (2^2) では剰余の算出処理が行われない.

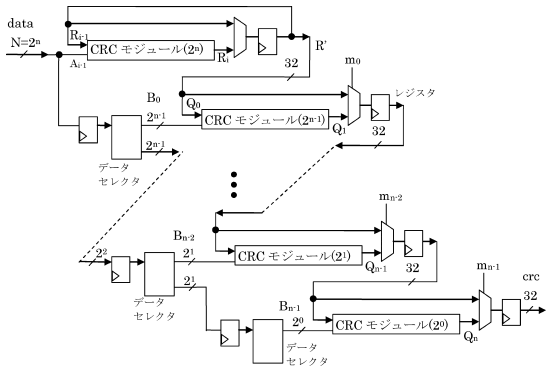

図 7 提案手法による回路の構成
Fig. 7 Composition of a proposed circuit.

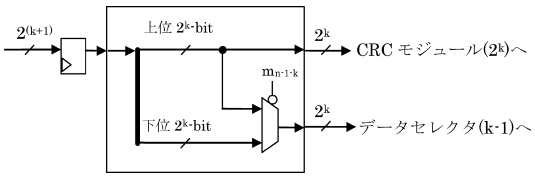

図 8 データセレクタの構成
Fig. 8 Composition of a data-selector circuit.

本研究では, 式 (18) が示すように次数 ($M < 2^n$) のデータ B を MSB から $2^{n-1}, \ldots, 2^0$ ずつ区切り, それぞれのデータ長に対応する CRC モジュール (2^k) ($n-1 \ge k \ge 0$) を用いて剰余 Q_j の算出を n サイクル処理して CRC 値 R を得る手法を提案する.

提案手法をもとに構成された, 1 サイクルの処理データ幅が N-bit ($N = 2^n$) で任意データ長に対応する CRC 回路を**図 7** に示す. 本回路は, $n+1$ 個の CRC モジュール (2^k) ($k = n, n-1, \ldots, 0$) が直列に構成されている. 各 CRC モジュール (2^k) 間には 2 入力セレクタとレジスタが配置されており, パイプライン構成となっている. また, 各 CRC モジュール (2^k) のデータ入力を適切に選択するデータセレクタが配置されている. データセレクタの構成を**図 8** に示す.

提案手法による回路では, 従来手法と同様に CRC モジュール (2^n) が使用され剰余 R' が算出される. そしてデータ B はデータセレクタにより m_{n-1-k} の値に従って区切られ, 各 CRC モジュール (2^k) へパイプライン式に入力される. また, 各 CRC モジュール (2^k) からの剰余出力は m_{n-1-k} の値に応じて 2 入力セレクタにより選択され, レジスタに格納される. たとえば, $M = 19 = [10011]$ の場合, 19-bit 長のデータ B は 2^4, 2^1, 2^0-bit のデータに分割され, それぞれのデータ長に対応した CRC モジュールに入力される. そして CRC モジュールからの出力が 2 入力セレ

クタにより選択され剰余の算出処理が行われる．このようにして n サイクル後に CRC 値 $R = Q_n$ を得る．なお CRC モジュール (2^n) の出力のセレクタは，$R = R'$ の場合（$M = 0$）の処理に使用される．

提案手法では CRC 値の算出がパイプライン式に処理されるため，従来手法に比べてレイテンシが n サイクル増加する．このレイテンシ増加は $O(\log N)$ となる．

また CRC 回路のデータセレクタ部を除いた回路規模は，CRC-32 モジュール (1) の回路規模を 1 とした場合，$n+1$ 個の CRC モジュール (h)（$h = 2^n, \ldots, 2^0$）より構成されることから式 (19) で算出され，$O(N)$ となる．

$$\sum_{k=0}^{n} 2^k = \frac{2^{n+1} - 1}{2 - 1} = 2N - 1 \tag{19}$$

データセレクタ部は $2^n, \ldots, 2^1$-bit のレジスタ $n-1$ 個と 2 入力セレクタ $n-1$ 個より構成されていることから，レジスタの規模は式 (19) と同様の計算により $O(N)$，2 入力セレクタの規模は $O(\log N)$ となる．

つまり，提案手法により回路の規模は処理データ幅 N に対して $O(N)$ となり，従来手法の $O(N^2)$ と比べ回路規模を大幅に削減できる．

3.2 CRC 回路生成ソフトウェア

任意データ長に対応した CRC 回路を自動的に生成するソフトウェアを Java 言語を用いて開発した．出力は Verilog-HDL の形式である．本ソフトウェアは CRC モジュール生成部と任意データ長に対応した CRC 回路生成部より構成されており，提案手法による回路と従来手法による回路を生成することができる．

CRC モジュール生成部では LFSR を仮想的に h サイクル動作させた処理データ幅 h-bit の CRC モジュール (h) を生成する．LFSR の仮想的な動作を図 2 のように for 文で記述すると，論理合成ツールの良し悪しにより回路規模や動作周波数が大きく左右される傾向にある．これは，for 文により記述した式が逐次的に評価されるときに XOR が鎖状に接続されたバランスの悪い XOR ツリーの回路が生成されるほか，XOR ツリーに含まれる冗長な XOR が削除されないためである．この問題を解決する手法として，for 文をあらかじめ評価して式に含まれる XOR を列挙し冗長な記述を削減する方法が提案されている[10]．本ソフトウェアでは文献 10) と同じ手法により冗長な記述の削減と XOR ツリーのバランスが行われた回路を生成する．

任意データ長に対応した CRC 回路生成部では処理データ幅が N-bit（$N = 2^n$）の回路を生成する．また，呼び出す関数によって提案手法による回路と従来手法による回路を選択できる．

任意データ長への対応が必要な応用としてはイーサネットで利用されている CRC-32 があげられる．このため本研究においては本ソフトウェアが対応する CRC は CRC-32 とした．

4. 検　　証

4.1 回路規模およびスループット

ソフトウェアにより自動生成した任意データ長に対応する CRC 回路のソースコードを論理合成し，回路規模とスループットにおいて提案手法と従来手法の比較・検証を行った．CRC 回路の処理データ幅は 16，32，64，128，256，512，1,024，2,048，4,096，8,192-bit とした．また回路規模やスループットの比較基準とする参考回路として，CRC モジュール (8) と制御回路で構成された処理データ幅が 8-bit の CRC 回路を用いた．実装対象デバイスとしては FPGA Xilinx 社 Virtex-II Pro 100（130 nm），Virtex-4 200（90 nm），Virtex-5 220（65 nm）を選択した．論理合成ツールは Xilinx ISE 8.1 sp3（Virtex-5 のみ 8.2 sp3）の xst を使用した．

まず予備実験として自動生成ソフトウェアで生成された CRC モジュール (2^k)（$k = 3, \ldots, 13$）が文献 9) に示されるように処理データ幅に比例した回路規模となるか検証した．対象デバイスを Virtex-II Pro 100 として論理合成した場合の結果を**表 1** と**図 9** に示す．表 1 中の width は処理データ幅を表している．図 9 中の縦軸は回路規模の Slice 数，横軸は処理データ幅を示しており，回路規模のオーダ推定の参考に適当な係数を持った一次式のグラフ $y = 4x$ を追加している．図 9 より，ソフトウェアで生成した CRC モジュールの回路規模も処理データ幅の $O(N)$ となることが分かる．

次に，提案手法と従来手法による CRC 回路の規模

表 1 CRC モジュールの回路規模
Table 1 Resource requirement of the CRC-module.

width (bit)	Slices	LUTs	Slice raito
8	32	55	1.00
16	55	96	1.72
32	107	186	3.34
64	172	299	5.38
128	296	515	9.25
256	509	886	15.91
512	858	1,492	26.81
1,024	1,507	2,621	47.09
2,048	2,682	4,665	83.81
4,096	4,722	8,212	147.56
8,192	8,415	14,634	262.97

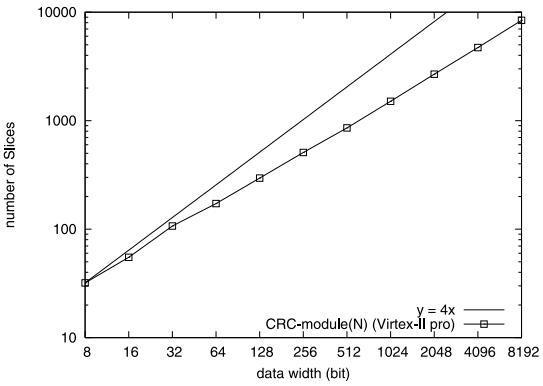

図 9 CRC モジュールの回路規模
Fig. 9 Resource requirement of the CRC-module.

表 2 CRC 回路の規模とスループット（Virtex-II Pro）
Table 2 Result of experimentation on Virtex-II Pro.

	width (bit)	Slices	Frequency (MHz)	Throughput (Gbps)
参考回路	8	91	313.48	2.508
提案手法	16	183	254.71	4.075
	32	314	240.79	7.705
	64	524	215.19	13.772
	128	842	206.44	26.424
	256	1,379	184.84	47.320
	512	2,337	189.11	96.823
	1,024	4,057	183.79	188.201
	2,048	7,100	167.98	344.029
	4,096	13,049	146.61	600.498
	8,192	24,627	144.16	1,180.918
従来手法	16	165	245.40	3.926
	32	375	197.71	6.327
	64	984	187.55	12.003
	128	2,503	161.45	20.665
	256	9,063	143.58	36.755
	512	28,394	124.02	63.498
	1,024	99,355	114.19	116.926

とスループット詳細の結果を Virtex-II Pro，Virtex-4，Virtex-5 を対象とした場合についてそれぞれ**表 2**，**表 3**，**表 4** に示す．表中の width は処理データ幅を表している．Virtex-5 を対象とした場合では，LUT（Look Up Table）と Flip-Flop の組の数に 0.25 を掛け小数点以下を繰り上げした数を Slice 数としている☆．従来手法による回路では，論理合成ツールが利用できるメモリの許容量を超えたため論理合成ができない処理データ幅があり，その際の結果は得られなかった．

☆ xst の出力結果には Slice 数は示されないが，LUT と Flip-Flop の組の数が Slice Logic Distribution に示されている．Virtex-5 では 1 つの Slice に LUT と Flip-Flop の組が 4 つ搭載されている[13] ため，Slice Logic Distribution に示されている数に 0.25 を掛け小数点以下を繰り上げした数が Slice 数に相当する．

表 3 CRC 回路の規模とスループット (Virtex-4)
Table 3 Result of experimentation on Virtex-4.

	width (bit)	Slices	Frequency (MHz)	Throughput (Gbps)
参考回路	8	82	348.92	2.791
提案手法	16	170	290.36	4.646
	32	302	271.59	8.691
	64	517	258.73	16.559
	128	820	249.88	31.984
	256	1,364	221.24	56.637
	512	2,318	208.16	106.578
	1,024	3,974	201.37	206.202
	2,048	6,577	189.47	388.026
	4,096	13,164	175.25	717.840
	8,192	24,604	161.24	1,320.862
従来手法	16	179	275.63	4.410
	32	348	226.55	7.250
	64	933	202.14	12.937
	128	2,963	175.00	22.400
	256	9,103	156.06	39.950
	512	28,357	133.92	68.567

表 4 CRC 回路の規模とスループット（Virtex-5）
Table 4 Result of experimentation on Virtex-5.

	width (bit)	Slices	Frequency (MHz)	Throughput (Gbps)
参考回路	8	36	403.28	3.226
提案手法	16	86	345.22	5.524
	32	171	344.61	11.028
	64	296	325.66	20.842
	128	437	307.32	39.337
	256	770	276.11	70.685
	512	1,267	258.71	132.457
	1,024	2,167	250.41	256.423
	2,048	3,141	232.04	475.218
	4,096	6,730	216.80	888.021
	8,192	12,219	202.68	1,660.314
従来手法	16	71	341.11	5.458
	32	181	267.18	8.550
	64	460	231.14	14.793
	128	1,459	209.35	26.797

提案手法と従来手法による Virtex-II Pro 使用時の CRC 回路の規模を比較したグラフを**図 10** に示す．図 10 中の縦軸は回路規模の Slice 数，横軸は処理データ幅を示しており，提案手法は our circuit，従来手法は traditional circuit，括弧中に使用デバイスを示している．また，図 10 には回路規模のオーダ推定の参考に適当な係数を持った一次式のグラフ $y = 10x$ と $y = 0.1x^2 + 10x$ を追加している．図 10 により，従来手法による回路の規模は $O(N^2)$ となり，提案手法の回路規模は $O(N)$ となることが分かる．

使用デバイスすべてにおける回路規模を比較したグラフを**図 11** に示す．Virtex-5 における Slice 数は 2.0

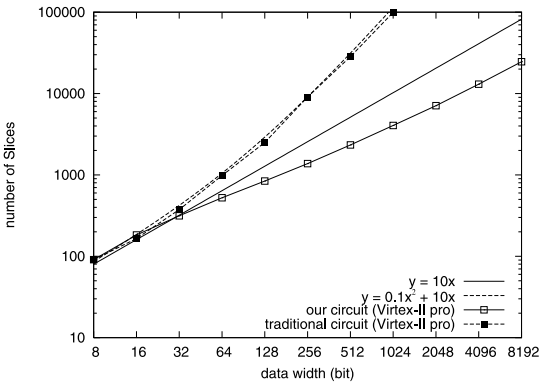

図 10 CRC 回路の規模の比較
Fig. 10 Comparison of the resource requirement.

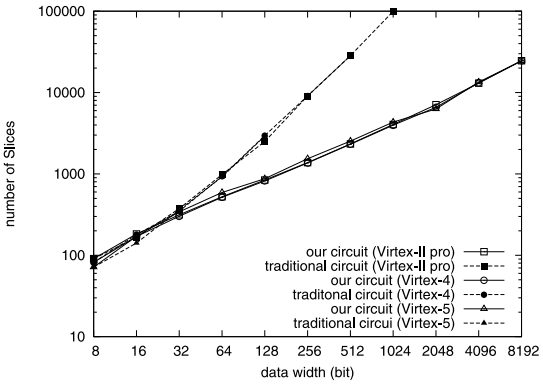

図 11 全実装対象デバイスにおける CRC 回路の規模
Fig. 11 Resource requirement of the circuits on all devices.

を掛けてプロットしている☆．いずれのデバイスにおいて比較しても $N = 32$-bit を超える処理データ幅で提案手法が従来手法より小さい回路規模となることが分かる．またデバイスによる回路規模の差は見られず，プロセスの変化により手法の優位性が変化しないことが分かる．一方，$N = 16$-bit のときに提案手法の方が大きい回路規模となる．$N = 16$-bit では CRC 回路の大半を占める CRC モジュールがどちらの手法でも同じ構成（CRC モジュール (8) と CRC モジュール (16)）となるが，提案手法はデータセレクタやパイプラインレジスタを持つため回路規模が大きくなると考えられる．なお Virtex-4 の場合は提案手法の方が小さい回路規模となっているが，これは CRC モジュール中の回路とその他の回路が同じ Slice に統合されたためであると考えられる☆☆．

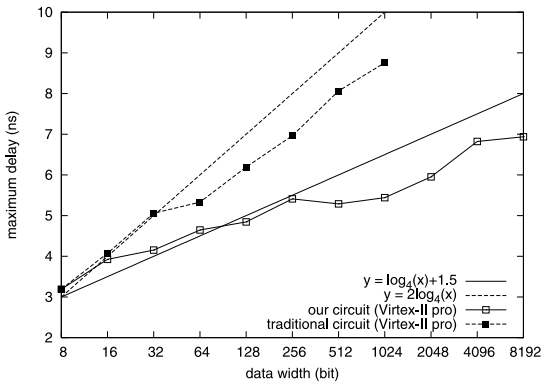

図 12 CRC 回路のスループットの最大遅延の比較
Fig. 12 Comparison of the maximum delay.

次に，提案手法と従来手法による Virtex-II Pro 使用時の CRC 回路の最大遅延を比較したグラフを**図 12** に示す．図 12 中の縦軸に最大遅延（表の Frequency の逆数），横軸は処理データ幅を示している．また，最大遅延のオーダ推定の参考に適当な係数と底を持ったグラフ $y = 2\log_4(x)$ と $y = \log_4(x) + 1.5$ を追加している．図 12 により，いずれの手法も回路の最大遅延は $O(\log N)$ となることが分かる．

最大遅延となるクリティカルパスは提案手法・従来手法のどちらも，剰余を保持するレジスタから CRC モジュールを通じ再び剰余を保持するレジスタへ戻る経路であった．この経路は主に CRC モジュールと CRC 値を切り替えるセレクタの 2 つを通る．1 つ目の CRC モジュールは 3.2 節で述べた XOR ゲートがツリー上に構成された回路であるが，その遅延は入力データ幅 N に対し $O(\log N)$ となる．2 つ目のセレクタは，提案手法においては入力数が固定の 2 入力であり一定の遅延となる．一方，従来手法においては N 入力セレクタとなるが，このセレクタは入力 $N = 2^n$ のとき n 段の 2 入力セレクタで構成されており，その遅延は入力データ幅 N に対し $O(\log N)$ となる．以上より，最大遅延はいずれの手法も入力データ幅 N に対し $O(\log N)$ であると考えられる．

提案手法と従来手法はセレクタの構成が異なるため，入力データ幅 N が大きくなるに従ってセレクタの遅延分だけ提案手法が優位であると考えられる．これは図 12 で提案手法に添えた $y = \log_4(x) + 1.5$

☆ Virtex-5 では Virtex-II Pro や Virtex-4 と比べ 1 つの Slice に含まれる LUT と Flip-Flop の数が 2 倍であり，Virtex-5 の Slice 数を 2 倍した数が他のデバイスの Slice 数に相当する[11)〜13)]．

☆☆ CRC モジュールをその他の回路と統合しない設定を行い検証したところ，提案手法は 213 Slice，従来手法は 156 Slice と Virtex-4 でも提案手法の方が大きい回路規模となることを確認した．なお，$N = 32$-bit 以上の処理データ幅においては提案手法の方が小さい回路規模となることも確認している．

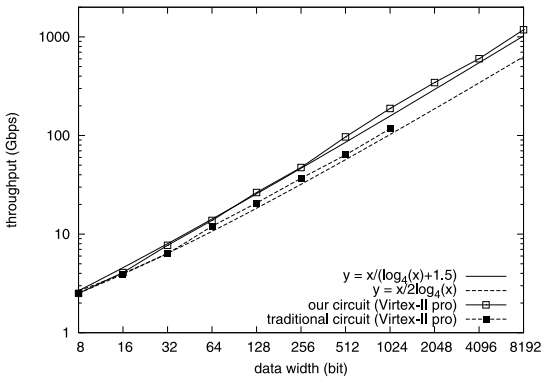

図 13 CRC 回路のスループットの比較
Fig. 13 Comparison of the throughput.

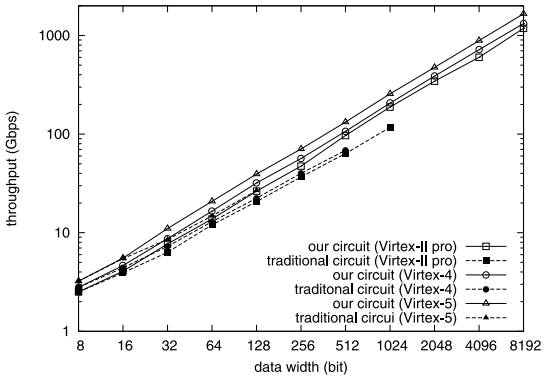

図 14 全実装対象デバイスにおける CRC 回路のスループット
Fig. 14 Throughput of the circuits on all devices.

のグラフに対し，従来手法に添えたグラフが $y = \log_4(x)+\log_4(x) = 2\log_4(x)$ であることからも，オーダは同じであるが提案手法の方が最大遅延で優位であると分かる．

このとき，回路のスループットは処理データ幅と最大遅延の逆数の積であり，いずれの手法も $O(N/\log N)$ となる．**図 13** に提案手法と従来手法による Virtex-II Pro 使用時の CRC 回路のスループットを比較したグラフを示す．図 13 より，いずれの手法もスループットは $O(N/\log N)$ となることが分かる．

次に，提案手法と従来手法のスループットを比較したグラフ**図 14** に示す．スループットはいずれのデバイスにおいても $O(N/\log N)$ となり，処理データ幅の拡張が有効であることが分かる．また，$N = 32$-bit を超える処理データ幅においては，提案手法が従来手法よりも総じてスループットが高くなっている．これは，従来手法では N 入力セレクタを使用しているが提案手法では 2 入力セレクタのみを使用しているために最大遅延が小さくなったと考えられる．

以上の実験結果により，従来手法では $O(N^2)$ であった回路規模を提案手法により $O(N)$ へ改善できること，テクノロジの変化により手法の優位性は変化しないこと，スループットを従来手法と同様に向上させることができることを確認できた．

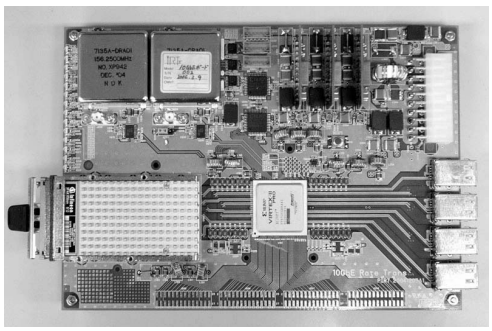

図 15 10 Gigabit Ethernet 試験ボード
Fig. 15 10 Gigabit Ethernet experimental board.

また提案手法による回路は Virtex-II Pro デバイスにおいて，処理データ幅 1,024-bit のとき回路規模 4,057 Slices（9.2%）で 188 Gbps のスループット，処理データ幅 8,192-bit のとき回路規模 24,627 Slices（55.8%）で 1.18 Tbps のスループットを達成することが分かり，次世代の超高速ネットワークへ対応可能であることを確認できた．

市販 IP として Calyptech 社の 40 Gbps のスループットを持つ CRC 回路[14] があげられるが，これは処理データ幅が 256-bit で Virtex-II Pro デバイス使用時に 5,859 Slices の回路規模となっている．一方，提案手法による同等のスループットを持つ 256-bit 処理の回路は 1,379 Slices と 1/4 以下の規模で実装可能であり，市販の IP コアを比較しても回路規模が小さく，高い実用性を持つことが確認できた．

4.2 実機による動作試験

提案手法による任意データ長に対応した CRC 回路を使用した，イーサネットフレームの FCS を検査する回路を FPGA デバイスへ実装し実際のネットワーク環境における動作確認を行った．

使用 FPGA は Xilinx Virtex-II Pro xc2vp-7 スピードグレード 6 であり，データ処理幅 64-bit，動作周波数 156.25 MHz，スループットは 10 Gbps となっている．実装は平成 16 年度経済産業省地域新生コンソーシアム研究開発事業「パターンマッチング回路の超高速化とフィルタリング装置への応用」において開発された 10 Gigabit Ethernet 試験ボード上へ行った．**図 15** に本ボードの写真を示す．また，試験環境を**図 16** に示す．10 Gigabit Ethernet 試験ボード上には FPGA と 10GBASE-SR 光モジュールが搭載されており，XAUI

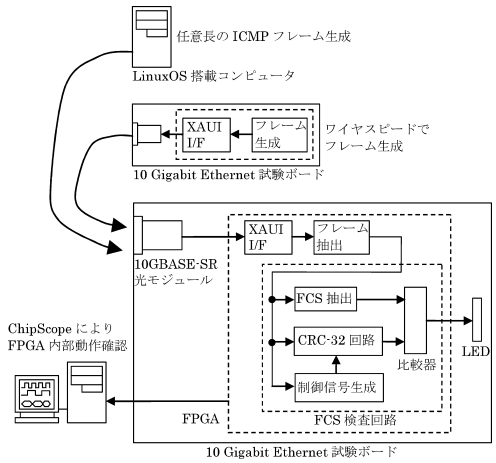

図 16 試験装置の回路構成
Fig. 16 Circuit composition of the experimental system.

(10 Gigabit Attachment Unit Interface) 経由でネットワーク環境へ接続することができる．FCS を検査する回路は図 16 に示すように，XAUI 用インタフェース，フレームデータ抽出回路，FCS 検査回路より構成されており，誤りが発見されたフレーム数の信号が LED に接続されている．

このようにして構成した装置に Intel PRO/10GbE SR Server Adapter を搭載した LinuxOS 搭載コンピュータや同様のハードウェア構成の 10 Gigabit Ethernet 試験ボードを接続し，CRC 回路の動作試験を行った．LinuxOS 搭載コンピュータを用いた試験では様々な長さの ICMP パケットを生成して装置へ入力することにより，任意長のフレーム入力における CRC 回路の動作を検証した．また，同型のボードを使用した試験ではワイヤスピードのフレーム入力における動作を検証した．検証においては装置の LED 出力や Xilinx 社 ChipScope ツールによる FPGA 内部の信号観察機能を用いた．

動作試験により，提案手法による CRC 回路は任意のデータ長に対応可能であり，また，連続して入力されたデータを正しくパイプライン処理できることを確認できた．

5. おわりに

本論文では，高速かつ軽量で任意データ長に対応する CRC 回路の構成手法を提案した．従来手法では CRC 回路の処理データ幅 N-bit に対し $O(N^2)$ であった回路規模を，提案手法により $O(N)$ へ改善することができる．

提案手法と従来手法における CRC 回路を自動的に生成するソフトウェアを開発し，提案手法と従来手法による CRC 回路の規模とスループットを比較検証した．その結果，提案手法により回路規模を $O(N)$ へ抑えることが可能であること，実装デバイスのプロセスによって手法の優位性が変化しないこと，処理データ幅の拡張にともなってスループットを向上可能であることを確認できた．特に，数 Gbps 以上のスループットに対応する処理データ幅が 32-bit 以上の CRC 回路においては，提案手法によって回路規模を大幅に削減でき，高いスループットを得ることができることを示した．また，現行のデバイスを利用しても 1 Tbps 超のスループットを持つ CRC 回路を実装可能であることが分かり，次世代の超高速ネットワークへ対応が可能であることを示した．

さらに，市販されている 40 Gbps のスループットを持つ IP と比較し提案手法による回路の優位性を示したほか，実際の FPGA ボード上へ提案手法による CRC 回路を実装して 10 Gbps 時の動作を検証することにより，提案手法の実用性を確認した．

謝辞 本研究の一部は，平成 16，17 年度地域新生コンソーシアム研究開発事業「パターンマッチング回路の超高速化とフィルタリング装置への応用」によるものである．

参考文献

1) Xilinx Inc.: xapp209, IEEE 802.3 Cyclic Redundancy Check.
2) Henriksson, T., Eriksson, H., Nordqvist, U., Larsson-Edefors, P. and Liu, D.: VLSI IMPLEMENTATION OF CRC-32 FOR 10 GIGABIT ETHERNET, *ICECS 2001*, pp.1215–1218 (2001).
3) Henriksson, T. and Liu, D.: Implementation of Fast CRC Calculation, *ASP-DAC 2003*, pp.563–564 (2003).
4) Xilinx Inc.: xapp562, Configurable LocalLink CRC Reference Design.
5) Ji, H.M. and Killian, E.: Fast Parallel CRC Algorithm and Implementation on a Configurable Processor, *ICC 2002*, pp.1813–1817 (2002).
6) 泉谷建司：*Ethernet*，ソフト・リサーチ・センター (1997).
7) Kounavis, M.E. and Berry, F.L.: A Systematic Approach to Building High Performance Software-Based CRC Generators, *ISCC'05*, pp.855–862 (2005).
8) NoBug Inc.: CRC Tool Computing CRC in Parallel for Ethernet (2004).
9) Sprachmann, M.: Automatic generation of

parallel CRC circuits, *Design & Test of Computers, IEEE*, Vol.18, No.3, pp.108–114 (2001).
10) Nair, R., Ryan, G. and Farzaneh, F.: A Symbol Based Algorithm for Hardware Implementation of Cyclic Redundancy Check (CRC), *VIUF 1997*, pp.82–87 (1997).
11) Xilinx Inc.: DS083, Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet.
12) Xilinx Inc.: UG070, Virtex-4 User Guide.
13) Xilinx Inc.: UG190, Virtex-5 User Guide.
14) Calyptech: Xilinx Alliance CORE: CRC-32 For 40 Gbps OC-768 Systems (CORE-CRC-256).

(平成 18 年 10 月 27 日受付)
(平成 19 年 4 月 6 日採録)

片下　敏宏

2006 年筑波大学大学院システム情報工学研究科卒業．博士（工学）．現在，産業技術総合研究所情報技術研究部門特別研究員．主としてFPGA，回路設計，ネットワークセキュリティに関する研究に従事．電子情報通信学会会員．

坂巻佳壽美

1974 年 3 月日本大学理工学部電気工学科卒業，1974 年 4 月から東京都立工業技術センター（現：地方独立行政法人東京都立産業技術研究センター）にて，中小企業への組み込みシステム技術に関する技術指導に従事し現在に至る．日本信頼性学会，電子情報通信学会各会員．

名古屋　貢

2007 年新潟大学大学院現代社会文化研究科社会文化論専攻卒業．文学修士．現在，デュアキシズ株式会社代表取締役社長．

寺島　康典

1991 年株式会社ビッツ入社，営業本部商品企画部を経て現在商品事業部商品企画部次長．主に通信機器関連の商品化に従事．

戸田　賢二（正会員）

1982 年慶應義塾大学大学院工学研究科修士課程修了．同年電子技術総合研究所入所．以来，並列コンピュータのアーキテクチャの研究に従事し，記号処理用データ駆動計算機や実時間処理用並列計算機の開発を行った．近年は，組み込み応用をターゲットとし，開発環境の整備とともに，実時間処理用ハードウェアやネットワークの実用化研究を推進中．