一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
VLD2018-73,CPSY2018-83,RECONF2018-47(2019-01)

拡張ユークリッド互除法における Leading Zero を利用した計算回数削減手法の提案

荻野　政樹†　田中　勇樹†　魏　書剛†

† 群馬大学 大学院 理工学府 理工学専攻　〒376-8515 群馬県桐生市天神町 1 丁目 5-1
E-mail: †{t171b019,ytanaka,wei}@gunma-u.ac.jp

あらまし　剰余乗算逆数は，離散対数問題による解析のされにくさから公開鍵暗号方式の秘密鍵の生成に利用されており，従来から高速な計算手法が望まれている．剰余乗算逆数を高速に計算するアルゴリズムとして，拡張ユークリッド互除法を使う方法がある．本研究では，ユークリッド互除法の除算と乗算を高速に行うため，除算と乗算を減算回路と加算回路で実現するパイプライン構造を採用する．互除法により，除算と乗算の演算数が小さくなることに着目し，Leading Zero 回路を導入することにより，乗算と乗算における減算と加算の回数を大幅に削減する．数値計算の実験を行い，加減算の回数が約 35 %削減できることを示し，提案の演算回路の高速性を明らかにしている．
キーワード　剰余乗算逆数，拡張ユークリッド互除法，リーディングゼロ

Proposal of reduction method of calculations by using Leading Zero in the Extended Euclidean Algorithm

Masaki OGINO†, Yuuki TANAKA†, and Shugang WEI†

† Graduate School of Science and Technology, Gunma University.
Tenjincho 1-5-1, Kiryu-shi, 376-8515 Japan.
E-mail: †{t171b019,ytanaka,wei}@gunma-u.ac.jp

Abstract　The modular multiplication inverse is used to generate the secret key of the public key cryptosystem from the difficulty of analysis due to the discrete logarithm problem, and high speed computation method has been desired conventionally. The extended Euclidean algorithm is an algorithm to calculate the modular multiplication inverse at high speed. In this study, the division and multiplication of the extended Euclidean algorithm are computed by using a subtraction circuit and an addition circuit with a pipeline structure. To reduce the times of the subtractions and additions in the division and multiplication of the extended Euclidean algorithm, we introduce Leading Zero circuit into the controlling unit. By experiments with some numeral computations, it is shown that a high speed modular multiplication inverse can be achieved.
Key words　Modular multiplicative inverse, Extended euclidean algorithm, Leading zero

1. はじめに

剰余演算は，信号処理の高速化処理や通信システムにおける暗号技術などでよく利用されている [1]．特に、離散対数問題による解析のされにくさから，公開鍵暗号方式の鍵の生成に役立てられている [2]．剰余加算，剰余減算，剰余乗算アルゴリズムおよび演算回路が多数提案されてきた [3] [4]．剰余除算は剰余乗算逆数に剰余乗算を行うことにより実現される．ある整数 A の剰余乗算逆数を求める演算は，M を法とした場合，

$$A \times X \equiv 1 \pmod{M} \tag{1}$$

を満たす X を探すことであり，Signed-Digit 数 [5] 表現を用いた加算回路により演算回路を提案している [6]．しかし，この方法では剰余加算が高速に行えるが，剰余加算の回数は非常に多くなってしまう．

本研究では，計算回数が比較的少なく，剰余乗算逆数を計算できる拡張ユークリッド互除法 [7] に着目し，これを高速・効率的に計算する回路を実装することを目的とする．拡張ユークリッド互除法は除算で計算される剰余が 1 になるまで，除算・乗算・減算の繰り返しにより，剰余乗算逆数を計算する．このうち，除算・乗算が演算におけるクリティカルパスとなっており，これらを高速化することは演算全体の高速化につながる．

ユークリッド互除法の除算と乗算を高速に行うため，除算と乗算を減算回路と加算回路で実現するパイプライン構造を採用する．互除を進めていく際に，除算と乗算の演算数が小さくなることに着目し，Leading Zero 回路を導入することにより，乗算と乗算における減算と加算の回数を大幅に削減する手法を提案し，高速性を明らかにする．数値計算の実験を行い，加算と減算の回数を約 35 %削減できることを確認した．

2. 拡張ユークリッド互除法を用いた剰余乗算逆数演算

最大公約数の性質を用いて自然数 a, b の最大公約数を求める方法をユークリッド互除法という．本研究ではこれをさらに拡張した，拡張ユークリッド互除法を用いて剰余乗算逆数を計算する．

2.1 剰余乗算逆数

整数 A と M が $1 < A < M$ の関係にあり，互いに素なものとする．このとき，M を法とする剰余数表現において式 (1) を満足する整数 X を，A の法 M における剰余乗算逆数という．また，式 (1) は式 (2) のようにも書く．

$$|A \times X|_M = 1. \tag{2}$$

2.2 拡張ユークリッド互除法

a，b が 0 でない整数であるとき，式 (3)（ベズーの等式）を満たす整数 x，y(ベズー係数) が存在する．ここで，$\gcd(a, b)$ は整数 a，b の最大公約数である．

$$a \times x + b \times y = \gcd(a, b). \tag{3}$$

a，b が互いに素であるとき，a と b の最大公約数は 1 であるため，式 (3) の右辺は 1 となる．このとき，$ax = -by + 1$ であるから，x は a の法 b における剰余乗算逆数とみることができる．A，M を互いに素な数とし，拡張ユークリッド互除法を用いて A の法 M における剰余乗算逆数を求めるには，以下の式 (4) を初期値を $Q_{-1} = M$，$Q_0 = A$，$X_{-1} = 0$，$X_0 = 1$ として計算する．

$$\begin{cases} Q_i &= \left\lfloor \frac{R_{i-2}}{R_{i-1}} \right\rfloor, \\ R_i &= |R_{i-2}|_{i-2}, \\ X_i &= X_{i-2} - Q_i \times X_{i-1}. \end{cases} \tag{4}$$

R_i が 1 になるまで式 (4) の計算を繰り返す．$R_i = 1$ となったときの X_i が A の法 M における剰余乗算逆数 X $(0 < X < M)$ である．

2.3 計 算 例

$A = 11$，$M = 15$ として，このときの剰余乗算逆数 X を求める．前述の定義に従って初期値を以下のように定める．

$$\begin{cases} Q_{-1} &= 15, \\ Q_0 &= 11, \\ X_{-1} &= 0, \\ X_0 &= 1. \end{cases}$$

これらを用いて，式 (4) に従って計算すると

$$Q_1 = 1,\ R_1 = 4,\ X_1 = -1,$$
$$Q_2 = 2,\ R_2 = 3,\ X_2 = 3,$$
$$Q_3 = 1,\ R_3 = 1,\ X_3 = -4.$$

したがって，求める剰余乗算逆数 X は

$$X = |-4|_{15} = |-4 + 15|_{15} = 11.$$

2.4 演算回路の構成

拡張ユークリッド互除法を用いた剰余乗算逆数回路のブロック図を図 1 に示す．ここで，入力は A（2 進数 n 桁），M（2 進数 n 桁）であり，出力は X（2 進数 n 桁）である．

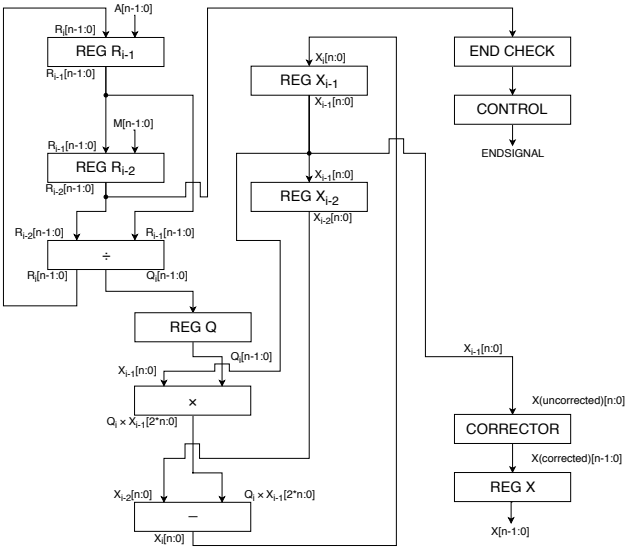

図 1　拡張ユークリッド互除法による剰余乗算逆数演算回路

図 1 において，÷ が除算ブロック，× が乗算ブロック，– が減算ブロック，END CHECK が終了判定ブロック，CONTROL が制御ブロック，CORRECTOR が補正ブロック，REG が各種レジスタを表す．CONTROL は回路全体を制御するブロックであり，順序回路で実装された除算・乗算に合わせて，各レジスタの制御を行う．END CHECK は計算終了を検出するブロックであり，その結果を CONTROL に伝える．CORRECTOR は X を補正するブロックである．拡張ユークリッド互除法では X の値が負数として算出される場合があり，必要に応じて法 M を加算して X の範囲を $0 < X < M$ に収める必要がある．これを行うブロックが CORRECTOR である．

入力として受け取った A と M をレジスタに格納し，式 (4) に従って除算・乗算・減算を繰り返し行うことで剰余乗算逆数を計算する．式 (4) では，除算結果である商を用いて乗算を行っているが，除算が完了するまで乗算を開始できないため，クリティカルパスの増大を招く．そこで，最適化のために除算と乗算の間にレジスタを挟み，パイプライン化を行う．これにより，乗算は除算よりも 1 工程分遅れて計算されるが，1 工程あたりの計算時間が大幅に削減できるため，総合的な計算時間は少なくなる．

n 桁 2 進数のユークリッド互除法における除算・乗算の最大計算回数はラメの定理から $(\log_2((1+\sqrt{5})/2))^{-1} \times n \simeq 1.44n$ 回以下となる [9] が，本構成では前述に述べたようにパイプライン化を行っているため，$1.44n+1$ 回以下となる．

図 1 における除算・乗算は加算器，マルチプレクサ，ビットシフトによって実現される順序回路である．除算を 1 回行うには n 回の加算が，乗算を 1 回行うには $n-1$ 回の加算が必要となる．

ユークリッド互除法の性質として，計算のステップが進めば進むほど，除算で扱う被除数と除数の桁数が小さくなっていくことが挙げられる．商の桁数も同様に小さくなることから，乗算で扱う乗数の桁数も小さくなっていく．この性質を利用することで，除算・乗算に必要な加算回数を削減することができる．

そこで，本研究では Leading Zero [8] とビットシフトを用いることで，除算・乗算の計算に要する加算の回数を最適化する．

3. Leading Zero

Leading Zero は入力の最上位桁から初めに見つかった非零桁との間にある 0 の個数を数える演算である．8 桁 2 進数の場合の Leading Zero の真理値表を，表 1 に示す．ここで，*は don't care を表し，0 と 1 の両方を満たす．

具体例として 8 桁の 2 進数 $(00011011)_2 = (27)_{10}$ を入力とした Leading Zero を考える．最上位桁から各桁を見ていくと，0 が 3 桁連続したあと，4 桁目で 1 が現れている．この場合では最上位桁から初めに見つかった非零桁との間にある 0 の個数は 3 個である．したがって，出力は $(3)_{10} = (0011)_2$ となる．

表 1　Leading Zero(8bit) の真理値表

入力	出力
00000000	1000
00000001	0111
0000001*	0110
000001**	0101
00001***	0100
0001****	0011
001*****	0010
01******	0001
1*******	0000

Leading Zero は入力の最上位桁から最下位桁まですべての桁を調べる必要があるため，桁数に比例した計算時間が必要となる．本研究では，高速化のため二分木構造を用いた Leading Zero 回路を設計し，除算・乗算の計算回数の最適化に利用する．

以下に仕様を示す．図 2 に二分木構造化した Leading Zero 回路のブロック図を示す．ここで，入力は IN（2 進数 n 桁），出力は OUT（2 進数 $\lceil \log_2 n \rceil + 1$ 桁）である．上記の例に従えば，8 桁の 2 進数入力 $IN = (00011011)_2$ に対する Leading Zero 出力は 3 であるが，ここでの出力 OUT は 4 桁の 2 進数で表され，$OUT = (011)_2$ となる．

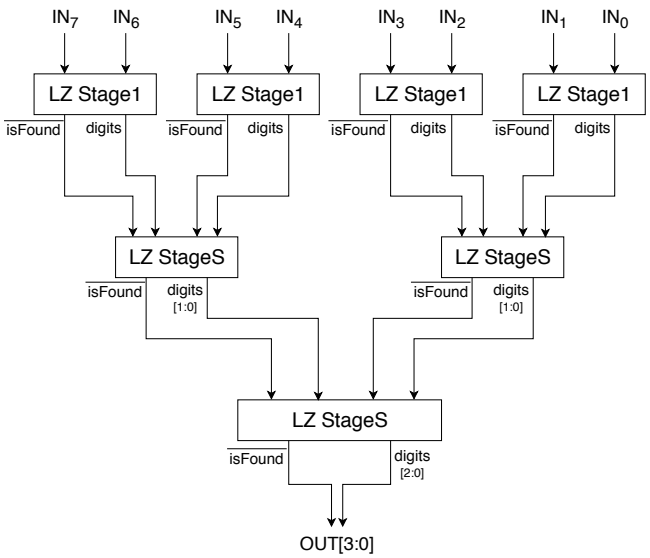

図 2　二分木 Leading Zero 回路 (8bit)

3.1 LZ Stage1

LZ Stage1 は二分木構造における 1 段目のブロックである．ここでは，2 桁分の入力を受け取り，入力値に 1 が含まれているかどうかの判定を行う．また，1 が含まれていた場合に，出力 $\overline{isFound}$ を 0 とし，入力値のどちらに含まれていたかを出力 $digits$ の 0/1 で下位のブロックに伝える．n 桁の Leading Zero では，並列に $n/2$ 個使用する．

真理値表を表 2 に示す．

表 2　LZ Step1 の真理値表

入力		出力	
x_{high}	x_{low}	$\overline{isFound}$	$digits$
0	0	1	0
0	1	0	1
1	0	0	0
1	1	0	0

3.2 LZ StageS

LZ StageS は二分木構造における 2 段目以降のブロックである．ここでは，上位のブロックから入力を受け取り，入力値に 1 が含まれているかどうかの判定を 2 つの入力 $\overline{isFound}_{high}$，$\overline{isFound}_{high}$ から行う．また，1 が含まれていた場合出力 $\overline{isFound}$ を 0 に，入力値のどちらに含まれていたかを出力 $digits$ として下位のブロックに伝える．なお，出力 $digits$ の桁数は 2 つの入力 $digits_{high}$，$digits_{low}$ の桁数よりも 1 つ大きくなる．

Leading Zero における段数を $S(\log_2 n > S > 1)$ とする．真理値表を表 3 に示す．

4. Leading Zero を利用した拡張ユークリッド互除法を用いた剰余乗算逆数演算

2. で述べた拡張ユークリッド互除法を用いた剰余乗算逆数演算に，3. で述べた Leading Zero を導入し，除算・乗算部分の

表 3　LZ StepS の真理値表

入力				出力	
$\overline{isFound_{high}}$	$\overline{isFound_{low}}$	$digits_{high}$	$digits_{low}$	$\overline{isFound}$	$digits$
0	0	D_{high}	D_{low}	0	0 D_{high}
0	1	D_{high}	D_{low}	0	0 D_{high}
1	0	D_{high}	D_{low}	0	1 D_{low}
1	1	D_{high}	D_{low}	1	0 D_{high}

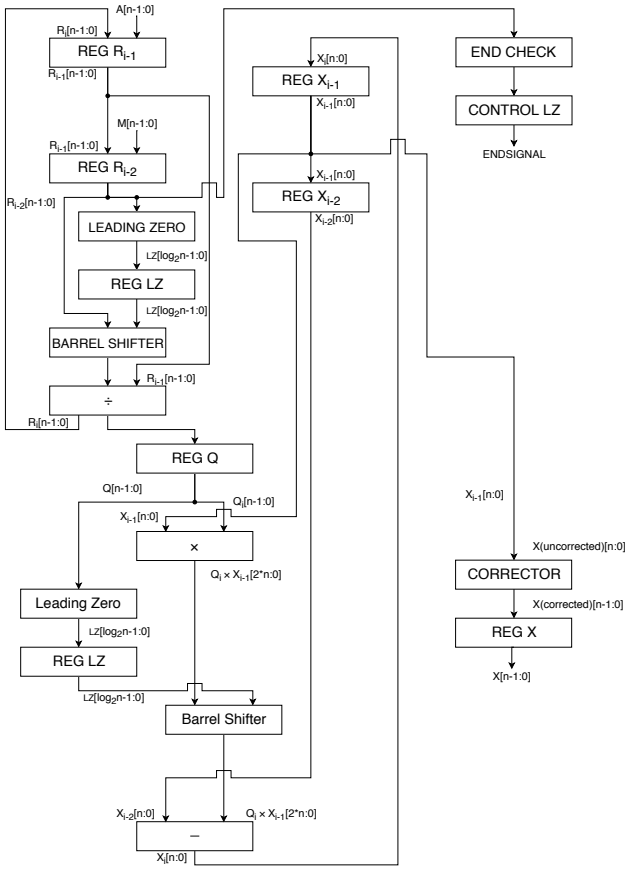

図 3　Leading Zero 回路を用いた拡張ユークリッド互除法による剰余乗算逆数演算回路

加算回数の最適化を行う．ブロック図を図 3 に示す．

CONTROL LZ が制御ブロック，LEADING ZERO が Leading Zero を計算するブロック，BARREL SHIFTER がバレルシフタである．LEADING ZERO ブロックで計算した LeadingZero を CONTROL LZ に伝えることで，除算・乗算の加算回数の制御に利用する．また，バレルシフタは任意の桁数だけシフトする回路であり，Leading Zero と組み合わせることで後述する除算・乗算の最適化に利用する．他のブロックについては図 1 と同様である．

4.1　除算の最適化

被除数 X（n 桁），除数 Y（n 桁）（ただし $X > Y$），商 Q（n 桁），剰余 R（n 桁）とした場合の除算では，演算 1 回あたり n 回の加算が必要となる．ここで $(x_{n-1} \cdots x_i) = (0 \cdots 0)$ があらかじめわかっていれば，$(q_{n-1} \cdots q_i) = (0 \cdots 0)$ が成り立ち，$n-1$ 桁目から i 桁目までの計算をする必要がない．これは i 桁同士の除算 $(x_{i-1} \cdots x_0)$ と考えることができるため，演算に必要な加算の回数を i 回に抑えることができる．

例として 8 桁同士の除算 $(00011001)_2 \div (00000111)_2$ を考える．通常の除算では 8 回の加算が必要となる．ここで，被除数 $(00011001)_2$ の Leading Zero 結果は 3 である．これは $(11001)_2 \div (00111)_2$ として考えることができ，5 回の加算で計算ができる．

以上を踏まえると，X の Leading Zero を算出，これを用いて X および Y の値を適切にシフトし，加算回数を制御することで無駄な加算を省き，除算の最適化が実現できる．

4.2　乗算の最適化

被乗数 X（n 桁），乗数 Y（n 桁），積 P（$2n$ 桁）とした場合の乗数では，演算 1 回あたりに $n-1$ 回の加算が必要となる．ここで $(y_{n-1} \cdots y_i) = (0 \cdots 0)$ があらかじめわかっていれば，$(p_{2n-1} \cdots p_{2n-i-1}) = (0 \cdots 0)$ が成り立ち，$n-1$ 桁目から i 桁目までの計算をする必要がなくなる．これは n 桁の被乗数と i 桁の乗数との乗算と考えることができるため，演算に必要な加算の回数を $i-1$ 回に抑えることができる．なお，使用する加算器の桁数は固定であるために，X の桁数は変動しない．

例として 8 桁同士の乗算 $(00011001)_2 \times (00000111)_2$ を考える．通常の乗算では 7 回の加算が必要となる．ここで，乗数 $(00000111)_2$ の Leading Zero 結果は 5 である．これは $(00011001)_2 \times (111)_2$ として考えることができ，2 回の加算で計算ができる．

以上を踏まえると，Y の Leading Zero を算出，これを用いて加算回数を制御することで無駄な加算を省き，乗算の最適化が実現できる．

5.　性 能 評 価

4. で提案した回路構成を VHDL を用いてハードウェア記述を行い，その VHDL コードより Design Compiler を用いて論理合成を行った．また，ターゲットライブラリとして 0.18μm CMOS ゲートアレイ設計技術ライブラリを用いて各回路の評価を行った．

拡張ユークリッド互除法の除算・乗算部分における加算回数を評価するために，$n = 32$ における互いに素な整数 A，M の組み合わせ（$2^{31} \leq A < M \leq 2^{32} - 1$）をランダムに 200 組用意した．それぞれにおいて Leading Zero を用いない回路（図 1），Leading Zero を用いた回路（図 3）の両方で剰余乗算逆数演算の計算ステップ数のシミュレーションを行った．200 組それぞれについて，計算に要した CLK 数を計測し，それらの平均を算出した．

表 4 に拡張ユークリッド互除法による剰余乗算逆数演算回路の評価を記載した．上から順に 1CLK あたりの最大遅延時間 [ns]，回路面積 [ns]，消費電力 [mW]，演算に必要な総 CLK 数の平均 [-]，総計算時間の平均 [ns]（1CLK あたりの最大遅延時間 [ns]× 演算に必要な総 CLK 数の平均 [-]）を示す．表 5 に Leading Zero 回路の評価を記載した．

表 4 において，1CLK あたりの遅延時間で見れば桁数 $n = 32$ でおよそ 11% ほど増加しているが，計算に要した CLK 数はおよそ 35% ほど削減できた．以上を踏まえて，演算に要した総計算時間では，およそ 27% の高速化が実現できた．

回路面積においては，Leading Zero，バレルシフタ，レジスタがそれぞれ 2 つずつ増えていることから，50% ほどの増加が見られた．

1CLK あたりの遅延時間に大きな差が生じなかったのは，クリティカルパスが同一のブロック（除算部分）になったからと考えられる．除算ブロックは両方において同一の回路を使用している．

6. おわりに

本稿では，拡張ユークリッド互除法を用いて剰余乗算逆数演算を高速・効率的に実装することを目的とし，Leading Zero を用いた加算回数の削減手法を提案した．

また，0.18μm CMOS ゲートアレイ設計技術を用いて Leading Zero を組み込んだ拡張ユークリッド互除法を用いた剰余乗算逆数演算回路を設計することで，Leading Zero を用いないものと比べて加算回数を削減し，演算の高速化が実現できることを明らかにした．

今後はさらに桁数の大きい剰余乗算逆数演算において，Leading Zero が拡張ユークリッド互除法における，除算・乗算の加算回数にどのような影響を与えるかを調査し，比較・評価を行う．

謝辞　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，日本シノプシス合同会社の協力で行われたものである．

文　献

[1] N.S. Szabo and R.I. Tanaka，“Residue Arithmetic and Its Applications to Computer Technology，” McGraw-Hill, New York，1967.

[2] 結城浩，“新版 暗号技術入門 秘密の国のアリス，” ソフトバンク クリエイティブ株式会社，2011 年 10 月．

[3] S. Wei, and K. Shimizu, “Residue Arithmetic with a Signed-Digit Number System，” 4th HPC-ASIA The Fourth International Conference/Exhibition on High Performance Computing in Asia-Pacific Region Volume 1，pp.349-354，May 2000.

[4] 陳土爽清，“冗長な数表現を用いた剰余算術演算回路に関する研究，” 群馬大学 大学院 工学研究科 情報工学専攻 修士論文，2002.

[5] A. Avizienis，“Signed-Digit Number Representations for Fast Parallel Arithmetic，” IRE Trans. Electronic Computers，Vol.EC-10，pp.389-400，Sept．1961.

[6] 賈鵬，魏書剛，“SD 数剰余加算を用いた剰余除算回路の構成，” 信学技報，vol．106,no．453，VLD2006-88，pp．19-24，2007 年 1 月．

[7] Alasdair McAndrew，“Introduction to Cryptography with Open-Source Software，” CRC Press，2011.

[8] Miller, Jane E, “The Chicago Guide to Writing about Numbers，” University of Chicago Press，2008.

[9] R.Honsberger，“Mathematical Gems II，” Dolciani Mathematical Expositions，No. 2，Mathematical Assn of Amer，June 1976.

表 4　拡張ユークリッド互除法による剰余乗算逆数演算回路の評価

	桁数 n	$n=4$	$n=8$	$n=16$	$n=32$	$n=64$
遅延時間 (1CLK)	通常	2.69	2.95	4.88	9.24	13.82
[ns]	Leading Zero	2.89	5.16	6.84	10.30	15.51
回路面積	通常	10368.09	19941.37	40256.94	78228.52	153286.88
[μm^2]	Leading Zero	13357.93	29041.01	58994.10	118326.60	240398.32
消費電力	通常	0.20	0.36	0.66	1.26	2.44
[mW]	Leading Zero	0.24	0.41	0.73	1.35	2.58
平均 CLK 数	通常	-	-	-	668.45	-
[-]	Leading Zero	-	-	-	434.98	-
平均総計算時間	通常	-	-	-	6176.48	-
[ns]	Leading Zero	-	-	-	4480.24	-

表 5　Leading Zero 回路の評価

	桁数 n	$n=4$	$n=8$	$n=16$	$n=32$	$n=64$
遅延時間 [ns]		0.64	0.74	1.22	1.55	2.08
回路面積 [μm^2]		109.81	255.11	590.95	1285.19	2683.37
消費電力 [mW]		0.03	0.08	0.16	0.33	0.70