Vol.2018-AL-169 No.5
2018/9/3

FPGAを用いたコラッツ予想の網羅的検証の高速化

佐伯 和人[1,a)]　伊藤 靖朗[1]　中野 浩嗣[1]

概要：コラッツ予想とは数論の未解決問題の 1 つである．任意の自然数 n が与えられたとき，n が偶数なら 2 で割り，n が奇数なら 3 を掛けて 1 を足す，得られた値に対しても同様の操作を繰り返すと有限回で 1 に到達するという予想である．本論文ではコラッツ予想の網羅的検証を高速に行う FPGA 実装を提案する．提案実装では回路のリソースを削減することでマルチコプロセッサシステムとしたときにより多くのコプロセッサを並べることを可能にした． 本研究は Xilinx Virtex UltraScale+ FPGA を対象デバイスとして設計を行い，4096 個のコプロセッサを並べたマルチコプロセッサシステムを実装した．結果として提案実装は 1 秒間に 6.16×10^{12} 個の 64bit 自然数の検証が可能であり，既存 GPU 実装と比較して約 2.56 倍の性能向上を達成した．

キーワード：コラッツ予想，FPGA，DSP ブロック，ブロック RAM

Acceleration for Exhaustive Verification of the Collatz Conjecture using the FPGA

Kazuto Saiki[1,a)]　Yasuaki ito[1]　Koji Nakano[1]

Abstract: The Collatz conjecture is a well-known unsolved problem in mathematics. Consider the following operation on an arbitrary positive number: if the number is even, divide it by two, and if the number is odd, triple it and add one. The conjecture asserts that, starting from any positive number, repeated iteration of the operations eventually produces the value 1. In this paper, we propose an FPGA implementation of the exhaustive verification for the Collatz conjecture. In the proposed implementation, a large number of coprocessors can be arranged on an FPGA by reducing the resource of circuits. We have implemented a multi-coprocessor system that has 4096 coprocessors on a Xilinx Virtex UltraScale+ FPGA. The experimental results show that our multi-coprocessor system can verify 6.16×10^{12} 64bit natural numbers per second. Also, our implementation on the FPGA attains a speed-up factor of 2.56 over the existing GPU implementation.

Keywords: Collatz conjecture, FPGA, DSP blocks, block RAMs

1. はじめに

FPGA（Field Programmable Gate Array）とは設計者が任意に論理回路を再構成できる半導体集積回路である．ハードウェアでありながらソフトウェアのように回路を書き換えることが可能であることに加え，高性能化，低消費電力化，低コスト化が進んでいることから様々な分野で利用されている．**図 1** に FPGA の内部構造を示す． FPGA 内部のアーキテクチャは，CLB(Configurable Logic Blocks)，IOB(Input Output Block)，ハードマクロに分類でき，それらを接続する内部配線から構成される．CLB の内部には LUT（Look Up Table）や FF（Flip Flop）が搭載されている．また，ハードマクロには，乗算や加算を高速に行う回路を含む DSP ブロックや，メモリの役割を果たすブロック RAM やウルトラ RAM がある．FPGA では，これらを効率よく使用してパイプライン化，並列化させた回路設計をすることで計算の高速化を図ることができる．

[1] 広島大学
Hiroshima University, Higashi-hiroshima, Hiroshima 739–8527, Japan
[a)] saiki@cs.hiroshima-u.ac.jp

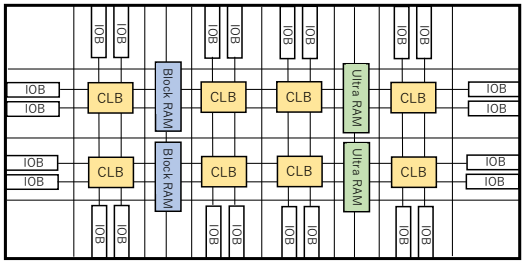

図 1 FPGA の内部構造
Fig. 1 An FPGA architecture

コラッツ予想とは，任意の自然数 n が与えられたときに次の操作を有限回繰り返すことで必ず 1 に到達するという予想である．

n が偶数の場合: n を 2 で割る (偶数操作)

n が奇数の場合: n に 3 を掛けて 1 を足す (奇数操作)

検証する初期自然数を 3 としたときは，以下のような操作が行われ，1 に到達する．

$$3 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$$

本論文ではコラッツ予想の網羅的検証を行う．網羅的な検証とは，1 から無限大までの全ての自然数に対して操作を繰り返して行っていくということであり，アルゴリズムは以下のように表すことができる．

for $m \leftarrow 1$ to ∞ **do**
  $n \leftarrow m$
  while $n > 1$ **do**
    if n is even **then**
      $n \leftarrow \frac{n}{2}$
    else
      $n \leftarrow 3n + 1$
    end if
  end while
end for

コラッツ予想の網羅的検証は様々なコミュニティで行われており [1], [2]，検証を高速に実行する GPU 実装 [8] や FPGA 実装 [3], [4], [5] が提案されている．

本論文の目的は FPGA を用いてコラッツ予想の網羅的検証を高速化することであり，既存実装 [4] のコプロセッサ回路を基に改良を行った．検証においての高速化手法として，n に対する複数の偶数・奇数操作を BC テーブルを用いた $B[n_L] \times n_H + C[n_L]$ の 1 回の操作で実行できることが知られている [5]．ここでの n_H は n の上位ビット，n_L は n の下位 bit である．また，S テーブルを用いて検証が必要な自然数を限定する手法も知られている [5]．既存実装 [4] は Virtex-6 FPGA を対象にしており，ブロック RAM と DSP ブロックを用いたパイプライン処理で $B[n_L] \times n_H + C[n_L]$ の計算の高速化を行っている．BC テーブルと S テーブルの格納にそれぞれブロック RAM を 1 つ用いており，DSP ブロックでは 17bit×17bit の符号なし乗算と内部の 17bit シフトを利用した桁上がりの計算を 3 ステージのパイプライン処理で行っている．なお，ブロック RAM からの読み出しに 1 ステージを要するため，実際には 4 ステージで計算を行っている．また，n_H の保存に 17bit レジスタを 6 つ用いているため 10bit の n_L と合わせて 112bit$(17 \times 6 + 10)$ まで計算可能となっており，n_H が格納されているレジスタの値によってパイプラインの条件分岐が発生し，4~6 クロックで 1 回の $B[n_L] \times n_H + C[n_L]$ の計算が完了する実装である．

Virtex UltraScale+を対象デバイスとした提案実装では，1 コプロセッサあたりのリソースを削減してマルチコプロセッサシステムとしたときにより多くのコプロセッサを並べることを可能にした．既存実装との変更点は主に 3 つである．1 つ目は S テーブルをホスト PC からの入力として与え，S テーブルにブロック RAM を用いないことである．この変更を行うことでブロック RAM の使用量を半減させることができる．なお，S テーブルを外部からの通信によって与えるため通信回数が増えるが，コプロセッサ回路での計算時間を増加させることで通信時間の隠蔽を行っている．計算時間と共に計算個数も同様に増加するため，計算時間の増加がスループットの増減には影響しない．2 つ目は 1 回の計算 bit 数の拡張である．既存実装では DSP ブロック内部の 17bit シフトを用いているため，DSP ブロックの入力ポート A は 17bit に固定されている．そこで提案実装では DSP ブロックの外部にシフトを設置し，入力ポート A の制限をなくしている．これにより既存実装での DSP ブロックで計算できる符号なし乗算は 17bit×17bit であるのに対して，提案実装では 26bit×17bit となり 9bit の拡張となっている．3 つ目は多倍長演算に使用するワード数を固定したことである．既存実装は n_H の値によって 4~6 のワードを使用する条件分岐ありのパイプライン実装であるのに対して，提案実装ではワード数を 4 に固定した条件分岐なしのパイプライン実装となっている．これにより回路を簡単化し，リソースを減らすことができることに加え，検証に必要なクロック数も減少する．また，n_H の保存に 26bit レジスタを 4 つ用いており n_L の 10bit と合わせて 114bit$(26 \times 4 + 10)$ まで計算できるため，既存実装より対応 bit が 2bit 増加している．提案実装のシングルコプロセッサではブロック RAM の数は半減しており，CLB も削減されている．また，4096 個のコプロセッサを並べたマルチコプロセッサシステムでは 1 秒間に 1.82×10^{12} 個の 64bit 自然数の検証が可能となった．さらに，S テーブルを外部入力としてブロック RAM の容量による制限がなくなったため，より大きな S テーブルを用いて検証を行うことが可能である．提案実装として 37bit の S テーブルを用いたコプロセッサ回路を作成した．これを 4096 個並べたマルチコプロセッサシステムでは 1 秒間に 6.16×10^{12} 個の自然数の検証が可能であり，既存 GPU 実装 [8] と比較し

て約 2.56 倍の性能を達成した．

2. 網羅的検証の高速化手法

コラッツ予想の検証を高速に行うために本論文では 3 つの手法を使用する [5].

1 つ目は，検証終了条件の変更である．コラッツ予想の検証は繰り返し操作の結果が 1 になる前に終了することができる．まず，1 から検証を始めて，自然数 $n-1$ までコラッツ予想が成り立っている状態を仮定する．そして，現在は自然数 n を検証しているとする．この自然数に偶数・奇数操作を繰り返していき，自然数 $n'(<n)$ が得られたとする．n 未満の自然数は既にコラッツ予想が成り立つことが示されているので，n' もコラッツ予想が成り立つ．これにより 1 になるまで操作を繰り返す必要がなく，n 未満の自然数が得られた時点で n に対しての検証を終了することができる．例として，13 を検証すると以下のようになる．

$$13 \to 40 \to 20 \to 10(<13)$$

10 は 13 より小さい自然数であり，既に検証が終了し，コラッツ予想が成り立つことが証明されている．よって，13 についても 3 回の操作で 10 に推移するので，コラッツ予想が成り立つことが示されたことになり，これ以上の操作を繰り返す必要がなく，検証を終了できる．

2 つ目は，複数の偶数・奇数操作をまとめるテーブルの作成である．コラッツ予想の操作では，複数回の偶数・奇数操作を 1 回の操作にまとめることができる．まず，検証する n を上位 bit n_H と下位 bit n_L に分ける．下位 bit n_L が 2bit の場合を考えると，$n = 4n_H + n_L$ と表すことができる．このとき，n_L のとる値は，$(00)_2, (01)_2, (10)_2, (11)_2$ の 4 通りとなり，以下のように複数回の操作を 1 回の操作に変更できる．

$n_L = (00)_2$： 最下位 bit が 0 なので偶数操作を行い，$2n_H + 0$ となる．また，最下位 bit が 0 なので偶数操作を行い，$n_H + 0$ となる．

$n_L = (01)_2$： 最下位 bit が 1 なので奇数操作を行い，$12n_H + 4$ となる．また，最下位 bit が 0 なので偶数操作を行い，$6n_H + 2$ となる. さらに，最下位 bit が 0 なので偶数操作を行い，$3n_H + 1$ となる．

$n_L = (10)_2$： 最下位 bit が 0 なので偶数操作を行い，$2n_H + 1$ となる．また，最下位 bit が 1 なので奇数操作を行い，$6n_H + 4$ となる. さらに，最下位 bit が 0 なので偶数操作を行い，$3n_H + 2$ となる．

$n_L = (11)_2$： 最下位 bit が 1 なので奇数操作を行い，$12n_H + 10$ となる．また，最下位 bit が 0 なので偶数操作を行い，$6n_H + 5$ となる. さらに，最下位 bit が 1 なので奇数操作を行い，$18n_H + 16$ となる．最後に，最下位 bit が 0 なので偶数操作を行い，$9n_H + 8$ となる．

$n_L = (11)_2$ の場合，2 回の偶数操作と 2 回の奇数操作を省略して $9n_H + 8$ を得ることができる．よって，偶数か奇数かが判定できない値，つまり n_H の係数が奇数になるまでの複数回の操作は 1 回で行うことができる．表 1 に，n_L が 2bit の場合の上位 bit n_H に掛ける値を B として，加算する値を C としてまとめた 2bit の BC テーブルを示す．

表 1 2bit の BC テーブル
Table 1 2bit table BC

n_L	B	C
00	1	0
01	3	1
10	3	2
11	9	8

表 2 4bit の S テーブル
Table 2 4bit table S

	S
0	0111
1	1011
2	1111

本論文では，複数の偶数・奇数操作を省略するために以下のように定義づけたテーブル操作を使用する．

テーブル操作： 自然数 n を最下位 bit の n_L と残り bit の n_H に分ける．それらの値より，$B[n_L] \times n_H + C[n_L]$ を計算し，新しい自然数を得る．

3 つ目は，検証する自然数の限定である．網羅的検証を高速化するために，検証する自然数の限定を行う．2.2 節でのテーブル作成時に 1 度でも元の自然数 n を下回ることがあった n_L については，コラッツ予想が成り立つことが証明されているので，そのような n_L は検証不要である．このことを，**表 1** の 2bit の BC テーブルを具体例に挙げて考える．

$n_L = (00)_2$： テーブル作成開始時の値は $n = 4n_H + 0$ であり，テーブル完成までの最小値は $n' = n_H + 0$ であるので，$n > n'$ が成りたち，元の自然数よりも小さくなる．

$n_L = (01)_2$： テーブル作成開始時の値は $n = 4n_H + 1$ であり，テーブル完成までの最小値は $n' = 3n_H + 1$ であるので，$n > n'$ が成りたち，元の自然数よりも小さくなる．

$n_L = (10)_2$： テーブル作成開始時の値は $n = 4n_H + 2$ であり，テーブル完成までの最小値は $n' = 2n_H + 1$ であるので，$n > n'$ が成りたち，元の自然数よりも小さくなる．

$n_L = (11)_2$： テーブル作成開始時の値は $n = 4n_H + 3$ であり，テーブル完成までの最小値は $n' = 6n_H + 5$ であるので，$n < n'$ が成りたち，元の自然数よりも大きくなる．

以上の結果より，$n_L = (00)_2, (01)_2, (10)_2$ の場合は，元の自然数を下回ることが保証されるので，コラッツ予想が成り立つ．よって，2bit のテーブルを用いる場合は，$n_L = (11)_2$ となる自然数 n のみの検証を行っていけばよい．この操作で得られた検証が必要な n_L を S テーブルに格納する．**表 2** に，n_L が 4bit の場合の S テーブルを示す．表 2 より，4bit の S テーブルのサイズは 3 なので，全 16

表 3　BC テーブルのサイズ
Table 3　The size of table BC

下位 bit	要素数	ワードサイズ	平均操作数
4	16	14	6.0
5	32	16	7.5
6	64	20	9.0
7	128	24	10.5
8	256	26	12.0
9	512	30	13.5
10	$1k$	32	15.0
11	$2k$	36	16.5
12	$4k$	40	18.0
13	$8k$	42	19.5
14	$16k$	46	21.0
15	$32k$	48	22.5
16	$64k$	52	24.0

通りの n_L のうちの 3 通りのみの検証が必要となる．これは，自然数全体の 0.1875 の比率の自然数に検証を限定できることを意味する．

3. 既存実装

S テーブルと BC テーブルを用いたコラッツ予想の検証は，下位 bit を $d$bit，作成した S テーブルのサイズを s_d としたとき，アルゴリズムは以下のようになる．

for $m_H \leftarrow 1$ to ∞ **do**
  for $i \leftarrow 0$ to $s_d - 1$ **do**
    $m_L \leftarrow S[i]$
    $n \leftarrow m \leftarrow 2^d m_H + m_L$
    while $n \geq m$ **do**
      $n \leftarrow B[n_L] \times n_H + C[n_L]$
    end while
  end for
end for

表 3 に下位 bit に応じた BC テーブルの要素数，ワードサイズ，1 度のテーブル操作が平均何回の偶数・奇数操作に相当するかを示す．また，**表 4** に下位 bit に応じた S テーブルの要素数，ワードサイズ，その要素数の自然数全体における比率を示す．既存実装では，10bit の BC テーブルと 15bit の S テーブルをそれぞれ 36kb ブロック RAM に格納して利用する．表 3 より，10bit の BC テーブルを使用した 1 回の値の更新は 15 回の偶数・奇数操作を行ったことと等しいことが分かる．また，表 4 より，15bit の S テーブルを使用することで自然数全体の 96%の検証を省略できることが分かる．

3.1 ブロック RAM と DSP を使用した $B[n_L] \times n_H + C[n_L]$ の計算

既存実装は Virtex-6 FPGA[6] を対象デバイスにして

表 4　S テーブルのサイズ
Table 4　The size of table S

下位 bit	要素数	ワードサイズ	比率
4	3	4	0.1875
5	4	5	0.1250
6	8	6	0.1250
7	13	7	0.1016
8	19	8	0.0742
9	38	9	0.0742
10	64	10	0.0625
11	128	11	0.0625
12	226	12	0.0552
13	367	13	0.0448
14	734	14	0.0448
15	1295	15	0.0395
16	2113	16	0.0322
⋮	⋮	⋮	⋮
37	967378591	37	0.0070

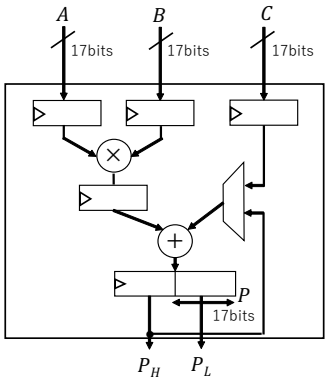

図 2　既存実装の DSP48E1 のブロック図
Fig. 2　A part of the block diagram of DSP48E1 of previous implementation

おり，搭載されている 36kb ブロック RAM は，$32k \times 1$，$16k \times 2$，$8k \times 4$，$4k \times 9$，$2k \times 18$，$1k \times 36$ のいずれかとして構成できる．表 3 より，10bit の BC テーブルを $1k \times 36$ の 36kb ブロック RAM で構成し，表 4 より，15bit の S テーブルを $2k \times 18$ の 36kb ブロック RAM で構成する．また，$B[n_L] \times n_H + C[n_L]$ の計算を行うために，DSP ブロック (DSP48E1)[10] を**図 2** のように設計する．

図 2 より，DSP48E1 は $P \leftarrow A \cdot B + C$ または，$P \leftarrow A \cdot B + P_H$ の計算が可能であり，A ポートは 25bit 以下，B ポートは 18bit 以下，C ポートと P ポートは 48bit 以下に対応している．また，DSP ブロックは 2 の補数乗算を行うため，符号なし乗算は 24bit×17bit まで対応している．出力ポート P は上位 31bit の P_H と下位 17bit の P_L に分けられる．図 2 からも分かる通り，A, B, C ポートをすべて 17bit に設定する．

次に，$n \leftarrow B[n_L] \times n_H + C[n_L]$ の計算方法を説明する．$B[n_L]$ と $C[n_L]$ の値は常に入力ポート B, C から得られる．

また，n_H は k 個の符号なし 17bit 整数 $n_0, n_1, n_2, ..., n_{k-1}$ に分けられ，$n_H = n_0 \cdot 2^{0 \cdot 17} + n_1 \cdot 2^{1 \cdot 17} + n_2 \cdot 2^{2 \cdot 17} + ... + n_{k-1} \cdot 2^{(k-1) \cdot 17}$ と表せる．同様に，計算結果 p は $k+1$ 個の符号なし 17bit 整数 $p_0, p_1, p_2, ..., p_k$ に分けられ，$p = p_0 \cdot 2^{0 \cdot 17} + p_1 \cdot 2^{1 \cdot 17} + p_2 \cdot 2^{2 \cdot 17} + ... + p_k \cdot 2^{k \cdot 17}$ と表せる．$n_0, n_1, n_2, ..., n_{k-1}$ は A ポートへ順に与えられ，$p_0, p_1, p_2, ..., p_k$ は P_L ポートから順に出力される．最下位 17bit から計算を行うことで，桁上がりを P_H レジスタに保存することができ，桁上がりの計算を円滑に行うことができる．以上より，$n \leftarrow B[n_L] \times n_H + C[n_L]$ の計算のアルゴリズムは以下のように表すことができる．

$P \leftarrow B[n_L] \times n_0 + C[n_L]$
for $i \leftarrow 0$ to $k-2$ **do**
  $p_i \leftarrow P_L, P \leftarrow B[n_L] \times n_{i+1} + P_H$
end for
$p_{k-1} \leftarrow P_L, p_k \leftarrow P_H$

また，動作周波数の向上を図るために図 2 のように DSP ブロックの内部レジスタを利用した．この DSP ブロックでは 3 ステージのパイプライン処理で $B[n_L] \times n_H + C[n_L]$ の計算を行うことができる．しかし，ブロック RAM から $B[n_L]$ と $C[n_L]$ 読み出す処理に 1 ステージを要するため，実際には 4 ステージのパイプライン処理となっている．

3.2 コプロセッサ回路

図 3 に既存実装のコプロセッサ回路を示す．図 3 の 1 つ目のブロック RAM は検証する初期値の下位 15bit の値を保存するために使用し，2 つ目のブロック RAM は 10bit の BC テーブルを保存するために使用する．コラッツ予想を検証する自然数 m は，46bit 整数 M，17bit 整数 j，15bit 整数 $S[i]$ で構成される．m は 6 個の 17bit レジスタに検証途中の暫定値 n として保存される．17bit レジスタの値は順に DSP ブロックの A ポートに与えられる．n の下位 10bit の n_L は BC テーブルへ入力され，BC テーブルの出力は，$B[n_L], C[n_L]$ となる．$B[n_L], C[n_L]$ はそれぞれ DSP ブロックの B ポート，C ポートへ与えられる．DSP ブロックを使用して $B[n_L] \times n_H + C[n_L]$ を計算した後に，コラッツ予想を検証する m と計算によって得られた検証途中の n を比較する．$n \geq m$ ならば n の値を再び 6 個のレジスタに保存する．$n < m$ ならば m についてはコラッツ予想が成り立つことが証明され，m の値を更新する．

また，全てのブロック RAM を**図 4** のようにデュアルポートで使用する．デュアルポート RAM は，異なるアドレスへの読み出しと書き込みを同時に行うことを可能にする．さらに，ブロック RAM に格納する BC テーブルと S テーブルは読み出ししか行わないので，それぞれを読み出し専用のデュアルポート ROM として保存し，2 つのコプロセッサで共有して使用する．

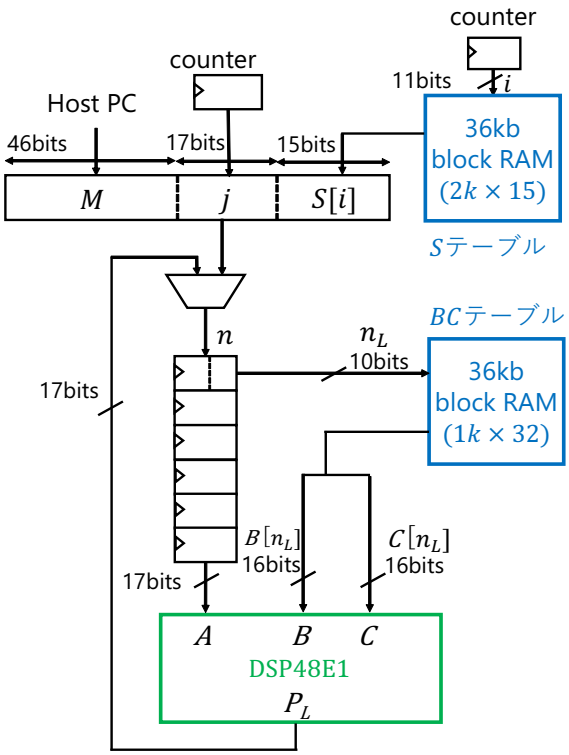

図 3 既存実装のコプロセッサ回路
Fig. 3 A coprocessor of previous implementation

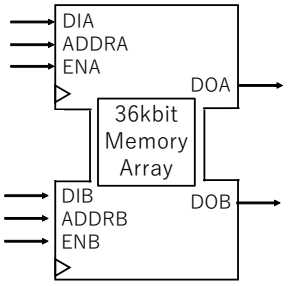

図 4 デュアルポートブロック RAM
Fig. 4 A dual-port block RAM

4. 提案実装

本論文の提案実装は，Virtex UltraScale+[7] を対象デバイスとして，既存実装に改良を加えたものである．**図 5** に提案実装のコプロセッサ回路を示し，本章でその改良部分を説明する．

4.1 S テーブルを外部入力

S テーブルは検証する自然数の下位 bit を構成するものである．既存実装のコプロセッサ回路では，1 つの M と 1 つの $S[i]$ に対して j を 0 から $2^{17}-1$ までループさせ検証を行う．その後，i の値を 1 増やし，再度同じように j をループさせる．つまり，M と $S[i]$ の 1 ペアに対して 2^{17} 個の自然数の検証を行うことになるため，S テーブルへのアクセス回数は 2^{17} 個の自然数の検証終了ごとに 1 回であ

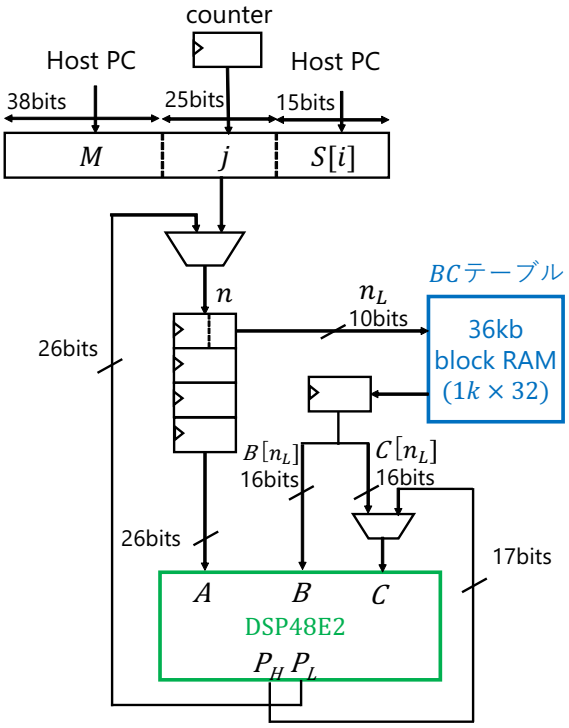

図 5　提案実装のコプロセッサ回路
Fig. 5　A coprocessor of our implementation

る．2^{17} 個の自然数の検証時間はホスト PC と FPGA 間の通信時間より非常に長いため，$S[i]$ の値も M と同様にホスト PC からの入力として与えても検証時間はほとんど変化しないと考えられる．よって提案実装では，図 5 のように $S[i]$ も M と同様にホスト PC からの入力で与えることでブロック RAM の使用数を半減させた．また，1 つの $M, S[i]$ のペアに対する検証個数は 2^j 個であり，j の bit 数を増やすことで検証個数と検証時間を線形的に増加させることが可能である．逆に j の bit 数が少ないと 1 つの $M, S[i]$ のペアについての検証がすぐに終わってしまい，頻繁にホスト PC との通信を行わなければならなくなる．提案実装ではホスト PC と FPGA 間の通信時間の隠蔽を目的として j の bit 数を 25 に増やしている．

4.2　計算 bit 数の拡張

既存実装の DSP ブロックでは，$P \leftarrow A \cdot B + C$ または，$P \leftarrow A \cdot B + P_H$ の計算が可能である．計算結果 P は上位 31bit の P_H と下位 17bit の P_L に分けられ，P_L は 17bit レジスタに保存され，P_H は次の計算に加算することで桁上がり計算を行うことができる．DSP ブロックは 17bit シフトのみに対応しているため，DSP ブロックの入力ポート A を 17bit に設計している．

図 6 に提案実装の DSP ブロック周辺の図を示す．提案実装では，17bit シフトを DSP ブロック外部に設置して DSP ブロックの入力ポート A の 17bit という制限を解除した．また，Virtex UltraScale+に搭載されている DSP ブロックは DSP48E2[11] であり，DSP48E1 との主な違いは入力ポート A が 25bit から 27bit に拡張されていることである．よって，DSP48E2 の符号なし乗算は 26bit×17bit まで対応しているため，DSP ブロックの入力ポート A を上限の 26bit とした．これにより 1 回の DSP ブロックでの計算 bit 数は 26bit×17bit となり，既存実装は 17bit×17bit であるため 9bit の拡張となっている．

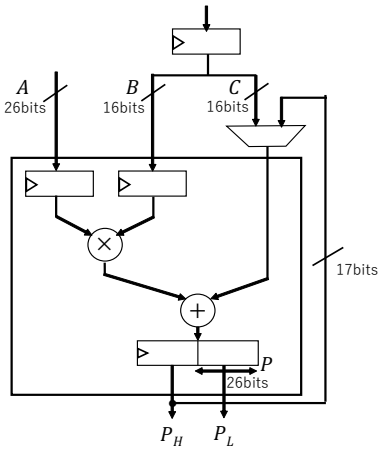

図 6　提案実装の DSP48E2 周辺のブロック図（四角内が DSP ブロック）
Fig. 6　A part of the block diagram of DSP48E2 of our implementation (DSP block correspond to inside square)

また，動作周波数の向上を図るために図 6 のように DSP ブロックの内部と周辺にレジスタを設置した．この DSP ブロックは 2 ステージのパイプライン処理で $B[n_L] \times n_H + C[n_L]$ の計算を行うことができる．しかし，外部にシフトを置いたことでレジスタが 1 つ DSP ブロックの外部に出たことと，ブロック RAM からの読み出しに 1 ステージを要することから，実際には既存実装と同様の 4 ステージのパイプライン処理となっている．

4.3　多倍長演算のワード数の固定

検証する自然数は複数のレジスタに分割保存され，1 つのレジスタを 1 ワードとして DSP を用いた多倍長演算を行う．例として 64bit 自然数を検証することを考える．既存実装では n_H の保存に $n_1, n_2, ..., n_6$ の 6 つの 17bit レジスタ用いているため，n_1, n_2, n_3, n_4 には必ず 0 以外の値が格納されており，n_5, n_6 に格納されている値は検証中の暫定値によって 0 の場合と 0 以外の場合がある．n_5, n_6 が 0 の場合は，DSP ブロックに入力して乗算を行う必要がなく，多倍長演算での使用するワード数 k は 4 となる．同様に n_6 のみが 0 の場合 $k = 5$ となり，n_6 が 0 でない場合には $k = 6$ となる．よって，既存実装の使用ワード数 k の変動は 4 ～ 6 である．

ここで，既存実装はレジスタの値によって使用ワード数が変動する回路であるため，パイプラインの条件分岐が増えて回路が複雑化し，リソースを多く使用してしまう．そ

こで提案実装では，図 5 のように n_H の保持に用いるレジスタを 4 つにして，演算に使用するワード数を $k = 4$ で固定した．これにより使用ワード数の変動がなくなり，回路が簡単化されるためリソースを削減できる．さらに，1 回の $B[n_L] \times n_H + C[n_L]$ に要するクロック数が 4 に固定になったため，4 ～ 6 のクロック数を要する既存実装より検証に必要な総クロック数も減少することが予想される．

また，既存実装では n_L の 10bit を含めて 112bit（17bit×6+10bit）まで暫定値を保存可能であるのに対して，提案実装では 114bit（26bit×4+10bit）であるため，対応 bit 数は 2bit 増えており，オーバーフロー確率は減少している．

5. 実験結果

本論文では，Xilinx Virtex UltraScale+ XCVU9P-L2FLGB2104E FPGA を対象デバイスとしてコラッツ予想の網羅的検証を行う回路を作成した．開発環境には Vivado Design Suite を使用し，使用言語はハードウェア記述言語の Verilog HDL を使用した．

提案実装では回路への入力として 24bit 入力 M を 100 個ランダムに生成し，それぞれの M に対してすべての j と $S[i]$ の値を検証するものとする．この検証に必要な総クロック数と回路の最大動作周波数を用いることで 1 秒間に計算可能な 64bit 自然数の個数を計算する．24bit 入力 M を 1 個検証するのに下位 40bit が変化するので，この検証方法では 100×2^{40} 個の 64bit 自然数の検証を行ったことになる．したがって，100×2^{40} 個の自然数の検証に必要な総クロック数を別途プログラムで計算した．なお，回路が 1 秒間に検証できる 64bit 自然数の数をスループットと定義する．

表 5 にシングルコプロセッサ回路においての既存実装と提案実装の論理合成結果と性能を示す．表 5 より提案実装は，既存実装と比べてブロック RAM の数は半減，CLB も減少している．これは，S テーブルをホスト PC からの入力としてことと多倍長演算で使用するワード数 k を固定にして回路を簡単化したことが要因である．このことから提案実装の方がリソース効率が良いことが分かり，マルチコプロセッサシステムとしたときにより多くのコプロセッサを FPGA 上に並べることができる．

コプロセッサ数を変えて計測した提案実装のマルチコプロセッサシステムの論理合成結果と性能を**表 6** に示す．提案実装で対象としたデバイスは，147780 個の CLB，2160 個の 36kb ブロック RAM，6840 個の DSP ブロックを持つ．1 つの CLB は 8 個の LUT と 16 個の FF を持つ．マルチコプロセッサシステムではブロック RAM をデュアルポート ROM として用いて 2 つのコプロセッサで 1 つを共有するため，コプロセッサ数の上限はブロック RAM の上限 2160 個の 2 倍の 4320 となる．

表 5 論理合成結果と性能 ($d = 15$)
Table 5 The results of synthesis and performance ($d = 15$)

	既存実装	提案実装
対象デバイス	XC6VLX240T -1FF1156	XCVU9P -L2FLGB2104E
最大動作周波数 (MHz)	360	521
スループット (個/s)	4.99×10^8	6.99×10^8
ブロック RAM 使用数	2	1
DSP ブロック使用数	1	1
CLB 使用数	52	45

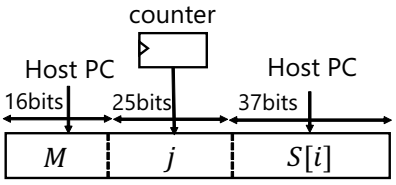

図 7 初期値の構成（$d = 37$）
Fig. 7 The structure of initial value ($d = 37$)

表 6 より，提案実装のスループット最大値はコプロセッサ数 4096 のときの 1.82×10^{12} 個/s である．コプロセッサ数が増加するほど最大動作周波数が下がる原因としては，回路が大きくなるにつれクロックの遅延が大きくなるからである．また，コプロセッサ数が上限の 4320 のときには最大動作周波数が著しく下がるため低いスループットとなっている．これは，回路生成時に FPGA 上に配置されているブロック RAM をすべて配線する必要があるために，配線の最適化の際に融通が利かなくなり長いパスを生成せざるを得なくなったことが原因であると考えられる．

また，既存 GPU 実装 [8] との比較も行った．この実装は，NVIDIA TITAN X を用いて行われており，18bit の BC テーブルと 37bit の S テーブルを使用して最大スループット達成している．表 3, 表 4 より下位 bit が大きくなるほど BC テーブルでは 1 回のテーブル操作による偶数・奇数操作数が増え，S テーブルでは検証が必要な自然数を限定できる．提案実装では 1 つの 36kb ブロック RAM に BC テーブルを格納して計算に用いているため，11bit 以上の BC テーブルを用いた実装はできない．しかし，S テーブルに関してはブロック RAM を使用せずホスト PC からの入力としたためブロック RAM の容量に制限されることなく拡張することが可能である．S テーブルを拡張することで検証が必要な自然数をより限定することができ，総クロック数が削減されスループットの向上が見込まれる．本論文では既存 GPU 実装との比較のため 37bit の S テーブルをホスト PC からの入力として与える回路を作成した．変更点は初期値の構成のみであり，図 5 上部の初期値の構成 $M, j, S[i]$ を**図 7** のようにして実装を行う．

スループットは先程と同様に検証に必要な総クロック数

表 6　提案実装の論理合成結果と性能 ($d = 15$)

Table 6　The result of synthesis and performance of our implementation($d = 15$)

コプロセッサ数	512	1024	1536	2048	2560	3072	3584	4096	4320
最大動作周波数 (MHz)	412	395	382	375	359	350	336	331	239
スループット ($\times 10^{12}$ 個/s)	0.28	0.54	0.79	1.03	1.23	1.44	1.62	1.82	1.39
LUT 使用数	122955	244933	367353	487911	609962	731962	853940	975891	1028312
FF 使用数	154524	308972	463420	617868	772316	926764	1081214	1235660	1302116
ブロック RAM 使用数	256	512	768	1024	1280	1536	1792	2048	2160
DSP ブロック使用数	512	1024	1536	2048	2560	3072	3584	4096	4320
CLB 使用数	24131	47435	71402	96061	117332	133185	143912	147224	147708

表 7　提案実装の論理合成結果と性能（$d = 37$）

Table 7　The result of synthesis and performance of our implementation($d = 37$)

コプロセッサ数	512	1024	1536	2048	2560	3072	3584	4096	4320
最大動作周波数 (MHz)	417	396	380	379	360	360	350	308	235
スループット ($\times 10^{12}$ 個/s)	1.04	1.98	2.85	3.79	4.50	5.40	6.12	6.16	4.95
LUT 使用数	122465	243905	365868	487798	604952	725912	843395	963871	1015948
FF 使用数	154524	308972	463420	617868	772316	926764	1081214	1235660	1302116
ブロック RAM 使用数	256	512	768	1024	1280	1536	1792	2048	2160
DSP ブロック使用数	512	1024	1536	2048	2560	3072	3584	4096	4320
CLB 使用数	24996	48128	73576	97906	121848	135250	143512	147263	147749

と最大動作周波数を用いて計算する．**表 7** に 37bit の S テーブルを入力とした提案実装のマルチコプロセッサシステムの論理合成結果と性能を示す．表 7 より，37bit の S テーブルを入力としたときの提案実装の最大スループットは 6.16×10^{12} 個/s である．既存 GPU 実装のスループットは 2.41×10^{12} 個/s より，提案実装は既存 GPU 実装と比較して 2.56 倍の性能を達成した．また，15bit の S テーブルを入力とした実装と初期値の構成以外は同じ回路であるためリソース量はほとんど変化していない．

6. まとめ

本研究では，コラッツ予想の網羅的検証を行う回路を Virtex UltraScale+ FPGA を対象デバイスとして実装した．既存実装を基に改良を行い，シングルコプロセッサにおいてのリソースを削減してより多くのコプロセッサを FPGA 上に並べることを可能にした．結果として，$d = 15$ のときのマルチコプロセッサシステムにおいて 1.82×10^{12} 個/s のスループットを達成した．また，$d = 37$ のマルチコプロセッサシステムでは 6.16×10^{12} 個/s のスループットを達成し，NVIDIA TITAN X を用いた既存 GPU 実装の 2.56 倍となった．

参考文献

[1] Tomas, O.: Maximum excursion and stopping time record-holders for the $3x + 1$ problem: Computational results, *Mathmatics of Computation,* 1999.

[2] Eric, R.: On the $3x + 1$ problem (online), 入手先 ⟨http://www.ericr.nl/wondrous/index.html⟩（参照 2018-07-25）.

[3] Fengwei, A. and Koji, N.: An architecture for veryfing collatz conjecture using an FPGA, *Proc. of the International Conference on Applications and Principles of Informatin Science*, pp.375–378, 2009.

[4] Yasuaki, I. and Koji, N.: Efficient Exhaustive Verification of the Collatz Conjecture using DSP blocks of Xilinx FPGAs, *International Journal of Networking and Computing*, Vol.1, No.1, pp.49–62, 2011.

[5] Yasuaki, I. and Koji, N.: A hardware-software cooperative approach for the exhaustive verification of the collatz conjecture, *Proc. of International Symposium on Parallel and Distributed Processing with Applications*, pp.63-70, 2009.

[6] Xilinx Inc: Virtex-6 Family Overview (online), 入手先 ⟨https://www.xilinx.com/support/documentation/data_sheets/ds150.pdf⟩（参照 2018-07-25）.

[7] Xilinx Inc: UltraScale Archtecture and Product Data Sheet: Overview (online), 入手先 ⟨https://www.xilinx.com/support/documentation/data_sheets/ds890-ultrascale-overview.pdf⟩（参照 2018-07-25）.

[8] Takumi, H., Yasuaki, I. and Koji, N.: GPU-accelerated Exhaustive Verification of the Collatz Conjecture, *International Journal of Networking and Computing*, Vol.7 No.1, pp.69–85, 2017.

[9] Amazon EC2 F1 instance (online), 入手先 ⟨https://aws.amazon.com/ec2/instance-types/f1/⟩（参照 2018-07-25）.

[10] Xilinx Inc: Virtex-6 FPGA DSP48E1 Slice User Guide (online), 入手先 ⟨https://www.xilinx.com/support/documentation/user_guides/ug369.pdf⟩（参照 2018-07-25）.

[11] Xilinx Inc: UltraScale Architecture DSP Slice User Guide (online), 入手先 ⟨https://www.xilinx.com/support/documentation/user_guides/ug579-ultrascale-dsp.pdf⟩（参照 2018-07-25）.