社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2003－ＳＬＤＭ－111（13）
2003／10／23

遺伝的アルゴリズムによる複数の定数乗算回路のハードウェア設計

込宮　英幸†　　佐々木孝雄†　　豊嶋　久道†

† 神奈川大学 工学部　〒221–8686 神奈川県横浜市神奈川区六角橋 3–27–1
E-mail: †{komi,sasaki,toyo}@tysm.ee.kanagawa-u.ac.jp

あらまし　複数の定数乗算はシフトと加算で実現されるが，ハードウェア設計としては加算に桁上げ保存加算器 (CSA) を用いることが多い．本研究では，機能評価の簡略化のため遺伝子を桁上げ伝播加算器 (CPA) として表現し，機能を満たす遺伝子を CSA に変換し，その回路規模，遅延時間を最小とするような回路を遺伝的アルゴリズムを用いて設計する方法を提案する。

キーワード　複数の定数乗算回路，遺伝的アルゴリズム，桁上げ保存加算器，半加算器

Hardware Design of Multiple Constant Multiplication Circuits Using Genetic Algorithm

Hideyuki KOMIYA†, Takao SASAKI†, and Hisamichi TOYOSHIMA†

† Faculty of Engineering, Kanagawa University　Rokkakubashi 3–27–1, Kanagawa-ku, Yokohama-shi, Kanagawa, 221–8686 Japan
E-mail: †{komi,sasaki,toyo}@tysm.ee.kanagawa-u.ac.jp

Abstract　A multiple constant multiplication (MCM) is represented as shifts and additions, and in a hardware design of MCM circuit, carry save adders (CSA) are often used for additions. This research proposes a genetic algorithm based MCM circuit synthesis method so as to minimize the circuit scale and delay time. In the proposed algorithm, though a gene is expressed as carry propagate adder (CPA) for simplification of functional evaluation, using transformation of CPA circuits into CSA circuits, CSA based hardware cost can be evaluated.

Key words　Multiple Constant Multiplicatoin, Genetic Algorithm,Carry Save Adder, Half Adder

1. はじめに

複数の定数乗算 (Multiple Constant Multiplication, MCM) 回路は複数の定数と変数間の乗算処理のことをいい，フィルタ，画像処理，数値計算などディジタル信号処理において頻繁に現われる回路であり，回路規模，消費電力の点からもその設計においては演算量を最小とする合成手法が望まれる．以下に入力 x に N 個の定数 $a_n(n=0,\cdots,N-1)$ を乗算した結果を y_n として出力する MCM の式，図 1 に回路図を示す．

$$y_n = a_n x \quad n = 0,\cdots,N-1 \tag{1}$$

MCM 回路はシフトと加算で実現し，各乗算器の演算を共有化する事で演算量，ハードウェア量をを削減することができるが，その組合せは無数にあり，最適化が極めて困難になることがある．これを MCM 問題といい，様々な解法が提案されている．

MCM 問題の解法として，回路をビット配列で表現する方法と，回路をグラフ表現する方法の二つに大別される．前者の手法は文献 [1]，文献 [2] で後者の方法は文献 [3]，文献 [4] で紹介されてい る．さらに 後者の考え方を進化論的アルゴリズムに適用した方法として文献 [5]，文献 [6], 文献 [7] がある．

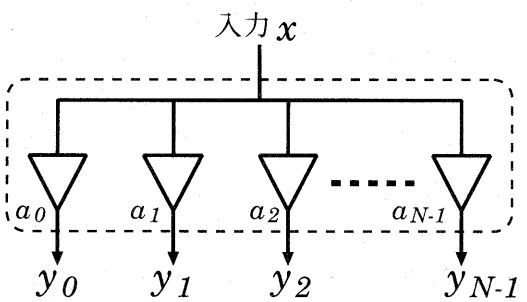


図 1　MCM 回路

文献 [6] では遺伝的プログラミング (Genetic Programming, GP) を採用しているが，遺伝子の表現が木構造に限定されるという問題がある．文献 [5] では，スタック型オペレータを遺伝

子とした遺伝的アルゴリズム (Genetic Algorithm, GA) による合成方法を採用しているが，複数のオペレータを遺伝子に使うと，定数の数やビット幅が増加した場合において探索範囲が膨大となり，局所解に陥りやすくなってしまう問題がある．そこで遺伝子情報として加算時の枝の組合せを並べたものを遺伝子とする [7] が提案された．枝情報を用いて遺伝子を表現することで，従来法と比べ遺伝子長が短くなり，無駄な遺伝子の生成確率を減少させることで，最適化の効率を向上させた．

MCM 問題では実際のハードウェアの加算に多く用いられる桁上げ保存加算器 (Carry Save Adder, CSA) ではなく桁上げ伝播加算器 (Carry Propagate Adder, CPA) を用いており，演算量として CPA の数しか考慮していないため，半加算器 (Half Adder, HA) や全加算器 (Full Adder, FA) 等ハードウェアレベルで見ると，より最適な MCM 回路が設計できる余地が残されている．そこで，本研究では組合せ最適化手法の一つである GA を用いた回路の設計法を提案する．遺伝子の表現法は機能評価の簡略化のため CPA として表現し，機能を満たす遺伝子を CSA に変換する手法をとる．こうすることで探索範囲の広さを維持したまま，効率の良い回路の探索ができ，その回路規模，遅延時間を最小とするような回路を設計することができる．

2. GA による複数の定数乗算回路の合成手法

GA は生物の進化の過程を模倣した組み合わせ最適化手法の一つである．解を表現する遺伝子の集団を選択・淘汰，交叉，突然変異を行い，次の世代の遺伝子を生成する．この操作を終了条件を満たすまで繰り返すことで，最適な解を探索する．以下に基本的な GA の流れを示す．

(1) 初期集団を生成する．

(2) 全ての遺伝子の適応度を算出する．

(3) 終了条件を満たしていれば終了，それ以外は (4) へ．

(4) 選択・淘汰：適応度により次世代に残す遺伝子を選択する．

(5) 交叉：交叉率に従って，新しい遺伝子の親となる遺伝子を交叉させる．

(6) 突然変異：突然変異率に従って，遺伝子を変形させ (2) へ．

2.1 遺伝子の表現と生成法

複数の定数乗算の回路合成において回路規模，遅延時間を減少させるには，定数を求める過程で加算される枝情報が重要となる．本研究では機能評価の簡略化のため遺伝子を CPA として表現し，機能を満たす遺伝子を CSA に変換する方式をとるため，遺伝子表現は枝情報から二値を加算する毎にその枝番号を二つずつ並べたものとする．加算する二値を選び出す方法にはランダムに選択，最も定数に近くなる組合せを選択の二通り用意する．こうすることにより，回路構造の多様性を確保したまま，高い確率で機能を満たす回路の生成が可能である．

以下に遺伝子長 L，遺伝子の各要素を $g_l(l=0,\cdots,L-1)$，定数の数 $N=2$，定数 $\{a_n\}=\{27,52\}$ の場合の遺伝子生成法を示す．

(1) 与えられた定数 $\{a_n\}$ をそれぞれ奇数になるまで 2 で割続け，各定数を 1 乗数とした定数乗算回路のコストを測り，それらで昇順にソートしたものを $\{a'_n\}$ とする．

$$\{a_n\}=\{27,52\} \Rightarrow \{a'_n\}=\{13,27\} \tag{2}$$

枝情報 $I_k(k=0,\cdots,\lfloor\log_2 a'_{N-1}\rfloor)$ を用意する．

(2) 遺伝子 $\{g_l\}$，枝情報 $\{I_k\}$ を全て 0，$k=l=n=0$，$I_k=1$(入力) と初期化する．

(3) 枝情報 $I_{k+m+1}=I_{k+m}\times 2(m=0,\cdots,M,\ M=\lfloor\log_2(a'_{N-1}/I_k)\rfloor)$ と更新する (図 2(a))．$k=k+m+2$ にする．

(4) 枝情報より加算を行う二つの枝番号 k_1, k_2 を以下の方法で選び出し，$I_{k_1}+I_{k_2}$ を I_k に格納する．但し $I_{k_1}+I_{k_2}>a'_{N-1}$ である場合は，k_1, k_2 を選び直す．また，$n=N$ の場合はランダムに k_1, k_2 を選ぶ．

- 確率 p：ランダム
- $1-p$：$I_{k_1}+I_{k_2}$ が最も a'_n に近い組合せ

$I_k=a'_n$ の場合，$n=n+1$ とする．$g_l=k_1$, $g_{l+1}=k_2$ として遺伝子を更新し，$l=l+2$ にする．(図 2(b)，3(a))

(5) (3)(4) の操作を $l<L$ の間繰り返す．(図 2(c)，2(d)，3(b)，3(c))

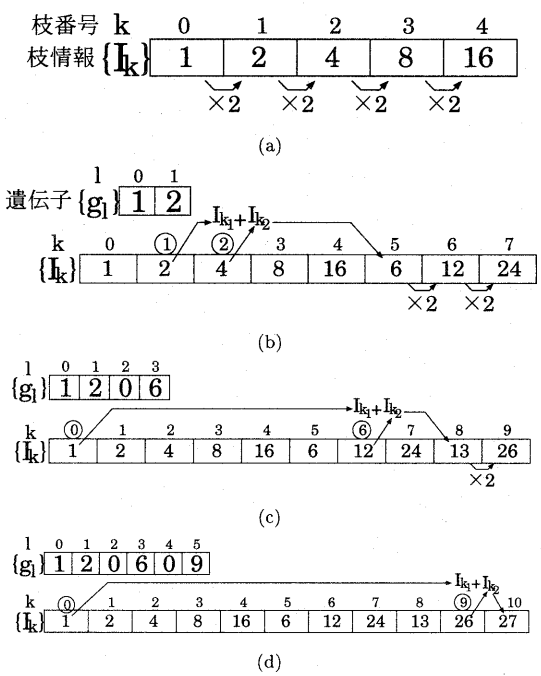
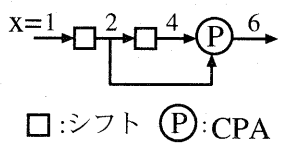


図 2 遺伝子の生成

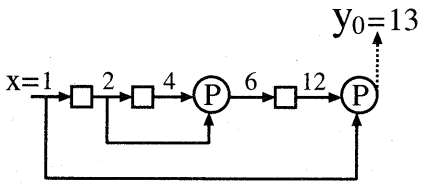
以上の操作により定数 $\{a_n\}=\{27,52\}$ を表す遺伝子が $\{g_l\}=\{1,2,0,6,0,9\}$ と表現された．

3. 遺伝子の評価

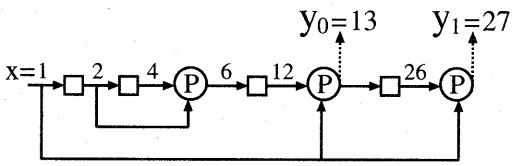
CPA で表現された機能を満たす遺伝子を CSA に置き換え HA の数，遅延時間を算出し評価する．



(a) 遺伝子 $\{g_l\} = \{1,2\}$



(b) 遺伝子 $\{g_l\} = \{1,2,0,6\}$



(c) 遺伝子 $\{g_l\} = \{1,2,0,6,0,9\}$

図 3　遺伝子が表現する回路

3.1　CSA への変換方法

（1）　4 行 $(L/2)$ 列の二次元配列 X, Y, Z を用意する．X には加算に用いた数値，Y には各 X の値のシフトする以前の値，Z には CSA の二出力か否かを判別するための値を代入する．$j = 0, \cdots, (L/2)-1$ に対して X_{0j}, X_{1j} に加算値 $I_{g_{2j}}$, $I_{g_{2j+1}}$ を，Y_{0j}, Y_{1j} に $I_{g_{2j}}$, $I_{g_{2j+1}}$ をシフトする前の値を，Z_{0j}, Z_{1j} に 1 を代入する (図 4)．

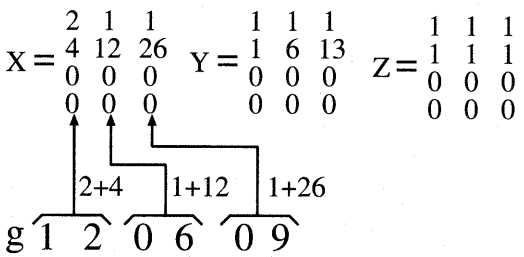


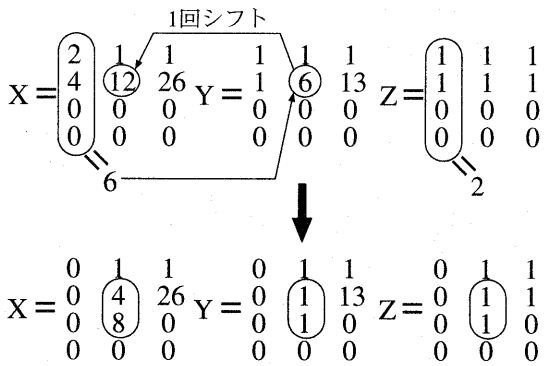
図 4　変換準備

$j = 0, \cdots, (L/2)-1$ に対して以下の (2)〜(4) の操作を繰り返す．

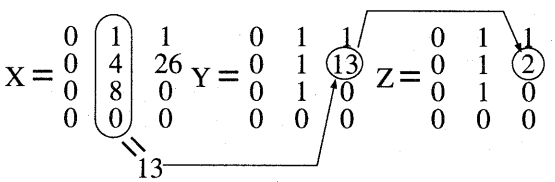
（2）　Z の j 列目の総和を取り，その値が 2 なら二値の加算つまり CPA の加算を示すので CSA にするため (3) へ．2 より大きい場合すでに CSA に変換済みなので (4) へ．

（3）　X の j 列目の総和と等しい値を j より大きい列の Y 内で探す．もし等しい値が $Y_{i'j'}$ に存在する場合，$X_{i'j'}$ を X の j 列目の値で置き換えることができる．$Y_{i'j'}$ が $X_{i'j'}$ になるまでのシフトの回数 (k に相当) をカウントし，$X_{i'j'}$ を 0 にした後，X の j' 列の値が 0 の所に k 回シフトした，値が 0 でない X の j 列目の値を入れる．X の j 列目の値は全て 0 にする．X の移動に応じて Y, Z も移動させる (図 5(a))．X の j 列目の総和と等しい値が Y 内に存在しない場合は，何も操作を行わない．$j = j+1$ にし (2) へ．

（4）　X の j 列目の総和と等しい値を Y 内で探す．もし $Y_{i'j'}$ に存在する場合，その値は CSA の二出力なので $Z_{i'j'}$ の値を 2 にする (図 5(b))．$j = j+1$ にし (2) へ．



(a)



(b)

図 5　変換行程

最後に CSA の 2 出力で表現された各定数を CPA に連結する．

以上の操作によって変換された図 3(c) の回路を図 6 に示す．

なお X, Y, Z の 4 行目は，図 7 のように，変換される数値が同じ列に 2 つある場合に使用する．

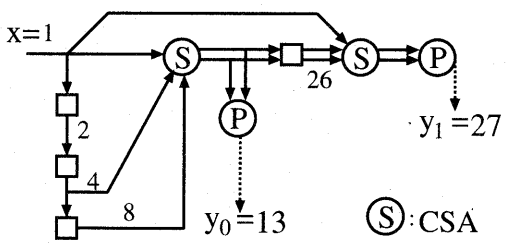


図 6　回路変換結果

3.2　HA，FA，遅延時間のカウント方法

回路を CSA へ変換した後で，回路の HA，FA，遅延時間などのハードウェアコストをカウントする．この時，加算器の入

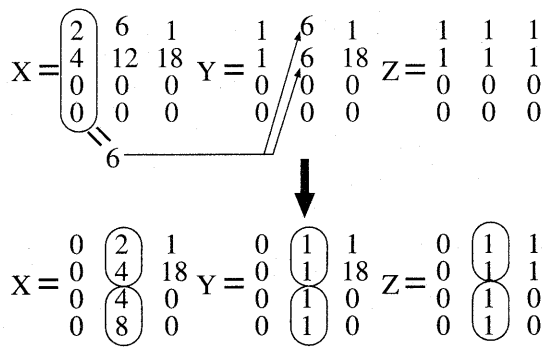


図 7　4 行目使用例

出力のビット幅の情報が重要となる．本研究はビット幅を表現する際，零ビットと有効ビットを用いた．零ビットとはシフトにより生成され値が必ず 0 であるビットをいい，有効ビットとは 0 または 1 どちらかの値に変化する可能性のあるビットを表す．入力値は全て有効ビットで表され，シフト，加算を通して有効ビットが変化するのを追いながら HA，FA，遅延時間のカウントを行う．CSA の場合，加算を行う 3 入力の同じ重みで有効ビットの重なりからカウントする．有効ビットが 1 つの場合は有効ビットを中間和 s へ出力する．有効ビットが 2 つの場合は HA を用いて，s と桁上げ出力 c へ有効ビットを出力する．3 つの場合は FA を用いて，HA と同様に s, c へ有効ビットを出力する (図 8)．

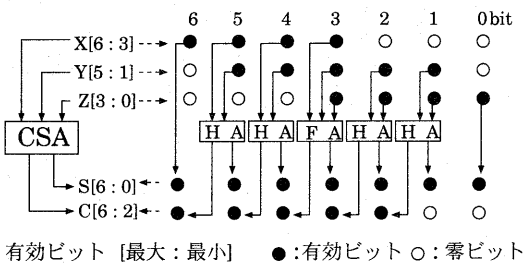


図 8　HA，FA のカウント (CSA)

CPA の場合では CSA の 3 入力目を下位からの桁上げ入力とし，s を和，c を上位への桁上げと考える事で，同様の方法で HA，FA をカウントすることができる (図 9)．

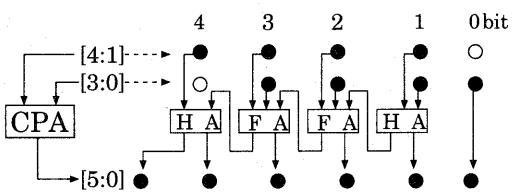


図 9　HA，FA のカウント (CPA)

各加算器の遅延時間は CSA の場合，各重みの加算が並列に行われるため，1ヶ所でも FA を使えば FA 一つ分，HA のみなら HA 一つ分，何も使用しない場合は 0 という遅延時間となる．CPA は HA，FA が直列に連結しているため使用した HA，FA の数だけ遅延時間がかかる．本研究では HA 一つ分の遅延時間を τ と表現した．また，FA は面積，遅延時間共に HA 二つ分としてカウントした．

以上の方法で各加算での HA，FA 数，遅延時間が分かるので，回路全体の加算の総和を取れば回路のコストを求める事ができる．だが遅延時間の場合，回路によっては並列に加算が行われるため，各加算での遅延の総和が回路全体の遅延とは必ずしもならないため，並列に行われる加算を見つける必要がある．

並列な加算を探すには CSA 変換時に用いた X, Y, Z に加え加算段数より求める．

（1） X, Y, Z と同じ列幅 $(L/2)$ を持つ加算段数の配列 $\{S_j\}$ を用意し全ての数値を 1，$j = 0$ に初期化する．

（2） X の j 列目の総和を取り，Y 内で探索し Y の i 列目で発見された場合，$S_i = S_j + 1$ とする (図 10(a))．この時，$S_i > S_j + 1$ の場合は何もしない．また，i 列目の Z の総和が 4 の場合はさらに S_i に 1 を加える．

（3） $j = j + 1$ とし，$j < (L/2)$ の間 (2)(3) の操作を繰り返す (図 10(b))．

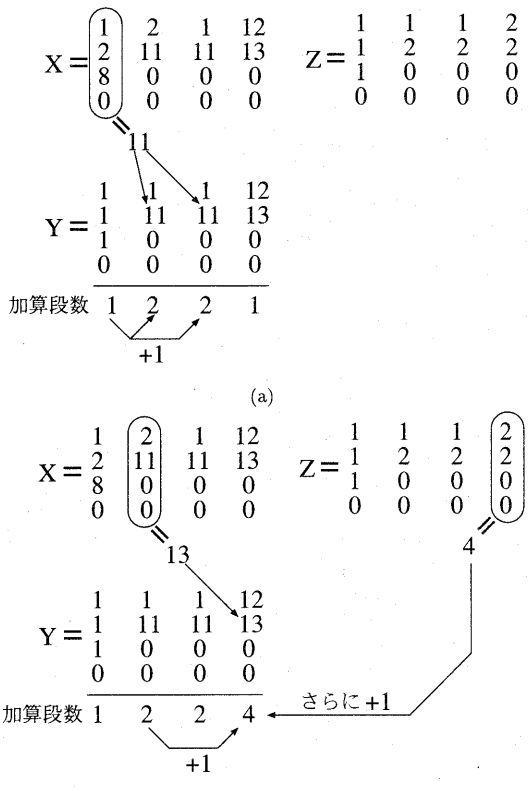


図 10　並列加算の探索

以上の方法で求めた加算段数のうち，並列に行われている加算では最大の遅延時間を採用し，全 CSA の遅延時間の総和に加え，CSA の 2 出力で表現された a'_{N-1} 番目の定数に連結さ

れる最終加算 (CPA) の遅延時間が回路全体の遅延時間となる．図 10 の加算段数 1 段目の加算による，遅延時間のカウントを図 11 に示す．

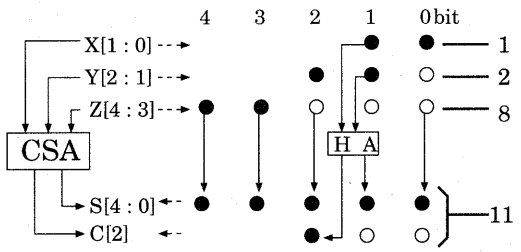


図 11　遅延時間のカウント (入力ビット幅 2bit)

図 11 より 1 段目の遅延時間は 1 τ となる．図 10 を回路表現したものを図 12 に示す．

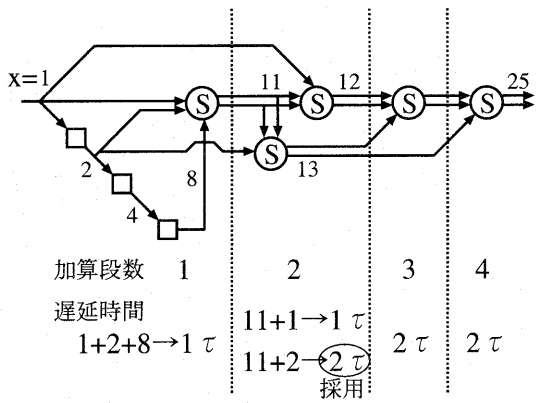


図 12　回路の遅延時間

図 12 よりこの回路の遅延時間 (CSA のみ) は 7 τ となる．

3.3　適応度の算出法

適応度 P を算出する方法を以下に示す．

$$P = \frac{F}{N}\frac{1}{AD} \tag{3}$$

A は HA の数，D は遅延時間 (τ)，N は求める定数の数，F は求まった定数の数を表す．FA は HA 二つ分としてカウントする．

3.4　GA オペレータ

提案法で扱う GA オペレータとして以下の方法を採用した．

- 選択：エリート保存戦略を用いて適応度の高い遺伝子を残し，ルーレット選択を用いて適応度の低い遺伝子も残る可能性を残しておく．
- 交叉：一点交叉を用いる．交叉率により親遺伝子を二つ選び出し，ランダムに交叉点を決め，交叉を行う．
- 突然変異：突然変異率により，突然変異を行う遺伝子を選択し 50%の確率で以下の操作のどちらか一方を行う．
  - 遺伝子一箇所変更：遺伝子のランダムな位置一点の要素を新しく生成させる．
  - 新しい遺伝子の生成：初期集団と同様の生成法で全く新しい遺伝子を生成させる．

4.　設　計　例

表 1 の遺伝的パラメータを与え，MCM 回路の設計を行い，HA 数の比較，遅延時間の比較を行った．比較対象には文献 [4] で設計した回路を CSA に変換したものを用いた．

表 1　設計に用いた遺伝的パラメータ

遺伝子長	100
世代数	500
1 世代の個体数	500
交叉率	70%
突然変異率	10%
ランダム率 p	50%

・例 1　定数 $= \{13, 55, 79\}$

HA 数の比較を図 13，遅延時間の比較を図 14 に示す．

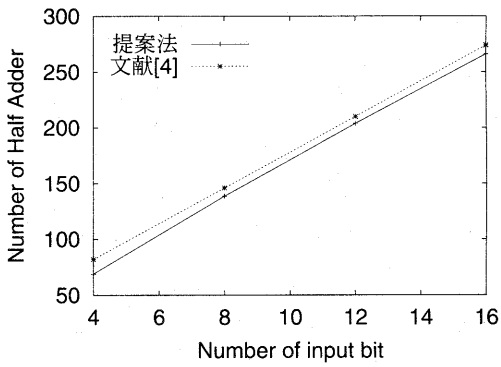


図 13　HA 数の比較　定数 $= \{13, 55, 79\}$

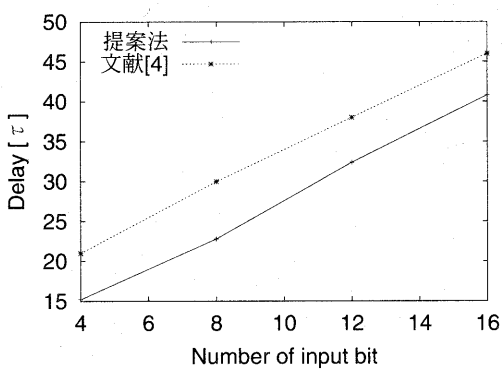


図 14　遅延時間の比較　定数 $= \{13, 55, 79\}$

・例 2　定数 $= \{117, 283, 361, 495\}$

HA 数の比較を図 15，遅延時間の比較を図 16 に示す．

・例 3　定数 $= \{531, 953, 1845, 5907, 16739\}$

HA 数の比較を図 17，遅延時間の比較を図 18 に示す．

図 13〜18 より従来法と比べ，より小規模，高速な回路の探索ができた．また，求める定数の数が多く，定数の値が大きい程，よりハードウェアコストを削減することができる．

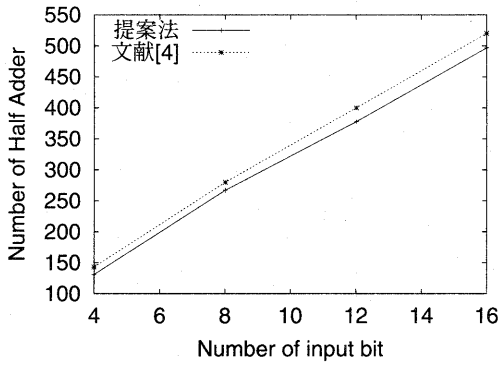


図 15　HA 数の比較　定数 = {117, 283, 361, 495}

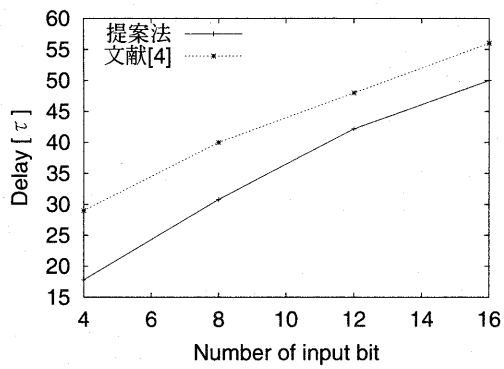


図 16　遅延時間の比較　定数 = {117, 283, 361, 495}

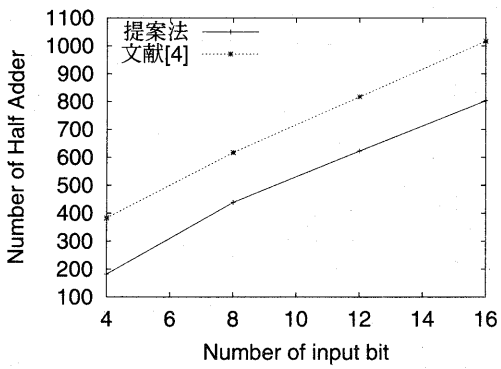


図 17　HA 数の比較　定数 = {531, 953, 1845, 5907, 16739}

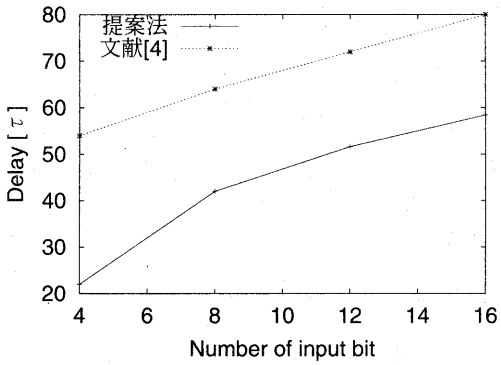


図 18　遅延時間の比較　定数 = {531, 953, 1845, 5907, 16739}

5. むすび

本研究では，MCM 回路を CSA を用いて，ハードウェアレベルで遺伝的に設計をする方法を提案した．機能の簡略化のため遺伝子を表現する際には CPA を用い，機能を満たすものを CSA に変換して評価することにより，探索範囲を広く維持したまま，無駄な遺伝子の生成を減少させ，効率良く回路の探索を行うことにより回路規模，遅延時間の小さい，回路の合成が行えた．

今後の課題として加算のみならず，減算も考慮しより小さく，高速な回路の設計方法を検討していく．

文　献

[1] M. Potkonjak, M. B. Srivastava, and A. P. Chandrakasan, "Multiple constant multiplication: Efficient and versatile framework and algorithms for exploring common subexpression elimination", IEEE Trans. Computer-Aided Des. Integrated Circuits & Syst., vol.15, no.2, pp.151–165, Feb. 1996.

[2] A. Matsuura, M. Yukishita, and A. Nagoya, "A Hierarchical Method for the Multiple Constant Multiplication Problem", IEICE Trans. Fundamentals vol. E80-A, No.10, pp.1767–1773, Oct. 1997.

[3] D. Bull, D. Horrocks, "Primitive operator digital filters", IEEE Proceedings-G Vol.138, pp.401–411, June 1991.

[4] A.G.Dempster and M.D.Macleod,"Use of minimum adder multiplier blocks in FIR digital filters, " IEEE Trans. Circuits Syst. II, vol. 42, pp. 569-577, Sept. 1995

[5] 磯尾洋介，豊嶋久道，"遺伝的アルゴリズムによる複数の定数乗算回路の自動合成 "，第 14 回 回路とシステムワークショップ，pp.225–230, Apr. 2001.

[6] H. Safiri, M. Ahmadi, G. A. Jullien and W. C. Miller, "A New Algorithm for the Elimination of Common Subexpressions in Hardware Implementation of Filters by Using Genetic Programming", Journal of VLSI Signal Processing 31, pp.91–100, 2002.

[7] 天川昌充，豊嶋久道，"遺伝的アルゴリズムによる複数の定数乗算回路の最適合成手法"，第 17 回 ディジタル信号処理シンポジウム，B2–3,

[8] D. Chen, T. Aoki, N, Homma, T. Terasaki, and T. Higuchi, "Graph-Based Evolutionary Design of Arithmetic Circuits", IEEE Transactions on Evolutionary Computation, vol. 6, No. 1, February 2002

[9] 北野 宏明 編，"遺伝的アルゴリズム"，産業図書，1993.