# GPを用いたディジタル回路の合成における遺伝子表現の検討

石田 康太郎†　　新井 浩志†
千葉工業大学 大学院†

## 1. はじめに

入出力事例だけをもとにしてディジタル回路を合成するための手法として,遺伝的プログラミング(以下 GP : Genetic Programming)を用いた方法が研究されている[1]. この様な研究では遺伝子の表現に論理ゲートを用いている. しかし,合成したい回路の規模が大きくなると探索領域が指数関数的に増大するため,一定以上の規模の回路の合成は困難であった. そこで本発表では,探索領域の抑制を目的として遺伝子を真理値表で表現する方法を提案し,遺伝子を論理ゲートで表現した場合との比較結果について報告する.

## 2. 従来手法

GP を用いてディジタル回路を合成するためには,ディジタル回路を遺伝子として表現する必要がある. 本研究ではムーア型の順序回路を対象とする. ムーア型の順序回路は,状態を保持する D-FF 群,FF 群の出力と回路の入力から FF 群の入力を決定する次状態決定回路,FF 群の出力から回路全体の出力を決定する出力決定回路に分けられる. 次状態決定回路と出力決定回路はそれぞれ組合せ回路であり,従来の我々の研究[2]では,これらを中間木の集合,出力木の集合としてゲートで表現していた. この遺伝子の例を図1に示す.

GP では,まず初期集団を生成したのちに,入出力事例すなわち教師データを用いて適合度計算を行う. 適合度とは,教師データと個体の出力がどれだけ近いかを評価した値である. 個々の個体に対して,教師データの入力を与えた場合の出力をシミュレーションにより求め,これと教師データの出力との一致割合を算出し,これを適合度とする. この適合度をもとに個体を選択し,交叉,突然変異などの GP 操作を行って新たな個体群を得る. この処理を繰り返す事でディジタル回路を合成する.

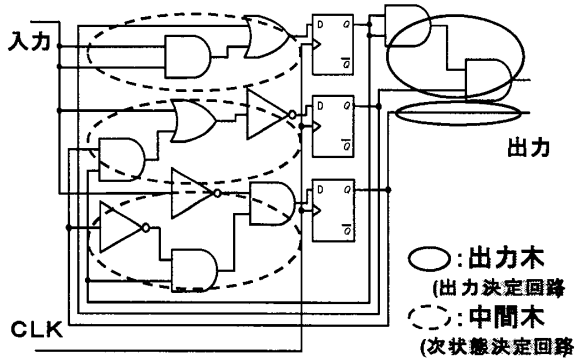


図 1. 順序回路の例

A study about Gene Coding on a GP-based Digital Circuit Synthesis
† Koutarou Ishida, Hiroshi Arai
Graduate School of Engineering, Chiba Institute of Technology.

## 3. 真理値表による遺伝子表現

ゲートによる遺伝子表現を用いた場合,1 つの論理関数を表現するためのゲートの組合せが無数にある. このため,目的とする回路の規模が大きくなるほど解の探索空間が広くなりすぎてしまい,結果として回路の合成が困難になる. そこで遺伝子の組合せ回路部分,すなわち個々の中間木と出力木を真理値表で表現する手法を提案する. これにより,1 つの論理関数の表現形式が唯一となり,探索空間が抑制されると考えられる. また,従来は一つの木の出力をゲートレベルのシミュレーションで求めていたが,これを真理値表の参照で実現できるため,シミュレーションに要する時間を短縮できると考えられる.

従来手法では出力木や中間木の一部を交換する事により遺伝子の交叉を実現していた. しかし,遺伝子を真理値表表現に変更した事にともない,真理値表での交叉処理が必要になる. 2 つの真理値表を遺伝子とみなして交叉させる場合,その真理値表が表現している論理関数の入力変数が全く同一であれば,真理値表の出力に相当するビット列の交叉によって実現することができる. 例えば 2 つの論理関数 f と g の入力変数の集合を $I_f$, $I_g$ とした場合, $I_f$=$I_g$ であれば真理値表の行数は等しく交叉は容易である. しかし,一般には $I_f \neq I_g$ であるので,入力変数を揃える事を目的として以下の伸長,圧縮処理を定義する.

1)伸長処理

f(A,B)に対して入力 C を追加し g(A,B,C)を作成する処理

$$\begin{cases} g(A,B,0)=f(A,B) \\ g(A,B,1)=f(A,B) \end{cases}$$

2)圧縮処理

f(A,B)に対して入力 B を削除して h(A)を作成する処理

h(A)=f(A,B)　　…f(A,0)=f(A,1) の時

h(A)= {f(A,0), f(A,1)} ランダムに決定　…f(A,0)≠f(A,1)の時

提案する手法では遺伝子の真理値表を入力部分と出力部分に分けて保持しているため入力の交叉と出力の交叉の 2 段階に分けて行う.真理値表で表現された論理関数 f と g を交叉する場合,入力の交叉ではまず双方の関数の入力変数をランダムに 1 つずつ選び出し,それを交換する処理を行う.例えば f の入力変数 A と g の入力変数 B が交叉対象として選ばれた場合,f に対して A を圧縮して B を伸長し,g に対して B を圧縮して A を伸長する.ただし g の入力変数に A がすでに含まれていた場合は,g に対しては B の圧縮処理だけを適用する.

出力の交叉では,まず 2 つの論理関数の入力変数の集合を同じにするために伸長処理を行う.例えば,論理関数 f と g に対して出力の交叉を行う場合 $I_g$ に含まれていて $I_f$ に含まれていない全ての入力変数に対して f を伸長し, $I_f$ に含まれていて $I_g$ に含まれていない全ての入力変数に対して g を伸長する.次に真理値表の出力部分に相当するビット列に対して 2 点交叉を行った後に,先の処理で伸長した入力変数を全て圧縮する.

## 4. 実験結果

遺伝子にゲート表現を用いた場合と真理値表表現を用いた場合の回路の合成結果を表 1 に示す.適合度 100%の個体が得られるか,又は 10000 世代経過した場合を終了条件とし,各々10 回実行した.以下 100%の個体が得られた場合を成功とする.成功率とは,10 回の試行において成功した割合を示す.世代数とは,合成に成功した時の進化に要した世代数の平均である.また時間も同様に合成に成功した時の時間の平均である.

表 1 からわかる通り,ゲートで表現されていた遺伝子を真理値表の表現にする事で,成功率が上がり,合成に要する世代数と計算時間を減少させる事ができた.これは,遺伝子をゲート表現から真理値表表現に変更した事により,1 つの論理関数の表現形式が唯一となり探索空間が抑制されたためであると考えられる.

しかし,より大きな回路を合成しようとした場合には,1 世代の処理に多くの時間を要してしまい回路の合成に至らなかった.この原因として以下の 2 点があげられる.まず,世代数が進むと突然変異によって FF が追加される場合があるため,各真理値表の入力変数の数が増える.このため真理値表が大きくなりやすく,交叉などの処理に要する時間が増大する.また,個体の適合度計算に要する時間については n ゲート分の論理シミュレーションを行う従来手法よりも真理値表を 1 回だけ参照する提案手法の方が早い.しかし,教師信号の入力に P 個の不定が含まれる場合には真理値表を $2^P$ 回参照することになる.このため,真理値表の入力変数が多く,さらに不定が多く含まれる場合には,ゲートの表現の方が 1 世代の処理に要する時間が短くなる.

## 5. まとめ

今回対象とした回路規模においては,遺伝子に真理値表表現を用いた場合の方が少ない世代数で合成することが可能であった.今後は遺伝子表現に BDD(Binary Decision Diagram)を使うことで論理関数の表現形式を唯一にしたまま不定入力に対する処理を抑える方法を検討中である.

参考文献


[1] A.P.Shanthi et.al.," Evolution of Asynchronous Sequential Circuits", Proc. of the 2005 NASA/DoD Conference on Evolvable Hardware. IEEE Computer Society,2005.

[2] 松井小百合 他," 遺伝的プログラミングを用いた論理回路合成における出力分解の適用",第 6 回情報科学技術フォーラム(FIT2007),F-037,2007.


表 1：回路の合成結果

| 合成対象回路 | ゲート表現を用いた場合 | | | 真理値表表現を用いた場合 | | |
|---|---|---|---|---|---|---|
| | 成功率(%)※1 | 世代数※2 | 時間(s)※2 | 成功率(%)※1 | 世代数※2 | 時間(s)※2 |
| 2bitバイナリカウンタ | 100 | 416 | 5.74 | 100 | 175 | 4.30 |
| 3bitバイナリカウンタ | 30 | 6587 | 150.36 | 100 | 747 | 56.34 |
| 4bitバイナリカウンタ | 0 | - | - | 20 | 2329 | 404.84 |
| 状態遷移機械A(3状態) | 100 | 1653 | 12.69 | 100 | 309 | 6.70 |
| 状態遷移機械B(4状態) | 0 | - | - | 50 | 2370 | 2073.53 |
| 状態遷移機械C(4状態) | 40 | 4960 | 110.87 | 70 | 897 | 191.31 |

※1 10000 世代までの成功率

※2 合成に成功した場合の平均世代数と平均時間