計測自動制御学会東北支部 第 199 回研究集会 (2001.12.15)
資料番号 199-11

進化的グラフ生成システム EGG
– オブジェクト指向フレームワークの設計 –

Evolutionary Graph Generation System
– Design of an Object-Oriented Framework –

○茂木 誠*，青木孝文*，樋口龍雄*

○ Makoto Motegi*, Takafumi Aoki*, and Tatsuo Higuchi*

*東北大学大学院情報科学研究科

*Graduate School of Information Sciences, Tohoku University

キーワード： 回路設計 (circuit design), 算術回路 (arithmetic circuits), 進化論的計算手法 (evolutionary computation), オブジェクト指向フレームワーク (object-oriented framework)

連絡先： 〒 980-8579　仙台市青葉区荒巻字青葉 05 東北大学大学院情報科学研究科 樋口研究室
茂木 誠，Tel.: (022)217-7169，Fax.: (022)263-9406，E-mail: motegi@higuchi.ecei.tohoku.ac.jp

1. はじめに

近年，科学技術計算やディジタル信号処理などの分野において要求される演算能力は増加の一途をたどっており，用途に応じた多種多様な演算回路の設計が必要とされている．また，近年，従来の 2 進数にとらわれず，冗長数系や多進数系などの特殊数系を積極的に活用したハードウェアアルゴリズム (Beyond-Binary Arithmetic) の有効性が示されている．このような設計手法の多様化に伴い，各用途ごとに最適な算術演算回路を設計することは，設計者の経験と知識がシステムの性能を左右する困難な作業になりつつある．

そのような状況に対して，筆者の所属する研究グループより，進化的グラフ生成手法 (EGG: Evolutionary Graph Generation) が提案されている．EGG は，進化的計算手法に基づいて回路構造を自動的に合成する手法である．EGG は，回路構造をグラフ構造として抽象化し，そのグラフ構造への直接的な構造操作を行うことにより，回路構造の最適化を効率的に実現する．これまで，算術演算回路の例として，16 ビット精度の定係数乗算回路や，順序回路型の加算回路などの合成が実現され，EGG の有効性が示されている [1)–3)]．

本稿では，さまざまな問題に幅広く EGG を適用するため，オブジェクト指向フレームワークと呼ばれるシステム設計法を導入し，拡張性の高い EGG システムを構築する．オブジェクト指向フレームワークとは，設計するシステムを各種応用（アプリケーション）に固有な部分とアプリケーションに依らない共通した部分とに分け，共通な部分をフレームワークとして構築する手法である．これにより，アプリケーションは，フレームワークにアプリケーション固有の部分を追加するのみ

で構築することが可能となる．一方，EGG は，いかなる構造の合成に適用した場合でも，生物の進化戦略を模擬したシステムフローや個体の構造操作の方法などは共通している．これらの共通点をあらかじめ構築しておき，問題によって異なる部分のみを追加するということができれば，さまざまな構造の合成に EGG を適用する作業の大幅な効率化が期待できる．そこで，EGG にオブジェクト指向フレームワークの概念を導入することにより，拡張性の高い EGG システムの構築を目指す．

以下では，まず，EGG の概念について述べ，構築した EGG システムのオブジェクト指向フレームワーク（EGG フレームワーク）を示す．その上で，EGG フレームワークを用いて開発したさまざまな回路合成へのアプリケーション（EGG アプリケーション）を示す．

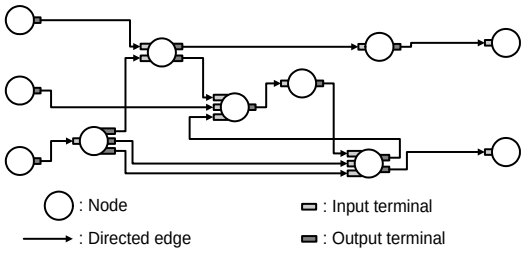

Fig. 1　回路グラフの例

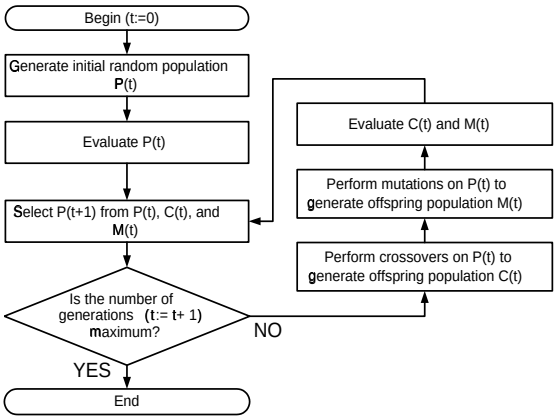

Fig. 2　EGG のシステムフロー

2.　進化的グラフ生成手法 EGG

進化的グラフ生成手法 (EGG: Evolutionary Graph Generation) は，進化的計算手法の 1 つととらえることができる．一般に，進化的計算手法とは，与えられた環境に適応する優れた個体が世代を重ねることで出現するという自然進化の過程を模擬した探索手法である．進化的計算手法としてよく知られている遺伝的アルゴリズム (GA: Genetic Algorithms) や遺伝的プログラミング (GP: Genetic Programming) では，それぞれ文字列や木構造を個体として用いる．しかし，回路構造を対象とした問題を取り扱う場合，文字列や木構造を用いた個体の表現方法は問題の解を直接的に表現するものではない．一方，EGG では回路構造を直接的に表現可能なグラフを個体として用いる．さらに，個体に対するグラフ論的な構造操作を定義することにより，効率的に回路構造を探索することが可能となる．

EGG では個体を Fig. 1 のような回路グラフとして表現する．回路グラフはノードの集合と有向辺の集合からなる．ノードには機能ノード，入出力ノードがあり，各ノードは機能および複数の入出力端子を持つ．有向辺は必ずノードの出力端子から入力端子へ向かうものとし，出力端子と入力端子は常に一対一で対応するものとする．ここで，未接続の端子が存在しない回路グラフを完全回路グラフと呼び，EGG では完全回路グラフのみを対象とする．

Fig. 2に EGG のシステムフローを示す．はじめに，初期世代として，設定した個体数だけ回路グラフをランダムに生成する．次に，新たに生成された個体に対して，環境に対する適応度を評価する．その後，評価値をもとに選択した個体に進化的操作を行い，次世代の個体群となる子孫を生成する．EGG では，進化的操作として,「交叉」と「突然変異」を定義している.「交叉」は，2 つの回

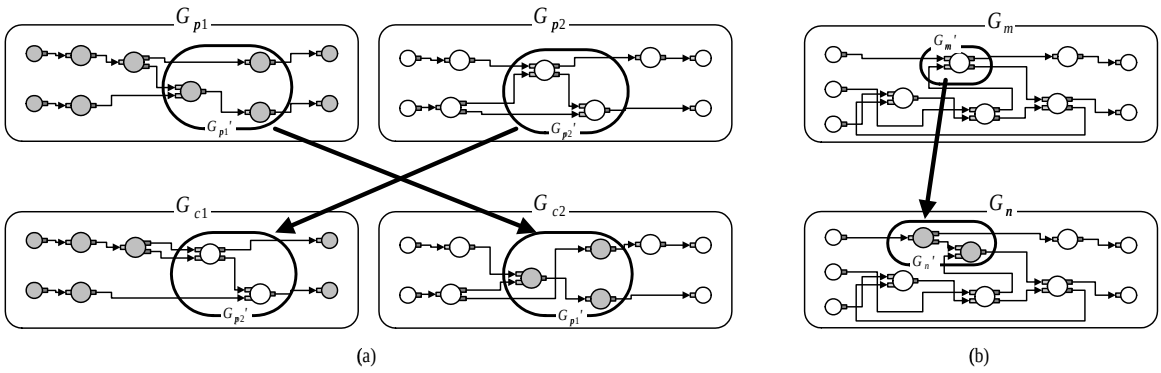

Fig. 3　進化的操作の例: (a) 交叉，(b) 突然変異

路グラフの部分回路グラフを交換して新たな回路グラフを作り出す操作である．一方,「突然変異」は，個体構造の一部をランダムに生成した構造に置き換える操作である．2 つの操作は，完全性を保存しながら回路グラフの構造を変更する．すなわち，操作対象の回路グラフが完全回路グラフであれば，操作後に生成される回路グラフも完全回路グラフとなる．進化的操作の例を Fig. 3に示す．

3.　EGG システムのオブジェクト指向フレームワーク

本章では，高い拡張性を実現するため，オブジェクト指向フレームワークというシステム構成法に基づいて EGG システムを設計する．以下では，まず，オブジェクト指向フレームワークに基づく EGG システム（EGG フレームワーク）を示す．次に，その EGG フレームワークを用いたアプリケーション（EGG アプリケーション）の開発方法について述べる．

3.1　EGG フレームワークの構築

EGG は，基本的なシステムフローや個体の構造操作の方法などはアプリケーション間で共通している．また，アプリケーション特有の変更点はそのアプリケーションが対象とする問題と EGG の接点である個体の表現方法と評価方法などに限定される．そこで EGG システムの共通部分を括り出し，変更されうる部分と切り分けることにより，拡張性の高いシステムを構築することが可能である．一方，オブジェクト指向に基づき高い拡張性を実現するシステム構成法として，オブジェクト指向フレームワーク（以下，フレームワーク）という手法が知られている [4]．フレームワークは，特定の分野を対象としたシステムの基本構造を提供する．アプリケーションの開発者は，フレームワークに用意された変更可能な部分（ホットスポット）をカスタマイズすることで，そのアプリケーションに要求される機能を実装する．フレームワークを構築することにより，フレームワークで定めた基本構造をアプリケーション開発時にそのまま使用可能となり，生産性や保守性の大幅な向上が期待できる．

構築した EGG フレームワークのクラス図を Fig. 4 に示す．クラス図は，UML (Unified Modeling Language) に定義される表記法の 1 つであり，システム内のクラス間の静的構造を示している．1 つの矩形は 1 つのクラスを表しており，クラス間を結ぶ線がそれらの関係を示している．構築した EGG フレームワークは，その役割の点から，EGG のシステムフローに関するクラス群と個体を表現するクラス群に分けられる．

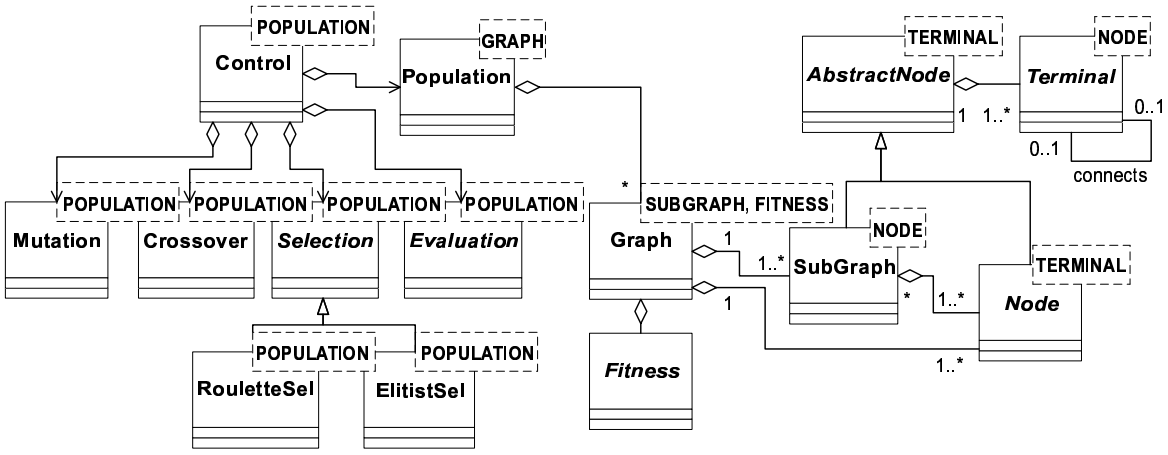

Fig. 4　EGG フレームワークのクラス図

システムフローに関するクラスは，Control，Population，Mutation，Crossover，Selection，Evaluation の 6 つである．Control は Population に対して各種操作を行い，Fig. 2で示した EGG のシステムフローを実行する．Population は，ある世代の個体群を表現し，最良の評価値を持つ個体を取り出す関数や個体群の平均評価値を算出する関数などを定義している．各種操作のアルゴリズムには，Mutation，Crossover，Selection，Evaluation があり，それぞれ突然変異，交叉，選択および評価のアルゴリズムを設定する．Mutation および Crossover には，EGG における標準的な突然変異および交叉のアルゴリズムを設定している．EGG アプリケーションの開発者は，必要であればこれらのクラスを継承し，新たな突然変異および交叉のアルゴリズムを表現するクラスを定義する．また，Selection については，それを継承することにより，一般的な選択法であるルーレット選択法やエリート保存選択法のアルゴリズムを設定する RouletteSel および ElitistSel をあらかじめ定義している．それ以外の選択法を用いる場合，Selection を継承してそのアルゴリズムを定義する．一方，アプリケーションが対象とする問題ごとに個体の評価方法は全く異なるため，Evaluation には，あらかじめ標準的な評価方法は定義しない．アプリケーション開発者は，Evaluation を継承して任意の評価関数を定義できる．

一方，個体を表現するクラスは，Graph，SubGraph，Node，Fitness，Terminal の 5 つである．Graph は，SubGraph，Node および Fitness を保持し，それらの取得に関する基本的な関数を定義している．Node は，ノードを表現し，端子を表現する Terminal を保持する．SubGraph は，部分回路グラフを表現し，Node オブジェクトをポインタで保持する．実際の個体データでは，Graph は SubGraph オブジェクトのリストを一定数だけ保持しておき，構造操作の際にそのリストの中から使用する SubGraph オブジェクトを選びだす．Fitness は，Graph の個体としての評価値を表現し，評価値の算出に必要な各種パラメータや定数などを保持する．また，AbstractNode は，SubGraph と Node の共通した機能を有するクラスとして定義してある．これら個体を表現するクラスは，継承により属性や操作を追加・変更することで，用途に応じた新たな個体を表現する．例えば，Node には，ノードの機能を表す番号を指定することでその機能に応じた入出力端子数でそのノードを初期化する関数をホットスポットとして定義している．ノードの種類は

アプリケーションごとに全く異なるため，この関数はアプリケーションごとに変更される．

3.2 EGG アプリケーション開発例

前節で設計したEGGフレームワークを用いてEGGアプリケーションを開発するには，はじめに，(i) 合成目標となる構造を回路グラフとして表現するために，ノードの種類や端子の属性などを決定する．次に，(ii) 得られる回路グラフの評価方法を決定する．その後，(i),(ii) の決定に従い，EGGフレームワーク中のクラスに新しい属性や操作を追加・変更する．これは，属性や操作の追加・変更の必要なクラスについてのみ，そのクラスを継承して新しいクラスを定義することにより行う．

EGGフレームワークの`Evaluation`を継承する例として，次章で述べる全加算器合成を行うEGGアプリケーションの評価クラス（`AppEvaluation`）をFig. 5に示す．ここで，`AppPopulation`は，全加算器合成のEGGアプリケーションのために既に定義した個体群であるとする．Fig. 4の`Evaluation`のテンプレート引数には`Population`かあるいはそれを継承したクラスが想定されている．そこで，`Evaluation`のテンプレート引数に`AppPopulation`を指定した後，それを継承し，`AppEvaluation`を定義する．この継承の際に`AppEvaluation`には，個体がどれだけ求める回路として評価値が高いかを計算するEvaluate関数の内容を記述する．ここでは，`AppEvaluation`に機能評価値を求めるEvaluateFunctionality関数と性能評価値を求めるEvaluatePerformance関数を追加している．Evaluate関数はこの2つの関数を使って評価値を算出する．

4. EGG フレームワークを用いたアプリケーション開発

本章では，EGGフレームワークを用いて開発したEGGアプリケーションとその実行結果を示す．まず，ゲートレベルによる組み合わせ回路合成への適用例として，全加算器および2×2bit乗算器の合成を行うEGGアプリケーションについて述べる．次に，順序回路合成への適用例として，多入力のビットシリアル加算器の合成を行うEGGアプリケーションについて述べる．

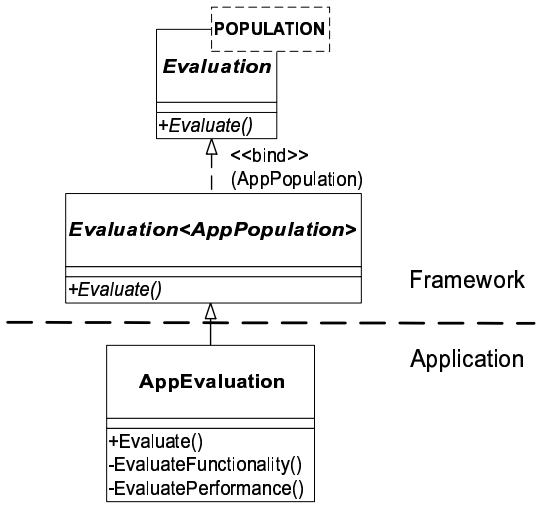

Fig. 5　`Evaluation`クラステンプレートの継承例

Table 1　全加算器合成に使用する機能ノード

Name	Delay	Area
2-input AND	τ	ρ
2-input XOR	τ	ρ
3-input AND	2τ	2ρ
3-input XOR	2τ	2ρ
Constant 1	-	-

4.1 全加算器の合成

Table 1に全加算器合成のために設定した機能ノードを示す．ここでは，FPGA (Field Programmable Gate Array) へのマッピングを想定し，AND-EXOR形の論理関数 [5] を実現する機能ノードを設定した．各ノードの遅延時間と面積の値は2入力1出力のゲートの値をそれぞれ単位とした．

回路グラフの評価値は，本稿で用いたEGGアプリケーション全てに共通して，機能評価値 F と

Table 2　アプリケーションで追加した記述とフレームワークとの記述量の比較

	Num. of files	Num. of lines
Framework	25	3905
Application	13	1316
Total	38	5221

性能評価値 P の和で表される．機能評価値 F は，その個体がどの程度目標とする回路機能を満たしているかを表す．開発したEGGアプリケーションでは，全入力パターンを入力したときに回路の出力が目標とする出力とどの程度合致するかを求める関数として，

$$F = 100 \times \frac{x}{m \cdot 2^n} \tag{1}$$

と定義される．ここで，n, m は，それぞれ合成する回路の入力数と出力数を指す．また，x は個体の出力と目標とする出力が合致した数を指す．全加算器合成の場合は $n = 3, m = 2$ であり，x は0から16の範囲の値をとる．一方，性能評価値 P は，その個体が表す回路の遅延 D と面積 A との積から

$$P = \frac{C}{DA} \tag{2}$$

で与えられる．ここで C には，$P_{\max}/F_{\max}$ が5/100程度となる定数を設定する．

以上のような個体表現および評価方法をプログラムに記述するために，EGGフレームワーク中の必要なクラスを継承した．継承したクラスは，`Graph`，`SubGraph`，`Node`，`Terminal`，`Evaluation`，`Fitness` の6つである．EGGフレームワークのソースファイルの数およびその行数と，このEGGアプリケーションのために新たに記述したソースファイルの数および行数をTable 2に示す．

開発したEGGアプリケーションを用いて，全加算器の合成実験を行った．本実験で使用したEGGの主なパラメータをTable 3に示す．Fig. 6に乱数の初期値を変えて100回試行した結果を示す．

Table 3　合成に用いる主なパラメータ

Population size	100
Max. num. of generations	3000
Crossover rate	0.7
Mutation rate	0.1

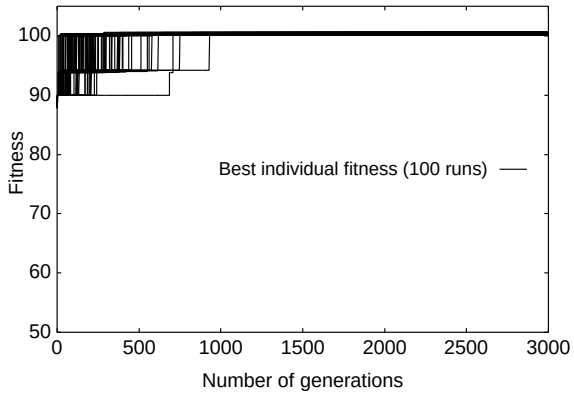

Fig. 6　全加算器合成実験の評価値の推移

ここで，横軸は世代数，縦軸はその世代における最良個体の評価値を示す．実験の結果，100回全ての試行において，1000世代以内に機能を満たす ($F = 100$) 個体が生成された．Fig. 7に実験で生成された最良の回路構造を示す．

4.2　2×2bit 乗算器の合成

前節で開発したEGGアプリケーションの一部を拡張し，2×2bit乗算器の合成実験を行った．2×2bit乗算器の合成は，式(1)における x の値を決定した全加算器の真理値表を2×2bit乗算器のものに置き換えることで実現できる．さらに本実験では，交叉や突然変異で選択される部分回路グラフを連結なものに限定するアルゴリズムを`Graph`から継承するクラスに追加することで，組み合わせ回路の効率よい探索を実現する．

本実験で用いたEGGの主なパラメータは，全加算器合成の場合と同様，Table 3の値を用いた．Fig. 8に乱数の初期値を変えて100回試行した結果を示す．実験の結果，100回の全ての試行で3000

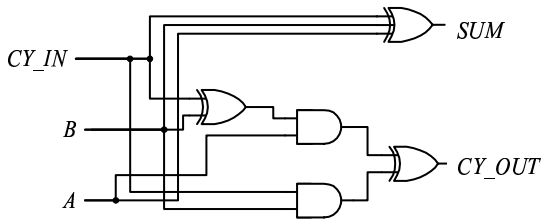

Fig. 7　全加算器合成実験で得られた最良回路

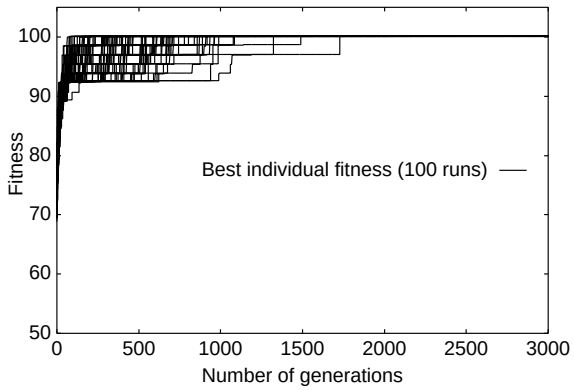

Fig. 8　2×2bit 乗算器合成実験の評価値の推移

世代以内に機能を満たす回路が生成された．Fig. 9 に生成された最良回路を示す．

4.3　ビットシリアル回路の合成

順序回路の機能を完全に検証する場合，レジスタを考慮する必要があるため，膨大なテストパターンを入力する必要がある．このテストパターンの数は入力のビット数に対して指数関数的に増大するため，シミュレーションによる機能の検証は実用的ではない．これに対応するため，開発する EGG アプリケーションには順序回路のための記号検証手法を導入する [2]．これは，回路から導出した連立方程式を解くことにより，算術演算回路の機能を高速に検証する手法である．

使用する機能ノードを Table 4に示すように設定すると，これらのノードから構成されるビットシリアル算術演算の機能は一般的に次式で表せる．

$$K_0 Y = \sum_{i=1}^{n} \hat{K}_i X_i + f(X_1, \cdots, X_n) \qquad (3)$$

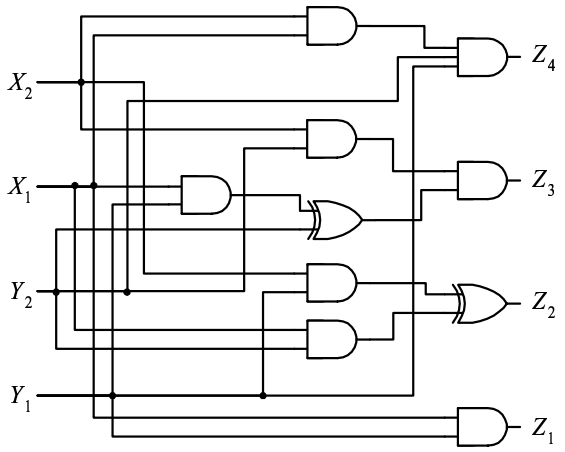

Fig. 9　2×2bit 乗算器合成実験で得られた最良回路

Table 4　ビットシリアル加算器合成に使用する機能ノード

Name	Mathmatical representation	Delay
Full adder	$2C + S = X_1 + X_2 + X_3$	2τ
Half adder	$2C + S = X_1 + X_2$	τ
1-bit register	$Y = 2X$	-
Branch	$Y_1 = X, Y_2 = X$	0

ただし，X_i $(i = 1, \cdots, n)$ はビット列の入力，Y は出力，$\hat{K}_i$ $(i = 0, \cdots, n)$ は非負整数定数，$f(X_1, \cdots, X_n)$ は入力に対する非線形関数である．機能評価値 F は，目標とする回路の係数値 K_i $(i = 0, \cdots, n)$ と式 (3) における係数 $\hat{K}_i$ $(i = 0, \cdots, n)$ との類似度により定まる．一方，性能評価値 P は前節と同様に式 (2) から求める．ここで，D は 2 入力 XOR ゲートを単位遅延としたときのレジスタ間の最大遅延とし，A はモジュール間配線数とする．

以上のような評価方法でビットシリアル回路合成のための EGG アプリケーションを開発した．開発した EGG アプリケーションは，目標とする回路の係数 K_i $(i = 0, \cdots, n)$ の値および n の値を指定することでさまざまなビットシリアル回路合成に適用できる．今回の実験では，$n = 10, K_0 =$

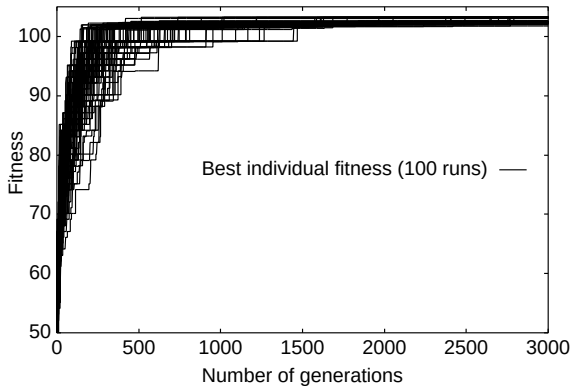

Fig. 10　10 入力ビットシリアル加算器合成実験の評価値の推移

$K_1 = \cdots = K_{10} = 1$ とし，10 入力のビットシリアル加算器の合成実験を行った．実験のパラメータは，Table 3と同じ値を使用した．Fig. 10に乱数の初期値を変えて 100 回試行を行った結果を示す．実験の結果，100 回の試行すべてで 3000 世代以内に機能を完全に満たす ($F = 100$) 個体が生成された．実験で得られた最も良い回路構造を Fig. 11に示す．

5.　おわりに

本稿では，拡張性の高い EGG システムとして，EGG にオブジェクト指向フレームワークの考え方を導入して構築した EGG フレームワークを提案した．本稿で示した全ての EGG アプリケーションは，構築した EGG フレームワークを用いて開発したものである．EGG アプリケーション開発は，アプリケーションに追加する必要のある機能のみをフレームワークから継承したクラスに記述することで行う．本稿では，EGG フレームワークの有効性を示すため，基本的な回路構造の合成のみを行った．今後，さらに複雑な問題に EGG を適用する際には，より多くの拡張を加える必要が生じると予想される．しかし，本 EGG フレームワークは，設計段階で変更可能な部分を明確に切り出しているため，その開発を効率的に行うことが可能

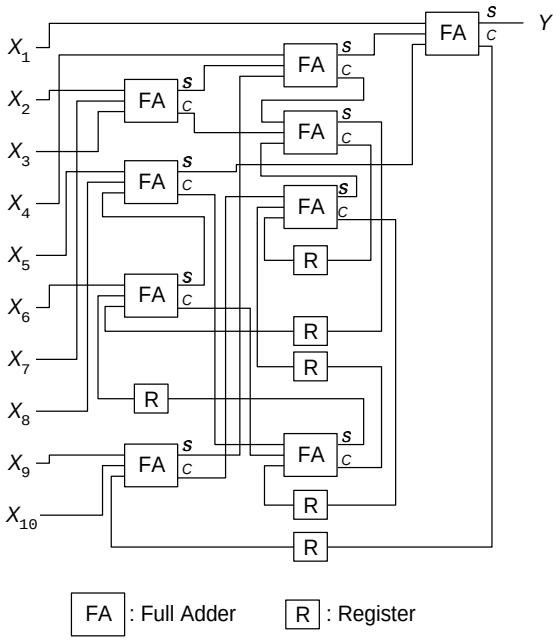

Fig. 11　10 入力ビットシリアル加算器合成実験で得られた最良回路

となる．

参考文献

1) N. Homma, T. Aoki, and T. Higuchi: Evolutionary synthesis of fast constant-coefficient multipliers, *IEICE Trans. Fundamentals*, vol.E83-A, no.9, 1767/1777 (2000)

2) T. Terasaki, T. Aoki, and T. Higuchi: Evolutionary Synthesis of Sequential Arithmetic Circuits, *Proc. of 2000 IEEE International Symposium on Intelligent Signal Processing and Communication Systems*, 1067/1072 (2000)

3) M. Natsui, T. Aoki, and T. Higuchi: Synthesis of Multiple-Valued Arithmetic Circuits Using Evolutionary Graph Generation, *Proc. of The 31th IEEE International Symposium on Multiple-Valued Logic*, 253/258 (2001)

4) 平野広美: 遺伝的アルゴリズムと遺伝的プログラミング, パーソナルメディア (2000)

5) 笹尾 勤: 論理関数の EXOR 論理を用いた表現と回路設計への応用, 電子情報通信学会誌, vol.79, no.2, 147/154 (1996)