Graph-based individual representation for evolutionary synthesis of arithmetic circuits

Naofumi HOMMA

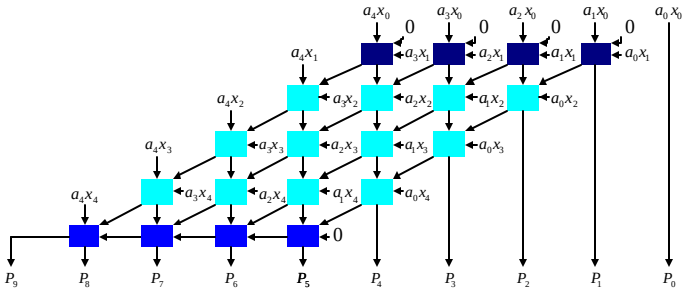
Takafumi AOKI

Tatsuo HIGUCHI

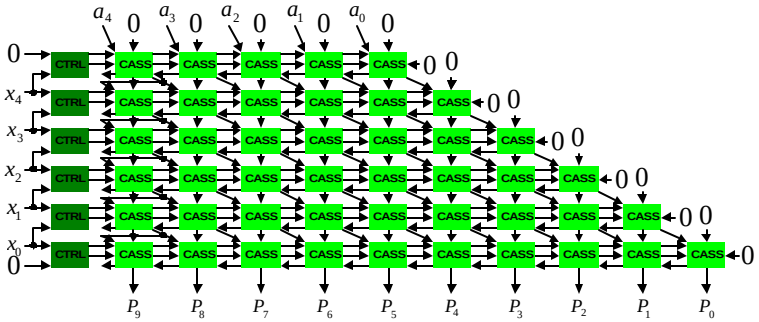
Graduate School of Information Sciences,

Tohoku University, Japan

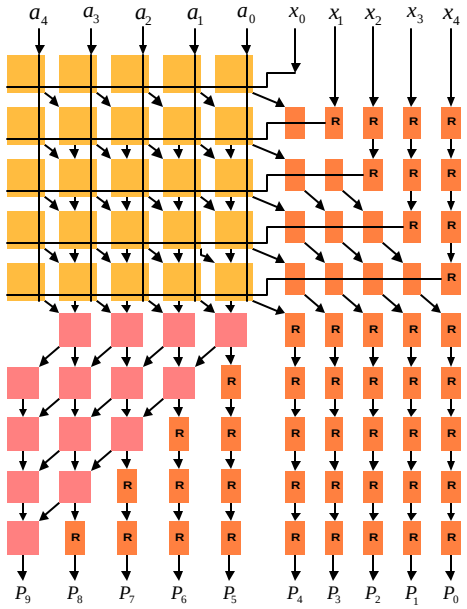
What is arithmetic circuit structure?



Array multiplier for two's complement numbers.



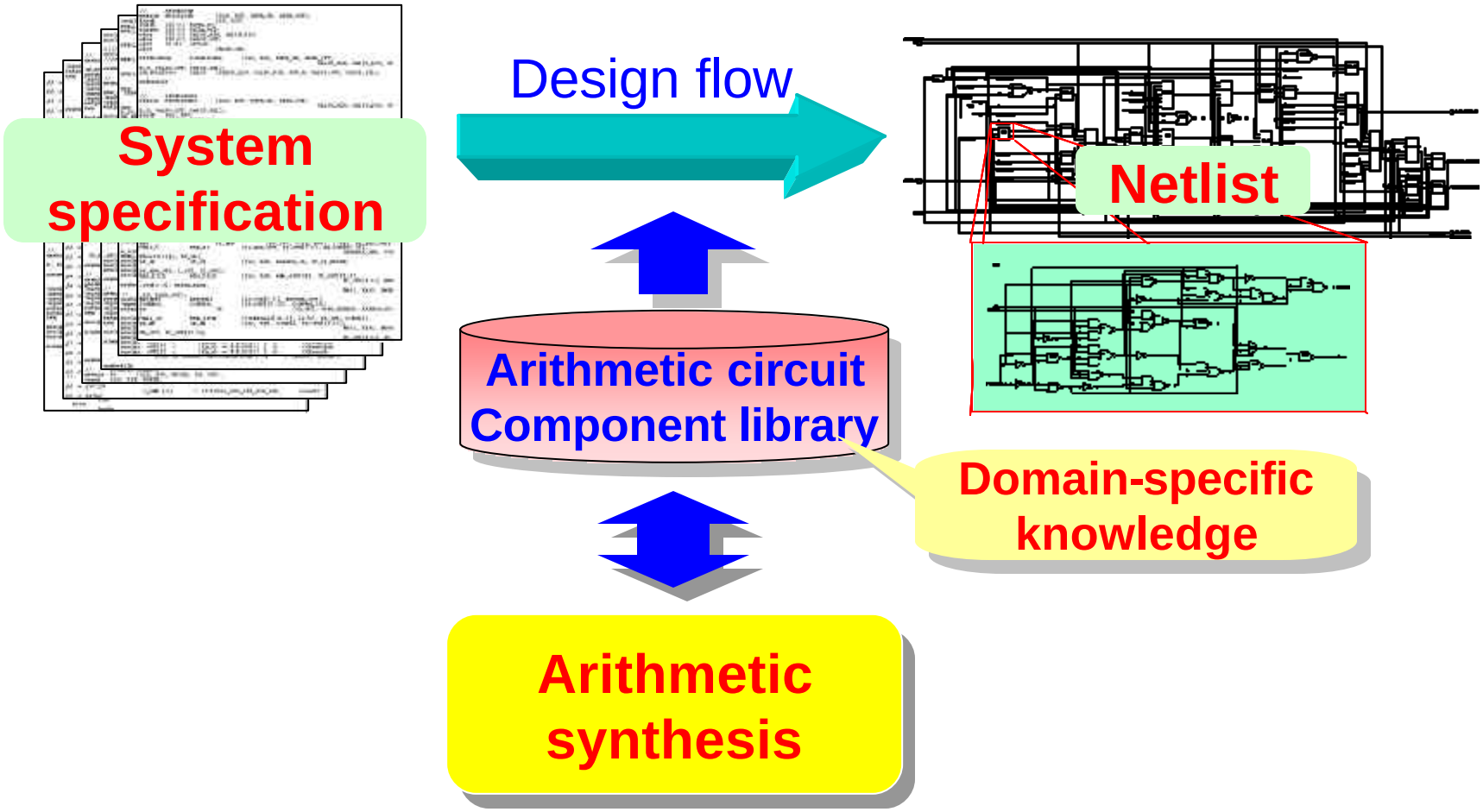
Booth's algorithm array multiplier.



Pipelined 5×5 array multiplier for unsigned numbers.

There is no systematic way of optimizing the circuit structure for a given function.

High-level synthesis flow



Is it possible to synthesize arithmetic circuits?

Graph-based evolutionary optimization technique

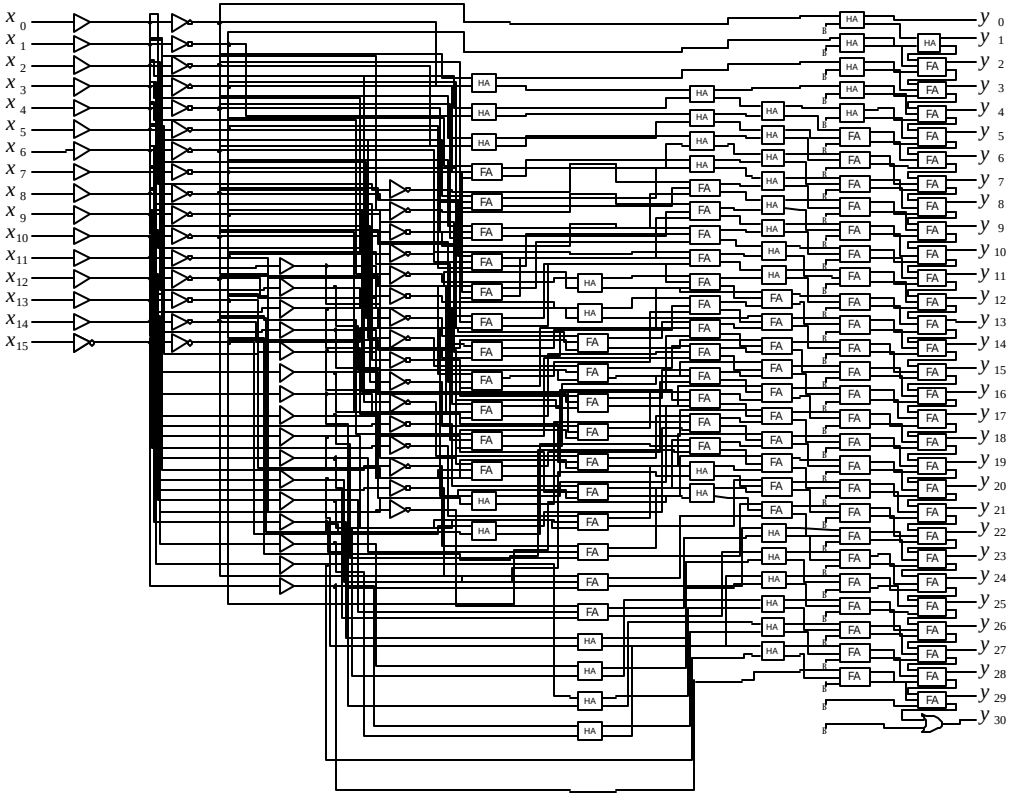
Evolutionary Graph Generation (EGG)

Key ideas of EGG

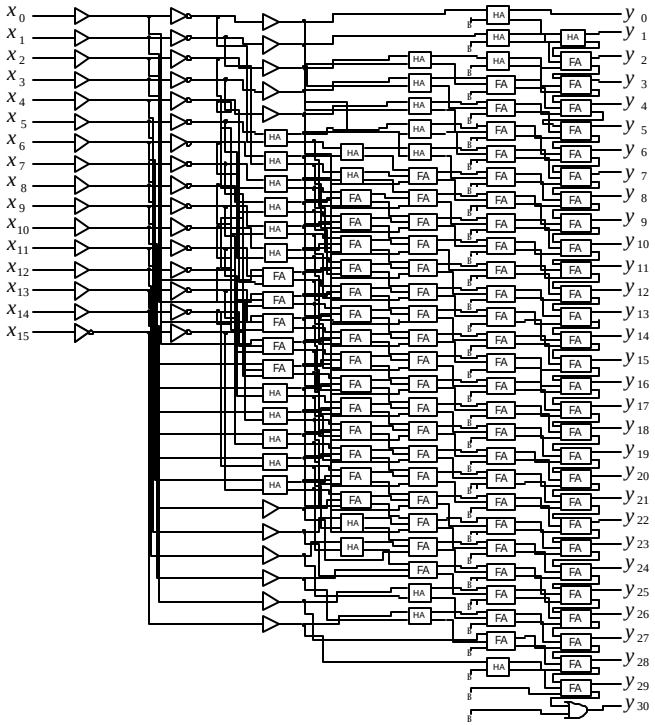
- To employ general graph structures as individuals
  - Search space reduction by graph theoretic constraints
  - Graph-based abstraction of arithmetic algorithms
- To introduce subgraph-based evolutionary operations
  - Hypothesis: every arithmetic circuit is a collection of many useful sub-circuits, and the total functionality of a circuit emerges from a collective behavior of its sub-circuits.

Constant-coefficient multiplier synthesis

Ex. Y=10075X (16-bit precision)



Conventional best multiplier
(CSD encoding and Wallace tree)



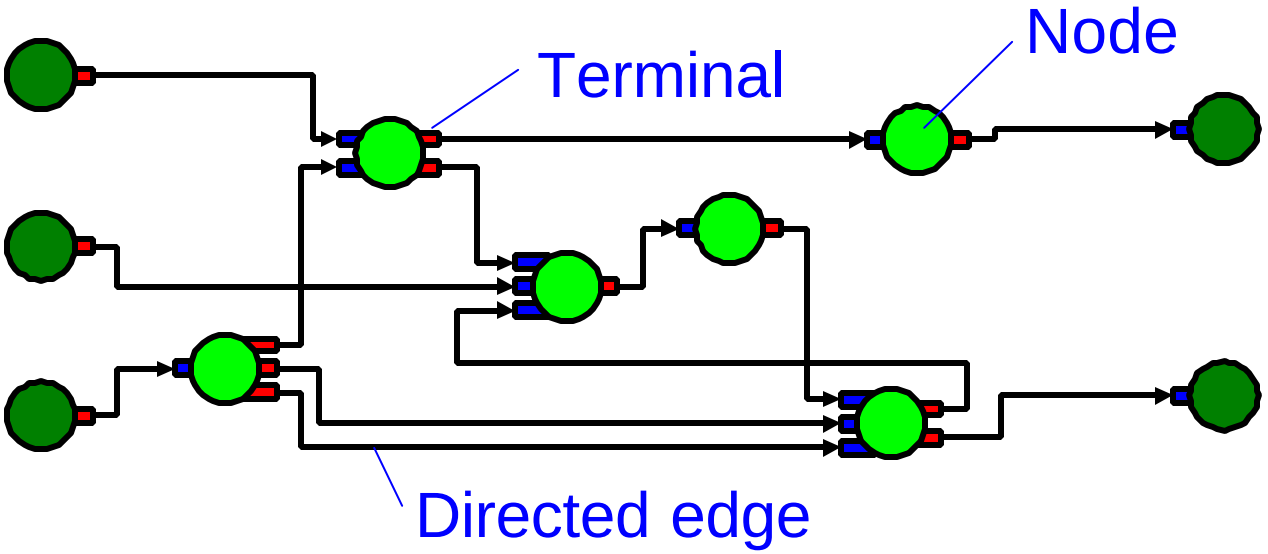
Solution generated
by the EGG system

Outline

- Backgrounds
- Basic concept of Evolutionary Graph Generation (EGG)
- System implementation for constant-coefficient multiplier synthesis
- Experimental result
- Conclusion and prospects

Graph-based individual representation

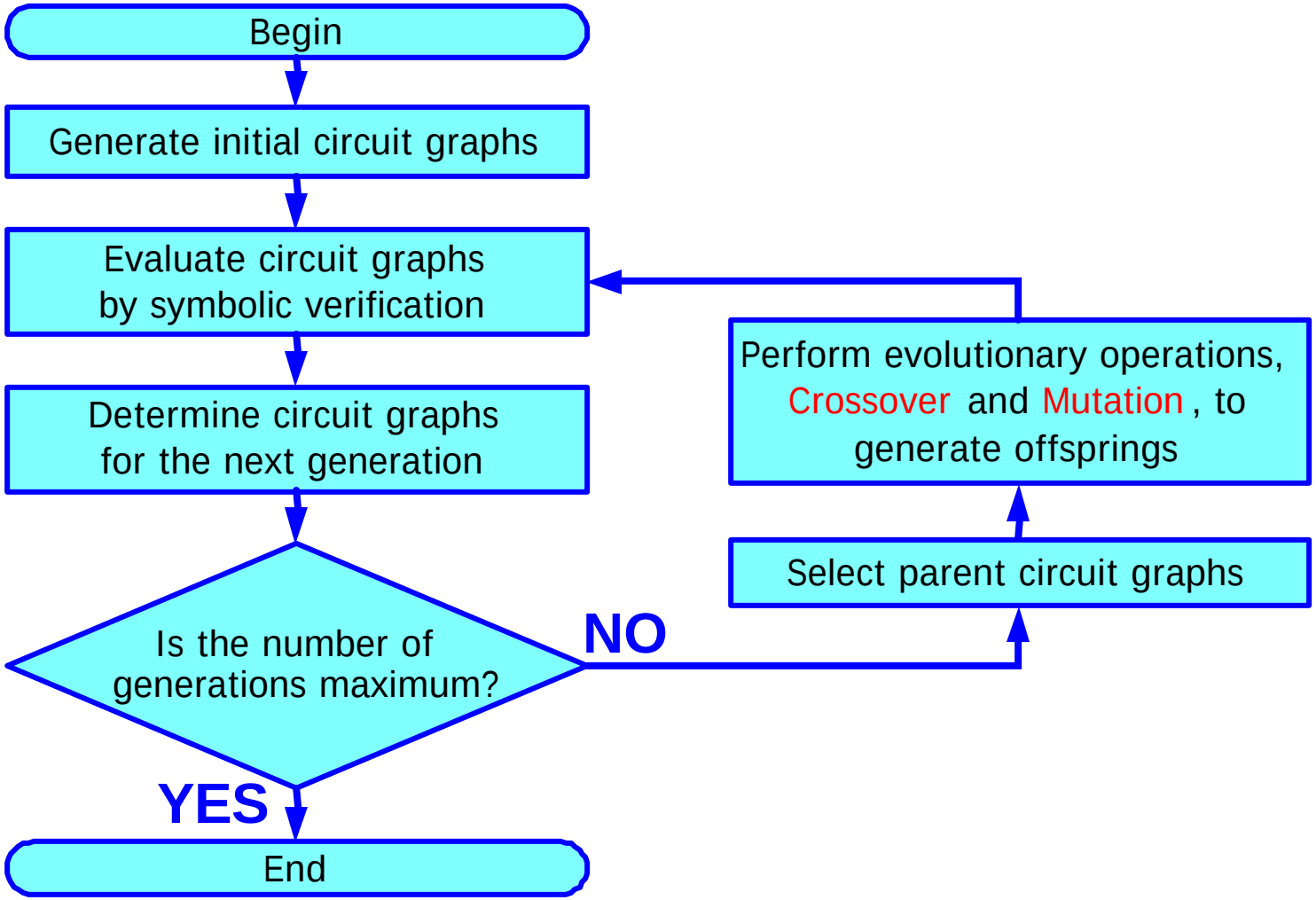
Circuit Graph $G = \left(N^G, T_O^G, T_I^G, n_O^G, n_I^G, e^G\right)$



No unconnected terminal = **Complete circuit graph**

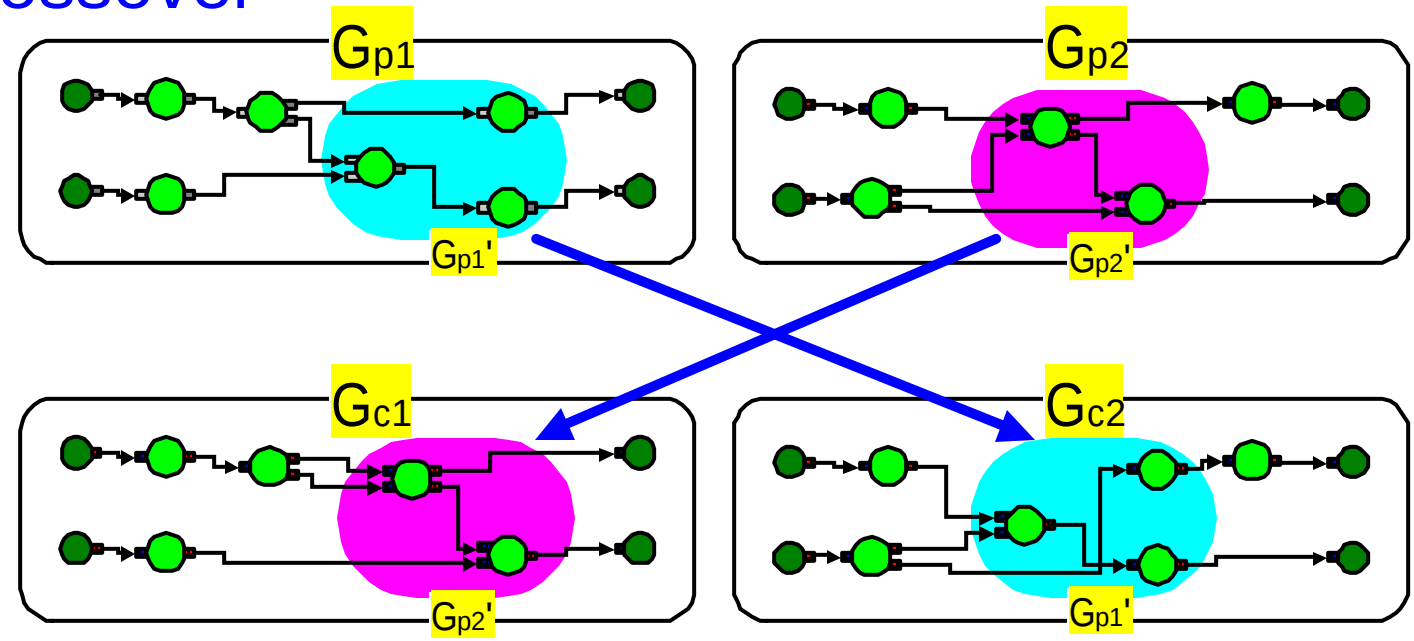
To guarantee valid circuit structures, the EGG system generates only complete circuit graphs.

EGG system flow

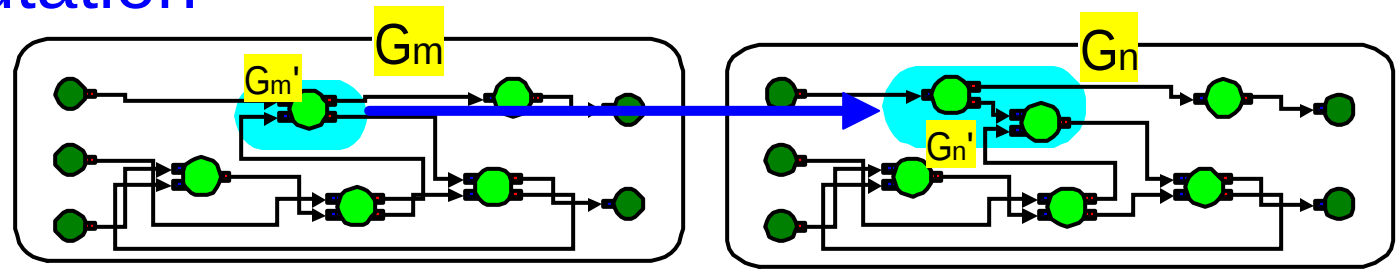


Evolutionary Operations

Crossover



Mutation



Constant-coefficient multiplication: $Y=RX$

Why constant-coefficient multipliers?

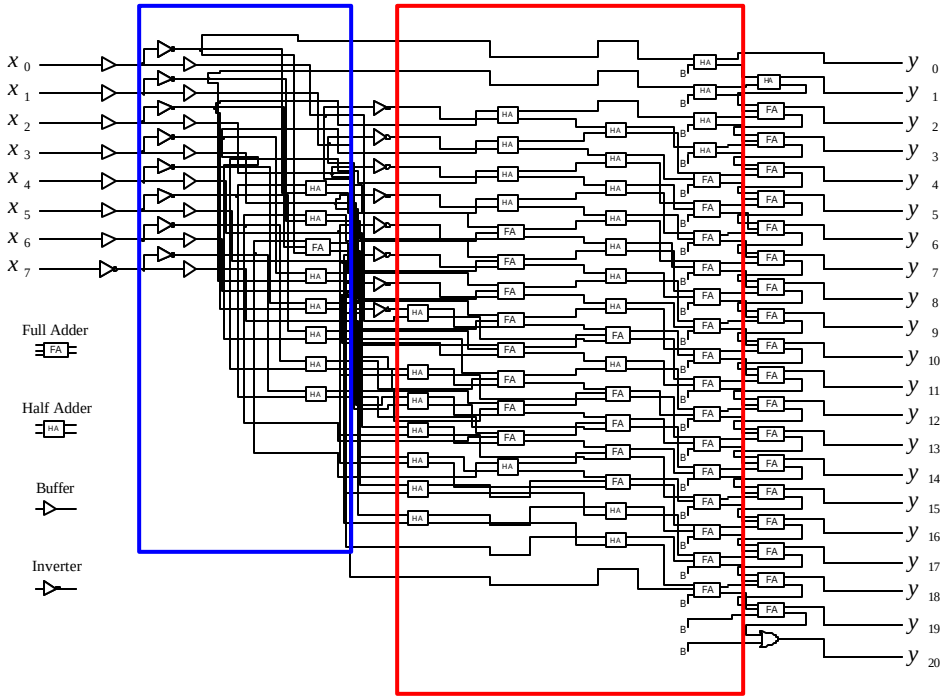
- There are many possible choices for the multiplier structure for a specific coefficient.
- The complexity of the multiplier structure significantly varies with the coefficient R.
- There is no systematic way of predicting the best structure even if we have the knowledge of fundamental arithmetic components.

Combinational constant-coefficient multipliers

- Partial product generation: CSD encoding

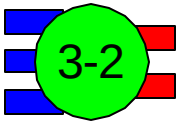
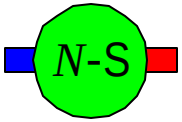
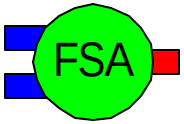
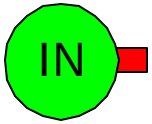
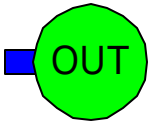
$$124_{10} = 01111011_2 = 10000\bar{1}00_{CSD}$$

- Partial product accumulation: Wallace tree



Design specification

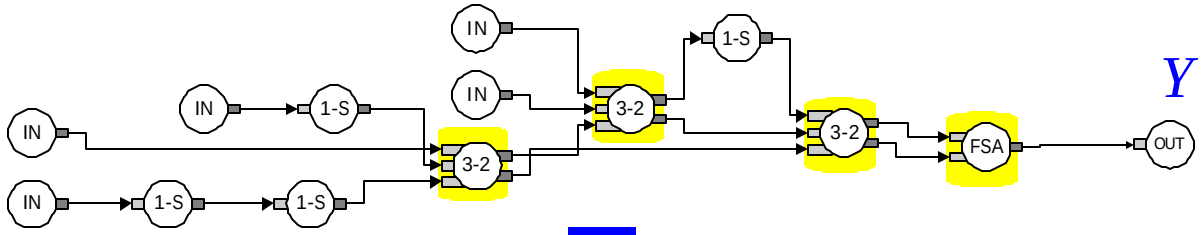
- Table of nodes used in the EGG system

Name	Symbol	Function
3-2 counter		3-input 2-output carry-free addition with output branches
N-bit shifter		*N*-bit arithmetic shifter (N:1,2,4)
Final-stage adder		Carry-propagate addition with a bias canceling stage
Operand input		Input signal
Operand output		Output signal

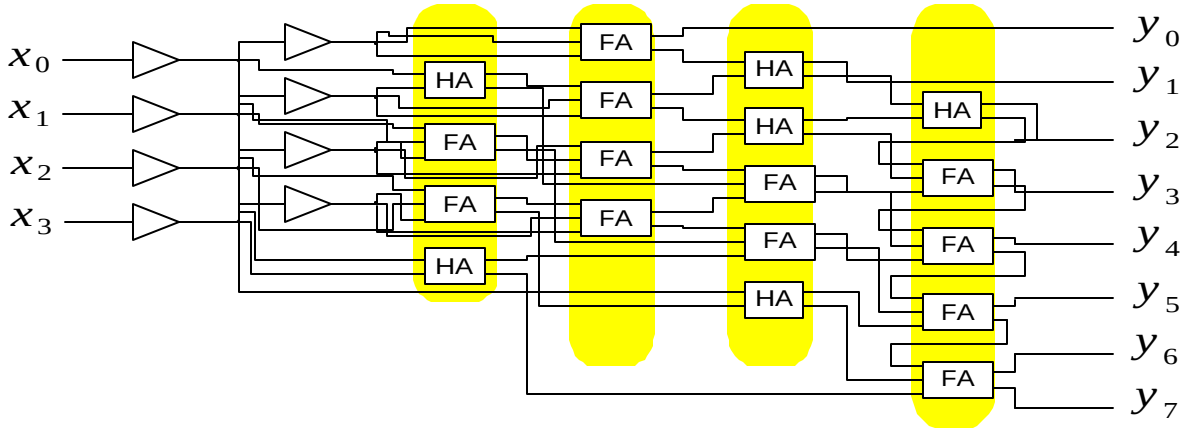
- FSA node is directly connected with OUT node.
- Individuals: Circuit graphs without any feedback loops

Individual circuit graph

Complete circuit graph without any feedback loops (directed acyclic graphs)



$$Y = \hat{R}X + f(X)$$



Ex., 4-bit unsigned binary data

Evaluation

$$Fitness = F + P \quad (P_{\max} / F_{\max} \cong 5/100).$$

Functionality measure (F)

Functionality measure evaluates validity of the logical function compared with the target coefficient.

$\hat{R}$: Coefficient obtained from circuit graph

Correlations

R: Target coefficient

Performance measure (P)

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

D: Number of counter stages

A: Number of inter-module interconnection

C: Constant adjusted to keep the ratio of P to F.

Experimental condition

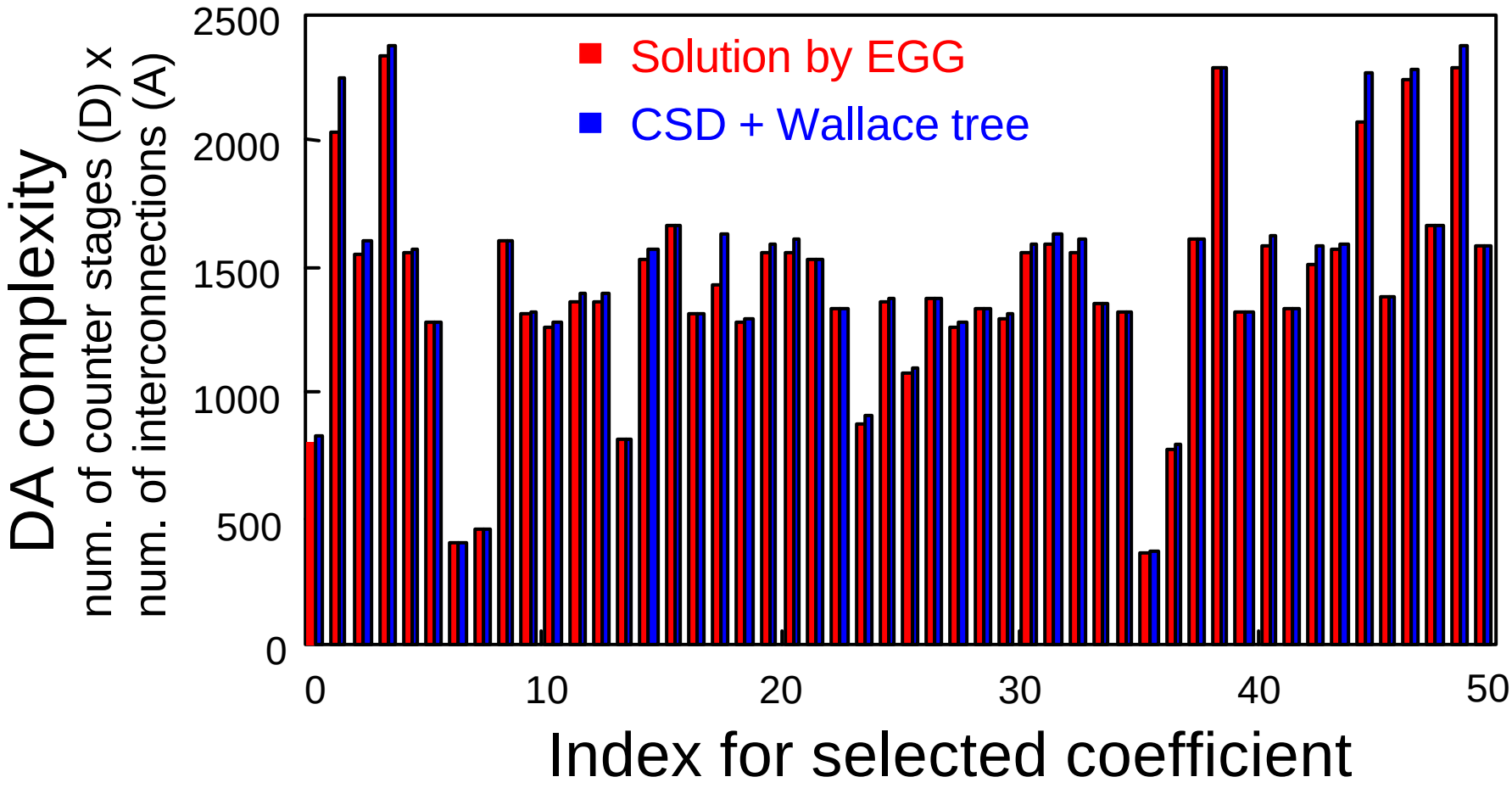
- Generate 50 distinct multipliers whose coefficients are selected randomly in the range of -32768 to 32767.

- Compare the solutions generated by the EGG system and the corresponding CSD multipliers using Wallace tree architecture.

Main parameter values

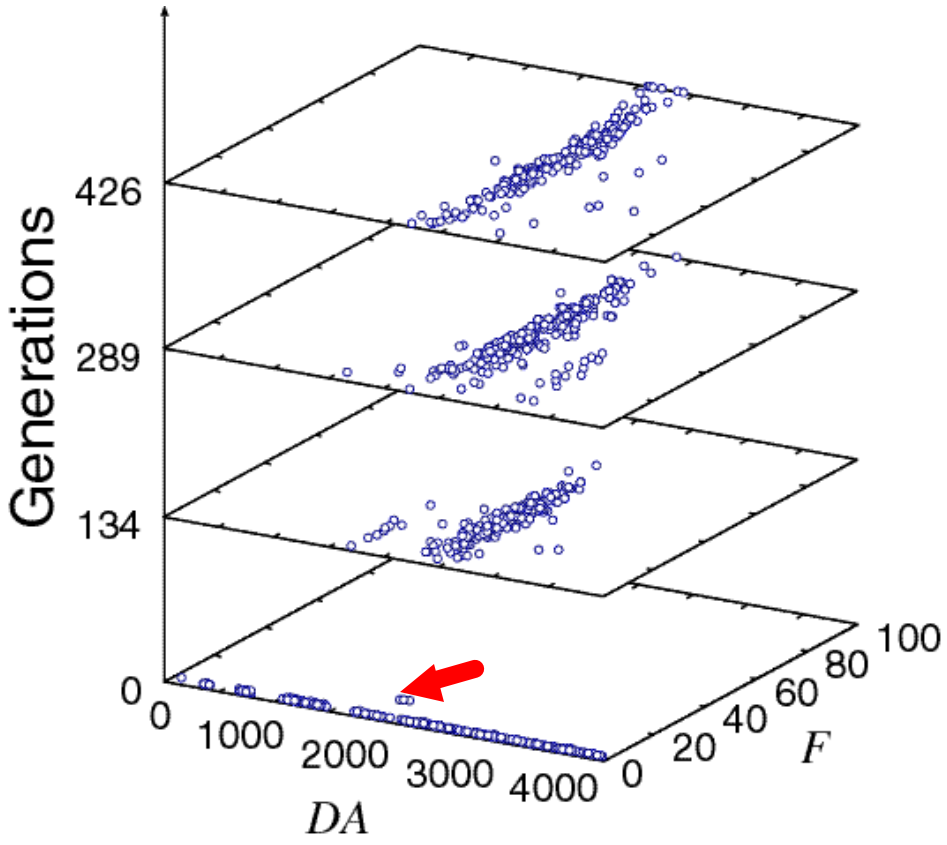
Population size	500	Crossover rate	0.7
Max. num. of generations	500	Mutation rate	0.2
Max. num. of nodes	50	Operand word-length	16

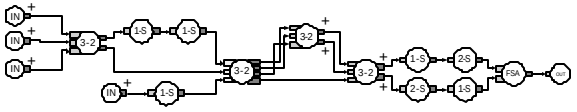
Experimental results

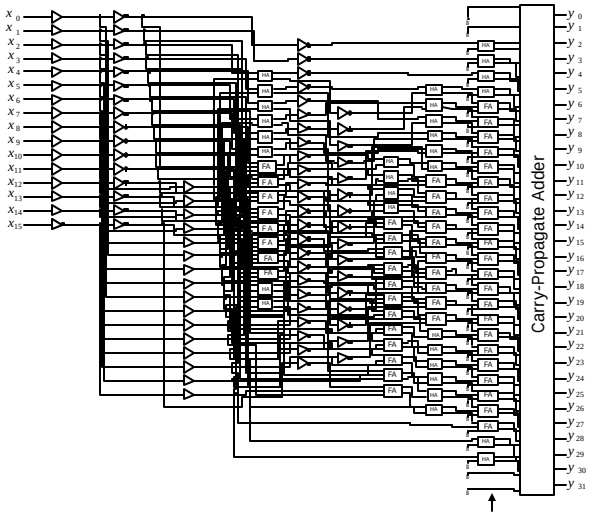


Example of evolution process

Ex. Y=26204X



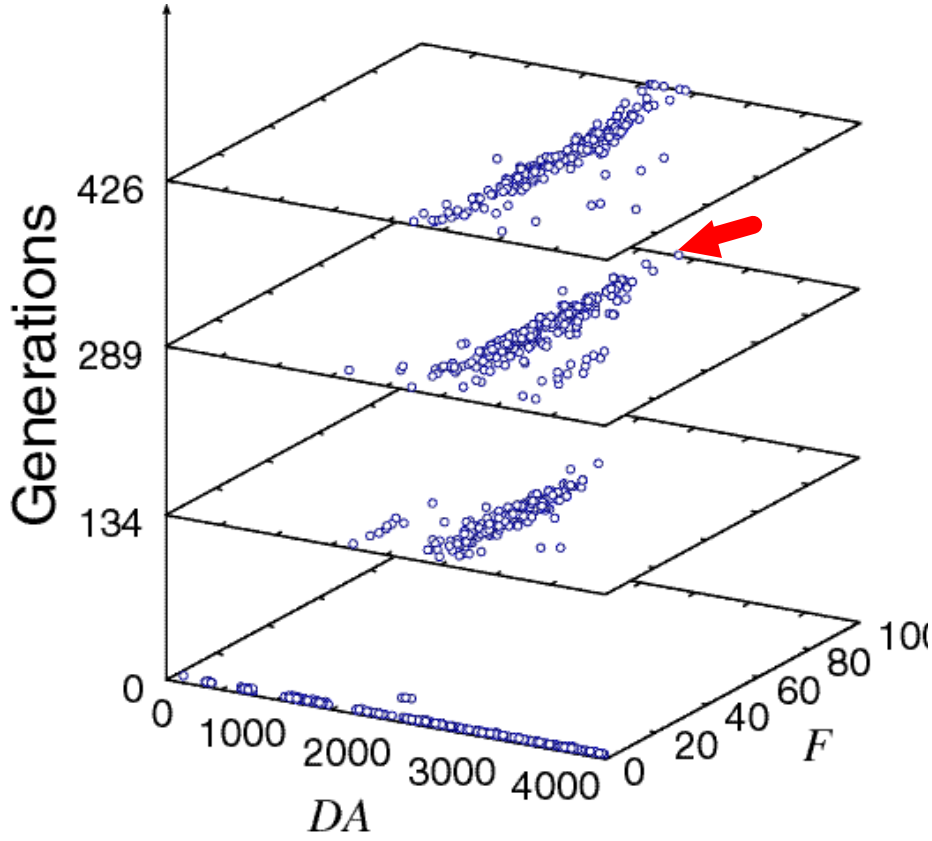


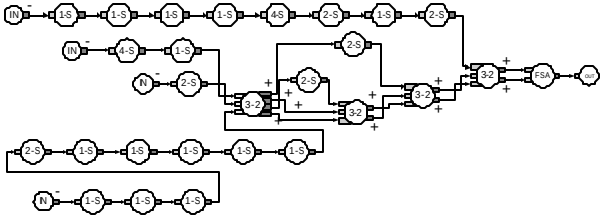


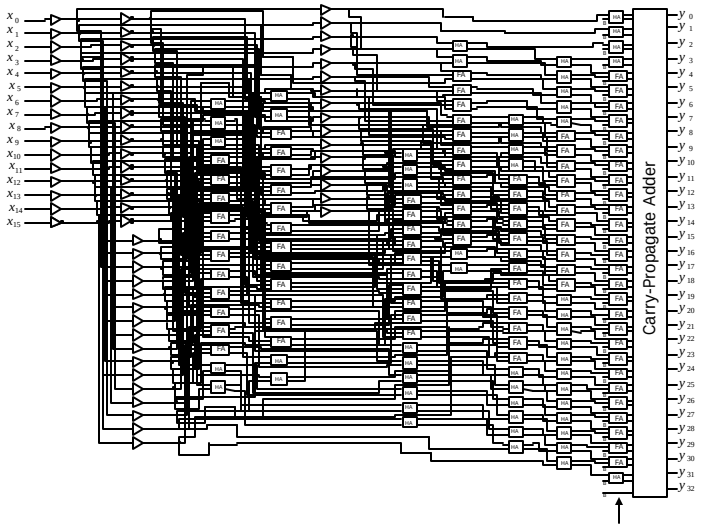
F = 17 ***DA*** = 1985

Example of evolution process

Ex. Y=26204X



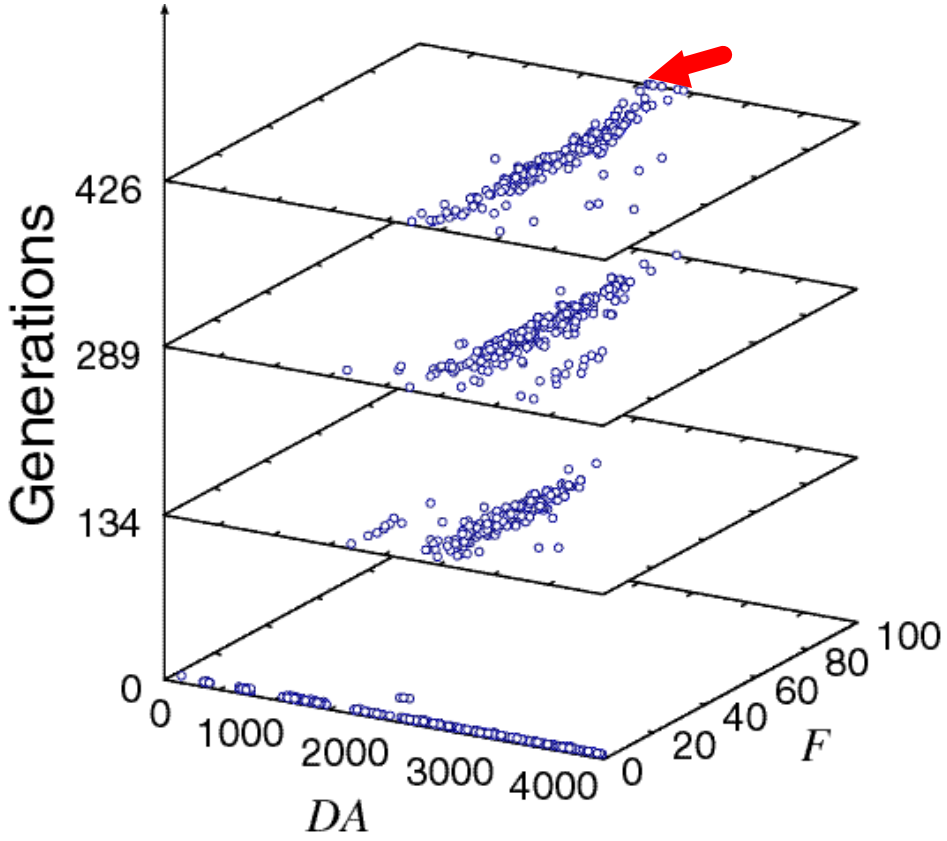


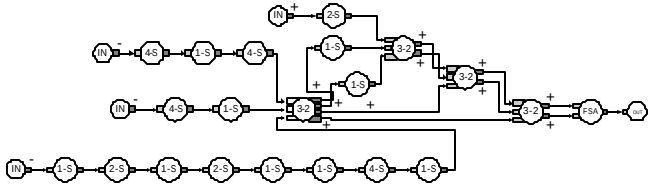


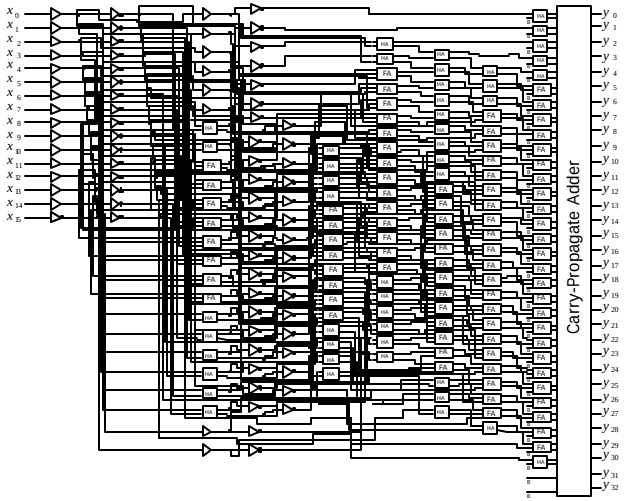
F = 100 ***DA*** = 2345

Example of evolution process

Ex. Y=26204X

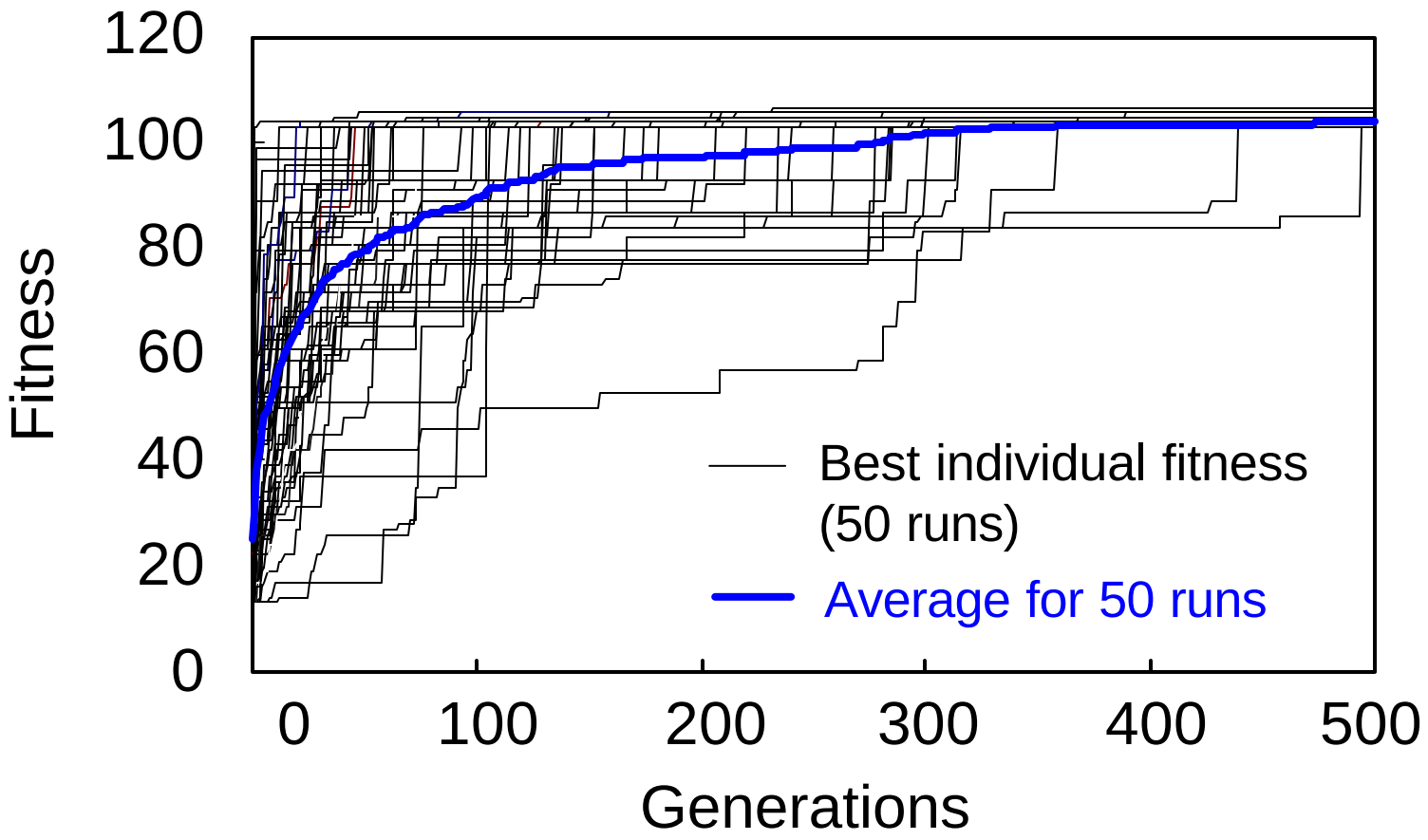






F = 100 ***DA*** = 2070

Transition of the best individual fitness



Average computation time required to obtain the solution is about 3.6 hours on a Linux PC with 700 MHz Pentium III.

Conclusion

Arithmetic circuit synthesis using Evolutionary Graph Generation (EGG)

Key ideas of EGG

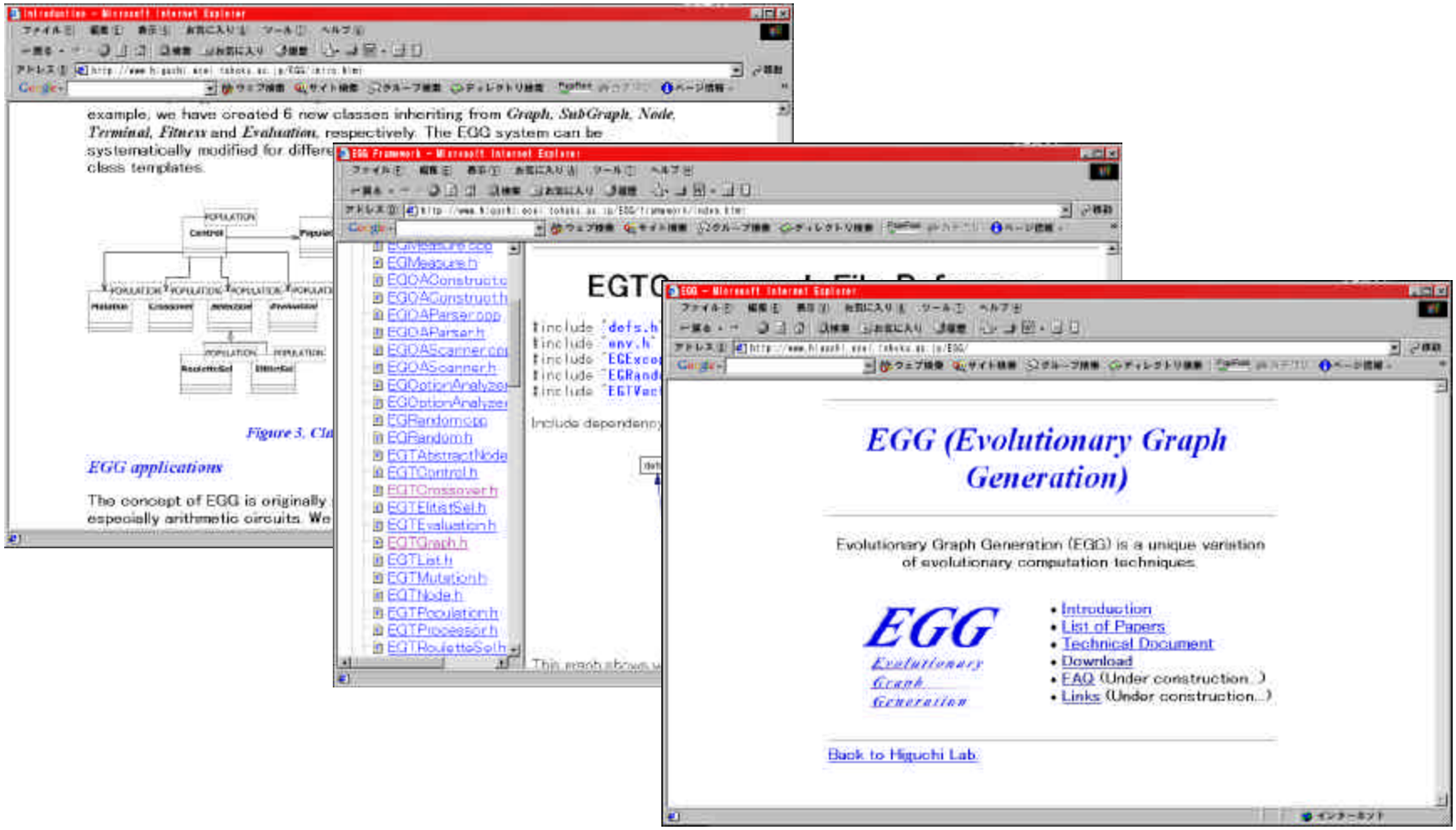
- To employ general graph structures as individuals
- To introduce subgraph-based evolutionary operations

Ex., constant-coefficient multiplier synthesis

- All the solutions generated by the EGG system are comparable with the conventional best multipliers.

URL: http://www.higuchi.ecei.tohoku.ac.jp/egg/

EGG framework



URL: http://www.higuchi.ecei.tohoku.ac.jp/egg/