Evolutionary algorithm for state assignment of finite state machines

Mariusz Chyży
Polish-Japanese Institute of Information Technology, Research Center
Centrum Badawcze PJWSTK, ul. Koszykowa 86, 02-008 Warszawa
e-mail: chyzym@pwstk.edu.pl

Witold Kosiński
Polish-Japanese Institute of Information Technology, Research Center
Centrum Badawcze PJWSTK, ul. Koszykowa 86, 02-008 Warszawa
e-mail:wkos@pjwstk.edu.pl

Introduction

A significant part of digital circuits is constituted by sequential synchronous circuits behaviour of which can be presented by a finite state machine (FSM). So nothing strange the FSM synthesis methods are continually developed (cf. the monographs [14, 19] or [17, 8]). One of the most crucial steps in FSM synthesis is the encoding of FSM states referred to as the state assignment problem (SAP). It consists in the unique assignment of bit strings to the states of sequential circuit (SC). This step of FSM synthesis is important because it affects the quality of realised SC (cost/area, maximum frequency, power consumption).

Effective algorithms for the state encoding were developed, e.g. NOVA [18] for two-level implementation targeted to Programmable Logic Arrays (PLAs) or MUSTANG [10] and JEDI [15] for multilevel FSM implementation. However, state assignments generated by these methods, for FSMs implemented in modern programmable devices [1, 7] allowing efficient implementations of digital systems, are far from optimum [9]. Taking above into account, and considering other conditions (cf. [5, 12, 16]) we decided to try to cope with SAP using evolutionary algorithm (EA).

Genetic and evolutionary algorithms are successfully used in VLSI CAD [11]. They were also applied to SAP [3, 4, 5]. In this paper we propose an evolutionary algorithm for SAP. We introduce the original crossover operators and next compare them with the known ones using.

State assignment problem

States of FSM are named but when a sequential circuit is implemented they are represented by bit strings. Therefore during FSM synthesis states have to be encoded i.e. uniquely assigned with binary strings (state codes). Assume FSM has m states from the set $S = \{s_1, s_2, ..., s_m\}$. The minimum number of bits that must be used to encode FSM states is $r_{min} = \lceil \log_2 m \rceil$, where $\lceil d \rceil$ is the smallest integer not less than d. The number A of possible assignments is

$$A = C_{2^r}^m \cdot m! = \frac{2^r!}{(2^r - m)!} \tag{1}$$

where $C_{2^r}^m$ is the number of $k-$element combinations without repetitions from n elements. Naturally, when good state assignment is searched for all values of r from the range $r_{min}, ..., m$ then the number A of possible assignments is

$$A = \sum_{r=r_{min}}^{m} C_{2^r}^m m! = \sum_{r=r_{min}}^{m} \frac{2^r!}{(2^r - m)!} \ . \tag{2}$$

However, the search space can be reduced when equivalence classes of state assignments are taken into account [19]. The space of all possible solutions of the SAP is huge, and the problem is NP-hard [8, 19]. Moreover, the quality of solutions strongly depends on the architecture of the target device in which FSM is to be implemented. When we add that the form of the optimisation function is difficult to define, the SAP seems to be a suitable field for application of EAs and evolutionary techniques [4, 3, 5].

Here we focus our attention on crossover operators (along with genotypes). In classical genetic algorithms a chromosome is represented by a binary string. For SAP the individual could be represented by a string of $m \cdot r$ bits, where each r consecutive bits constitute the code of the subsequent FSM state. For example, when $m = 5$ and $r = 3$, the chromosome 111011101001110 assigns strings $111, 011, 101, 001, 110$ to the states $s_1, s_2, s_3, s_4, s_5, s_6$, respectively.

Proposed evolutionary algorithm

In proposed EA for SAP chromosome is represented by a string of integers: i-th number constitutes a code of i-th FSM state s_i, e.g. an individual 73516 represents assignment of $7_{(10)}3_{(10)}5_{(10)}1_{(10)}6_{(10)}$ to the states s_1, s_2, s_3, s_4, s_5, respectively. Assume $U(i)$ and $W(i)$ (abbreviated as u_i and w_i, respectively) denote an allele (gene value) at the i-th position of chromosome U and W, respectively, l - chromosome length (number of chromosome genes/positions, here $l = m$) and $L(U)$ - the set of positions (loci) of chromosome U. Let
$C(U,W) = \{i : u_i \neq w_j,\ j = 1, 2, ...l\}$,
$D(U,W) = \{i : u_i = w_j, j = 1, 2, ...l, i \neq j\}$,
$E(U,W) = \{i : u_i = w_i, i = 1, 2, ...l\}$.
These three sets are mutually disjoint and $L(U) = C(U,W) \cup D(U,W) \cup E(U,W)$. It should be noticed that $|C(U,W)| = |C(W,U)|, |D(U,W)| = |D(W,U)|, E(U,W) = E(W,U)$, where $|A|$ is the cardinality of the set A. Analogously we define $C(W,U), D(W,U)$ and $E(W,U)$ with the same properties.

Assume F is a set of loci of a chromosome, $F \subset L$. Let $G(F,U,k,h) = \{f \in F : 0 < H(k,U(f)) \leq h\}$, where $H(i,j)$-the Hamming distance between numbers i and j.

Using that notation two crossover operations $M1$ and $M2$, and order-based mutation are defined. We chose the rank-based selection [20] and applied *elitist strategy* (n from the best chromosomes of the current population are unconditionally selected to the new one).

Experimental results

The effectiveness of the proposed crossover operators has been

checked with the use of several MCNC benchmark FSMs [21]. FSMs used in tests were synthesized by Altera Multiple Array MatriX Programmable Logic User System (MAX+PLUS II) in the device of MAX9000 family [2] (actually it was the EPM9320RC208-15 device). EA (program has been written in C/C++) called the MAX+PLUS II (further on referred to as MPII) to ascertain the fitness of chromosomes.

Based on experimental results it can be seen that EA is an effective algorithm for FSM state assignment, competitive - regarding generated results - to the commercially available state-of-the-art software containing a lot of domain knowledge.Although EA has searched only for state assignments that yield cheaper FSM implementation, obtained results, generally show also an increase in the FSM maximum (clock-to-clock) frequency. The main disadvantage of the EA is its execution time. In our experiments one EA run took from several hours up to 29 hours (this time cost was mainly because of calling the external software - here MPII - to evaluate generated solutions), but it must be added that experiments were done simply using personal computer. In our implementation the EA worked sequentially. EAs are parallel from their nature so parallel implementation of the EA shall radically decrease the run time.

On the other hand the experimental results confirm the high stability of EAs. It should be noted that EA parameters were chosen arbitrary. Moreover, the proposed M crossovers can be fine-tuned. We believe the better results can be obtained after EA and its operators parameters are adjusted. Proposed crossover operators $M1$ and $M2$ were designed for SAP. Since this problem belongs to the wider class of combinatorial optimisation problems (COPs) than commonly known travelling salesman problem (which can be formulated as a special case of SAP; of course fitness functions and schemata are different for these problems), hence operators proposed in this paper can be applicable to other COPs.

References

[1] www.altera.com

[2] Altera Digital Library 2001, Ver. 2.

[3] Amaral J.N. et al.: *Designing Genetic Algorithms for the State Assignment Problem*, IEEE Trans. on Sys., Man and Cybernetics, **25**, (4), (1995), 687-694.

[4] Almaini A. E. A. et al. *State assignment of finite state machines using a genetic algorithm*, IEE Proc.-Comput. Digit. Tech., 1995, **142**, (4), pp. 279-286.

[5] Chattopadhyay S., Chaudhuri P. Pal, *Genetic Algorithm Based Approach for Integrated State Assignment and Flipflop Selection in Finite State Machine Synthesis*, Proceedings of the IEEE International Conference on VLSI Design 1998, IEEE Comp. Society, Los Alamitos, CA, USA, pp. 522-527.

[6] Chyży M., Kosiński W., *Genetic Algorithm for the State Assignment Problem*, Communications of the 10th International Symposium Intelligent Information Systems, Zakopane, Poland, 17-21 June 2001, (2001), pp. 7-11.

[7] www.cypress.com

[8] De Micheli G., *Synthesis and Optimization of Digital Circuits*, McGraw-Hill , 1994.

[9] Deniziak S., Sapiecha K., *A Comparison of State Encoding Algorithms for Different Reprogrammable Architectures*, Proc. of the RUC Conference., Szczecin, Poland 1998 (in Polish).

[10] Devadas S., Ma H.-K., Newton R., Sangiovanni-Vincentelli A., *MUSTANG: State Assignment of Finite State Machines Targeting Multilevel Logic Implementations*, IEEE Transactions on Computer-Aided Design, **7**, (12), (1988), pp. 1290-1300.

[11] Drechsler R., *Evolutionary Algorithms for VLSI CAD*, Kluwer Academic Publishers May 1998, 196 pp.

[12] Gaat A., *A system of data collection for small satellite systems with interfaces built with the use of FPGAs*, Proc. of the RUC Conference, Szczecin, Poland 1999, pp.301-308 (in Polish).

[13] Grefenstette J.J., Gopal R., Rosmaita B., Van Gucht D., *Genetic Algorithm for the TSP*, Proceedings of the First International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Hillsdale, NJ, 1985, pp. 160-168.

[14] Kam T., Villa T., Brayton R., Sangiovanni-Vincentelli A., *Synthesis of Finite State Machines: Functional Optimization*, Kluwer Academic Publishers, Boston/London/Dordrecht 1998.

[15] Lin B., Newton R., *Synthesis of Multiple Level Logic from Symbolic High-Level Description Languages*, Proc. of the Int. Conf. on VLSI, München, 1989, pp. 187-206.

[16] Michalewicz Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin Heidelberg 1996, 3rd edition.

[17] Perkowski M., *Digital Design Automation: Finite State Machine Design*, http://www.ee.pdx.edu/ mperkows/ =FSM/finite-sm/finite-sm.html.

[18] Villa T., Sangiovanni-Vincentelli A., *NOVA: State Assignment of Finite State Machines for Optimal Two-Level Logic Implementation*, IEEE Transactions on Computer-Aided Design, 1990, **9**, (9), pp. 905-924.

[19] Villa T., Kam T., Brayton R., Sangiovanni-Vincentelli A., *Synthesis of Finite State Machines: Logic Optimization*, Kluwer Academic Publishers, Boston/London/Dordrecht 1998.

[20] Whitley D., *The GENITOR Algorithm and Selective Pressure: Why Rank-Based Allocation of reproductive Trials is Best*, Proceedings of the 3rd International Conference on Genetic Algorithms, D. Schaffer, ed., pp. 116-121, Morgan Kaufmann, 1989.

[21] Yang S., *Logic Synthesis and Optimization Benchmarks User Guide*, Microelectronics Center of North Carolina, Research Triangle Park, North Carolina 1991.