ЭВОЛЮЦИОННЫЙ АЛГОРИТМ КОДИРОВАНИЯ СОСТОЯНИЙ КОНЕЧНЫХ АВТОМАТОВ

Мариуш Чижи

Польско-японский институт информационных технологий, Исследовательский центр

Витольд Косински

Польско-японский институт информационных технологий, Исследовательский центр

Перевод с английского:

Евгений Татолов

Донецкий национальный технический университет

Введение

Значительная часть цифровых систем содержит синхронные последовательностные схемы, поведение которых может быть представлено конечными автоматами (FSM – Finite State Machine). Поэтому нет ничего странного в том, что методы синтеза конечных автоматов постоянно развиваются (монографии [14, 19] или [17, 8]). Одним из наиболее критичных шагов при синтезе конечного автомата является кодирование внутренних состояний (SAP – State Assignment Problem), задача которого состоит в присваивании уникальных битовых строк состояниям конечного автомата. Этот шаг процесса синтеза является важным, поскольку влияет на качество реализуемой последовательностной схемы (стоимость/площадь, максимальная частота, расход мощности).

Были разработаны эффективные алгоритмы кодирования состояний, например NOVA [18] для двухуровневых реализаций конечных автоматов на программируемых логических матрицах (PLAs – Programmable Logic Arrays) или MUSTANG [10] и JEDI [15] для многоуровневых реализаций. Однако, кодирование состояний, генерируемое этими методами для конечных автоматов, которые реализуются на современных программируемых устройствах [1, 7] (позволяющих эффективно имплементировать цифровые

системы), далеко от оптимального [9]. Приняв вышесказанное во внимание и рассмотрев другие условия [5, 12, 16], мы решили предпринять попытку решения SAP, используя эволюционный алгоритм (EA – Evolutionary Algorithm).

Генетические и эволюционные алгоритмы успешно используются в САПР СБИС [11] и применялись к SAP [3, 4, 5]. В данной работе мы предлагаем эволюционный алгоритм для решения SAP и представляем оригинальные операторы скрещивания, сравнивая их с уже известными.

Проблема кодирования состояний

Состояния конечного автомата имеют символические обозначения, но при реализации последовательностной схемы они представляются битовыми строками. Следовательно, при синтезе конечного автомата, состояния должны кодироваться, то есть им должны быть присвоены уникальные битовые строки (коды состояний). Примем, что конечный автомат имеет m состояний из множества $S = \{s_1, s_2, \dots, s_m\}$. Минимальное количество битов, которое может быть использовано для кодирования состояний, составляет $r_{min} = |\log_2 m]$, где $|d]$ – наименьшее целое число, не меньшее, чем d. Количество A возможных вариантов кодирования состояний равно:

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

где C_k^n – количество k-элементных комбинаций без повторений из n элементов. Очевидно, когда выполняется поиск эффективного кодирования состояний для всех значений r из диапазона $\overline{r_{min}, m}$, то количество A возможных вариантов кодирования состояний вычисляется следующим образом:

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

Однако, пространство поиска может быть уменьшено, если принимается во внимание эквивалентность классов кодирований состояний [19].

Пространство всех возможных решений SAP велико, и проблема является NP-полной [8, 19]. Более того, качество решений сильно зависит от архитектуры целевого устройства, на котором реализуется конечный автомат. Если добавить, что сложно определить форму оптимизационной функции, то SAP представляется подходящим полем деятельности для применения эволюционных алгоритмов [4, 3, 5].

В данной работе мы акцентируем внимание на операторах скрещивания (наряду с генотипами). В классических генетических алгоритмах, хромосома представляется битовой строкой. Для SAP, особь может быть представлена строкой из $m \cdot r$ битов, где каждая последовательность r битов является кодом соответствующего состояния конечного автомата. Например, если $m = 5$ и $r = 3$, хромосома 111011101001110 присваивает строки 111, 011, 101, 001, 110 состояниям s_1, s_2, s_3, s_4, s_5 соответственно.

Предлагаемый эволюционный алгоритм

В предлагаемом EA для решения SAP, хромосома представляется строкой целых чисел: i-ое число является кодом i-го состояния конечного автомата s_i, например особь 73516 представляет собой коды $7_{(10)}3_{(10)}5_{(10)}1_{(10)}6_{(10)}$ состояний s_1, s_2, s_3, s_4, s_5 соответственно. Примем, что $U(i)$ и $W(i)$ (обозначаемые в дальнейшем как u_i и w_i) указывают на аллель (значение гена) в i-ой позиции хромосом U и W; l – длина хромосомы (количество генов хромосомы/позиции, в нашем случае $l = m$) и $L(U)$ – множество позиций (локусов) хромосомы U. Пусть

$C(U, W) = \{i: u_i \neq w_j, j = 1,2, \dots, l\}$,

$D(U, W) = \{i: u_i = w_j, j = 1,2, \dots, l, i \neq j\}$,

$E(U, W) = \{i: u_i = w_i, i = 1,2, \dots, l\}$.

Три этих множества попарно не пересекаются и $L(U) = C(U, W) \cup D(U, W) \cup$

$\cup E(U,W)$. Отметим, что $|C(U,W)| = |C(W,U)|$, $|D(U,W)| = |D(W,U)|$, $|E(U,W)| = |E(W,U)|$, где $|A|$ – мощность множества A. Аналогично мы определяем $C(W,U), D(W,U)$ и $E(W,U)$ с теми же свойствами.

Примем, что F – множество локусов хромосомы, $F \subset L$. Пусть $G(F,U,k,h) = \{f \in F: 0 < H(k,U(f)) \leq h\}$, где $H(i,j)$ – расстояние по Хэммингу между числами i и j.

С использованием этой нотации определены две операции скрещивания $M1$ и $M2$ и порядковая мутация. Мы выбрали ранговую селекцию [20] и применили элитарную стратегию (n из лучших хромосом текущей популяции безусловно выбираются в новую популяцию).

Экспериментальные результаты

Эффективность предложенных операций скрещивания была проверена с использованием нескольких тестовых конечных автоматов MCNC [21]. Используемые конечные автоматы были синтезированы с помощью Altera Multiple Array MatriX Programmable Logic User System (MAX+PLUS II) для устройств семейства MAX9000 [2] (а именно, для EPM9320RC208-15). Программа на C/C++, реализующая EA, вызывает MAX+PLUS II (в дальнейшем называемую MPII) для установления пригодности хромосом.

Основываясь на экспериментальных результатах, можно заметить, что EA является эффективным алгоритмом для кодирования состояний конечного автомата, конкурентоспособным – в соответствии с полученными результатами – с доступными современными коммерческими программами, содержащими много отраслевых знаний. Хотя EA выполняет только кодирование состояний, что удешевляет реализацию конечного автомата, полученные результаты показывают также возрастание максимальной частоты конечного автомата. Основным недостатком EA является время его выполнения. В наших экспериментах, один EA работал от нескольких до 29 часов (такие затраты времени связаны, в основном, с вызовом внешнего программного обеспечения

– MPII – для оценки сгенерированных решений), но необходимо добавить, что эти эксперименты выполнялись на персональном компьютере. В нашей реализации EA работает последовательно, а эволюционные алгоритмы являются параллельными по своей сути, так что параллельная реализация значительно уменьшит время выполнения.

С другой стороны, результаты экспериментов подтвердили высокую стабильность EA. Отметим, что параметры EA выбирались произвольно. Более того, предложенные M скрещиваний могут быть доработаны. На наш взгляд, лучшие результаты могут быть получены после того, как параметры EA и их операторов будут отрегулированы. Предложенные операторы скрещивания $M1$ и $M2$ были разработаны для SAP. Поскольку SAP относится к более широкому классу задач комбинаторной оптимизации (COPs – Combinatorial Optimization Problems), как широко известная задача коммивояжера (которая может быть сформулирована как специальный случай SAP; конечно, функция пригодности и схемы для этих проблем различны), следовательно, предложенные в данной работе операторы могут быть применимы для других COPs.

Ссылки

[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*, IEEE 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] Chyzy M., Kosinski 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, M¨unchen, 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.