State assignment techniques — short review

Aleksander Ślusarczyk

The task of the (near)optimal FSM state encoding can be generally formulated as assignment of such binary vectors to the state symbols that the resulting binary next-state and output functions can be efficiently implemented with a given implementation technology. It is therefore clear why the evolution of state encoding algorithms is intimately related to the evolution of the implementation platforms.

The first attempts at algorithms of state assignment for computer program implementation date back to 1960s, when the implementation technology were diodes, transistors and simple gates. Following the methods of human designers, formulated by Humphrey [13] as the "rules for *state code adjacency*", the algorithms proposed in [3][2][10] are based on grouping together 1's in the Karnaugh tables of the resulting binary output and next-state functions, and in result on maximizing the size and minimizing the number of product terms in the sum-of-product expressions describing the combinational part of the FSM realization.

Even though the technology changed, the observations formulated by Humphrey became foundation for the next generations of algorithms.

This was also the case with the family of algorithms based on *symbolic minimization*. This novel idea implemented by de Micheli in KISS [8] in 1985 consists in performing the logic minimization phase *before* the state encoding. The minimization phase takes advantage of the well defined realization cost for PLA technology, which may be closely estimated by the number of inputs, outputs and product terms in minimized cover of the functions. The minimized functions could be realized in PLA technology at the cost (in terms of circuit area) determined prior to encoding, assuming that the constructed code will satisfy all encoding constraints resulting from the symbolic minimization phase. The state encoding problem is in this way reduced to satisfaction of the so called *face* or *input* constraints expressing certain relations between the codes of some groups of states.

The method implemented in KISS has however several shortcomings. First of all, in the symbolic minimization phase the binary next-state functions are assumed to have disjoint on-sets. Thus, the next-state function minimization is not taken into account. This aspect is especially apparent for counter FSMs, where minimized symbolic cover for $2p$-state counter has 2^p product terms, while the optimum is $O(p^2)$. Moreover, the FSM's feedback is not properly taken into account. Also the constraint satisfaction algorithm is just a simple-minded, greedy search.

The idea corresponding to symbolic minimization (minimization of generalized implicants) was even earlier pursued in [5]. However, the computational complexity of the presented (admittedly, much more sophisticated) method made it impractical for FSMs larger that 8 states.

In 1988, Jóźwiak published the method of maximal adjacencies, MAXAD, targeting two-level logic implementations (PLA and random logic AND-OR circuits) [14][15][17]. Although the method uses the "state code adjacency" concept, it differs considerably from the previous methods based on this idea. Jóźwiak considers many sorts of adjacencies, performs a sophisticated adjacency analysis and uses the results of the analysis very effectively in a sophisticated code construction performed in the framework of an effective and efficient double-beam search algorithm. This resulted in a state assignment tool that efficiently produced much better results than any other method published at that time. Compared to KISS, the machines encoded by MAXAD have realizations with on average 13% smaller PLA area and

28% smaller feedback area.

Some of the shortcomings of KISS are addressed in its successor — NOVA [28]. NOVA takes more efficient and flexible approach to constraints satisfaction, representing it as a graph embedding problem and solving in several, heuristic strategies producing superior results and offering quality/runtime trade-offs. Also the output encoding problem is considered here, however in a marginal manner, subordinate to the KISS-like input encoding. The best strategy of NOVA – *iohybrid* – produced results of comparable quality to the results of the maximal adjacency method.

The output encoding problem is addressed in two *dichotomy-based* methods, representing an interesting direction in the scope of the symbolic minimization based methods. Dichotomy-based algorithms take column-based approach to code construction, i.e. values of a particular binary state variable for all states (columns of encoding) of the code are directly decided, instead of the complete code for a particular state (row of encoding). Dichotomies are used for the implicit representation of the binary state variable values. Dichotomies are two-block partitions on certain sub-sets of a given set S, and in the case of state encoding, on some sub-sets of the FSM's states set. The first dichotomy-based method, DUET [7], introduces uniform framework for representation of input and output constraints as dichotomies. The problem of the encoding constraint satisfaction is transformed in this way to the problem of the compatible dichotomies merging. The resulting merged dichotomies define unambiguously certain encoding, which satisfies all the constraints represented by the component dichotomies. The second method in this group, DISA [24], introduces the concept of the dynamic constraint satisfaction. In the process of iterative code construction, the constraint set is modified at each stage, considering the partial information about the output constraints extracted from the current partial encoding. The constraints are then satisfied at a newly created code bit, which introduces the next-generation partial encoding.

In general, the above symbolic minimization-based methods only differ from each other in the level of detail in constraint consideration and in constraint satisfaction method. All of them assume two-level PLA implementation of the functions resulting from the state encoding. This assumption drives the symbolic minimization, which aims at generation of minimum-cardinality symbolic cover – a reasonably good approximation of the optimal FSM realization, especially for input-dominated FSMs. Unfortunately, no such good approximation has been discovered for any sort of multi-level logic (except for multiplexor networks). Multi-level implementations (be it levels of logic gates, PLAs, PALs or CLBs) pose a whole new group of challenges. The greatest potential of the multi-level realizations – the possibility of realizing multiple trade-offs (between area, delay, power, interconnect complexity, etc.) introduces a new kind of more complicated criteria into the process of sequential synthesis, and hence greatly complicates the process. It also makes the definition of the quality (cost) function very difficult. Moreover, each particular implementation platform (complex gates, CPLDs, FPGAs) necessitates certain platform-specific technology mapping, which further separates the state encoding from the actual physical realization, and hinders quality estimation of the constructed code. In particular, the forementioned implementation platforms (LUT-FPGAs, CPLDs, and complex gates) do not impose any constraints on a function type that can be implemented with a single logic building block. Instead, they impose hard constraints on the circuit structural parameters, such as the maximal number of inputs and outputs of a logic block, or the interconnection structure. For these reasons, most encoding methods for the multi-level implementations settle for crude predictions of the 1operations of subsequent logic synthesis steps, and attempt to create such an encoding, which produces output functions "easy" for a certain multi-level combinational logic minimizer.

One of the earliest multi-level state assigment method, MUSTANG [9], falls into this category. Designed to work with MIS logic synthesis system, it attempts to maximize the number and size of the common (sub-)cubes in expressions describing the output functions. These common (sub-)cubes will make it easier for MIS to realize its objective of minimizing the number of literals. The common cube maximization is realized in the process of adjacent code assignment to some selected pairs of states. The selection of the pairs, interestingly, is guided by rules similar to those of Humphrey. Two other well know methods, Jedi [23] and Muse [11], closely follow the concepts implemented in Mustang, with additional improvements introduced by elements of simulated annealing and consideration of Boolean (as opposed to algebraic) operations in common-cube extraction.

A different approach is taken in Mis-MV [21]. It follows in the footsteps of Kiss and applies symbolic minimization to the multi-level realizations. Mis-MV is actually not a state encoding method, but a multi-level logic minimizer able to handle multi-valued variables. It is therefore able to minimize the combinational component of FSM before the actual state encoding, when the state variable is still in its symbolic, multivalued form. Symbolically minimized function is *then* encoded with a simple algorithm based on simulated annealing, guided by the number of literals as its cost function.

Some of the interesting alternative approaches to state encoding include genetic algorithms. Methods presented in [1] and [6] fall into this category. Both methods, to calculate the quality of a chromosome, encode the machine with the code represented by the chromosome and minimize the resulting cover with Espresso. The assignment quality is represented by the size of the PLA implementation of the minimized cover. Unfortunately, the basic genetic algorithms are known to poorly handle the problems with complicated and time consuming quality function [19]. It seems to be the case also in the state encoding application. The run-times of the genetic algorithms exceed in some cases those of the classical algorithms by a factor of 100 on small benchmark examples. An interesting direction pointed in [6] is, however, the simultaneous encoding and selection of the types of flip-flops used to store state variables (D or J/K type). In some cases, the choice of J/K flip-flops (which are known to require less complicated excitation functions) reduced the combinational component of the FSM realization by as much as 80%.

The challenges introduced by the multi-level implementations were further deepened with the advent of FPGA devices. An FPGA is composed of a configurable network of configurable logic blocks (CLBs) capable of realizing (in particular, in lookup table based architectures) *any* function of a fixed number of variables. This characteristic invalidates estimation of the implementation cost of the function by the number of terms or literals. Till now, very little has been done in the field of sequential synthesis targeting FPGA implementations. Notable exceptions are the programs Lax [26] and Minisup [22].

Lax is dedicated to multiplexor-based FPGA (e.g. Act1) architectures. Exploring the direct correspondence between the function's BDD and its multiplexor realization structure, the algorithm attempts to minimize the ROBDD size of the resulting binary functions. It performs iterative improvement of a Mustang-generated encoding by introducing in it some random changes with decreasing level of disturbance.

Yet another approach is represented by Minisup. As a dichotomy-based method, Minisup uses a column-based approach. Aiming at minimization of the total input support of the resulting binary functions, it starts with an initial, long seed encoding and constructs the final encoding by merging the seed code columns. The process is guided by the heuristic that merging of two columns corresponding to two state variables present in the input support of a certain function will reduce the function's input support. The candidates for merging are the pairs of code variables, which appear together in the input supports of the largest number of the output functions.

A totally different view at the state encoding problem is presented in the algebraic structure theory of sequential machines due to Hartmanis and Stearns [12]. The theory utilizes the concepts of partitions and set systems to model the FSM's information structure, and trace dependencies between the information about the FSM's states, inputs and outputs. Observation of these dependencies gives hints related to the direction, in which the encoding of the states should follow to reduce the dependecies of the next-state and output functions from the state and input variables. In this way, some possibly interesting decompositions of the FSM can be discovered and the input supports of the binary next-state and output functions can be reduced. This approach seems to be especially interesting in the case of LUT-FPGAs, where reducing dependencies satisfies two important goals of reducing the functions' input supports and interconnections. Due to its computational difficulty the theory was however receiving little interest until it was further developed and generalized by Jóźwiak in [20][16][18], and used as a base for general decomposition of sequential machines and discrete functions and state assignment of sequential machines. In [18], a relation between the partitions, set systems or covers used to model the algebraic information structure and information about symbols is explicitly stated. The information is defined there as the ability to distinguish between the symbols (in the case of FSM: states, input or output values). Partitions, set systems and covers enable modeling of information in various points of discrete systems. The apparatus of information relationships and relationship measures enables analysis and measurement of the modeled information and relationships between information in the various points. In this way, the apparatus proposed by Jóźwiak makes operational the theory of partitions and set systems of Hartmanis and Stearns. It enables construction of effective and efficient decomposition and encoding algorithms based on this theory and its extensions.

Some applications of the algebraic structure theory (without its extensions) to FSM decomposition or encoding were also considered by some other authors (see a.o. [4][25][27][29]).

References

[1] A. E. A. Almaini, J. F. Miller, P. Thomson, and S. Billina. State assignment of finite state machines using a genetic algorithm. *IEE Proc. on Computers and Digital Techniques*, pages 279–286, July 1995.

[2] D. B. Armstrong. On the efficient assignment of internal codes to sequential machines. *IRE Trans. on Electronic Computers*, pages 611–622, October 1962.

[3] D. B. Armstrong. A programmed algorithm for assigning internal codes to sequential machines. *IRE Trans. on Electronic Computers*, pages 466–472, August 1962.

[4] P. Ashar, S. Devadas, and A. R. Newton. *Sequential Logic Synthesis.* Kluwer Academic Publishers, 1992.

[5] E. Bruce Lee and M. A. Perkowski. Concurrent minimization and state assignment of finite state machines. *Proc. of Int. Conf. on Systems, Man and Cybernetics*, pages 248–260, October 1984.

[6] S. Chattopadhyay and P. Pal Chaudhuri. Genetic algorithm based approach for integrated state assignment and flipflop selection in finite state machine synthesis. *Proc. of Int. Conf. on VLSI Design*, pages 522–527, 1997.

[7] M. J. Ciesielski and J. Shen. A unified approach to input-output encoding for fsm state assignment. *Proc. of 28th Design Automation Conf.*, pages 176–181, 1991.

[8] G. de Micheli, R. K. Brayton, and A. Sangiovanni-Vincentelli. Optimal state assignment for finite state machines. *IEEE Trans. on CAD*, pages 269–284, 1985.

[9] S. Devadas, H. TonyMa, A. R. Newton, and A. Sangiovanni-Vincentelli. MUSTANG: state assignment of finite state machines for optimal multi-level logic implementation. *Proc. of Int. Conf. on CAD*, pages 16–19, 1987.

[10] T. A. Dolotta and E. J. McCluskey. The coding of internal sates of sequential circuits. *IEEE Trans. on Electronic Computers*, pages 549–562, October 1964.

[11] X. Du, G. Hachtel, B. Lin, and A. R. Newton. MUSE: a multilevel symbolic encoding algorithm for state assignment. *IEEE Trans. on CAD*, pages 28–38, January 1991.

[12] J. Hartmanis and R. E. Stearns. *Algebraic Structure Theory of Sequential Machines.* Prentice Hall, 1966.

[13] W. S. Humphrey. *Switching Circuits with Computer Applications.* McGraw-Hill, New York, 1958.

[14] L. Jóźwiak. *Minimal realization of sequential machines: The method of maximal adjacencies, EUT-Report 88-E-209.* Eindhoven Univ. of Tech., the Netherlands, 1988. ISBN 90-6144-209-5.

[15] L. Jóźwiak. Efficient suboptimal state assignment of large sequential machines. *Proc. of EDAC*, pages 536–541, 1990.

[16] L. Jóźwiak. Simultaneous decomposition of sequential machines. *Microprocessing and Microprogramming*, 30:305–312, 1990.

[17] L. Jóźwiak. An efficient heuristic method for state assignment of large sequential machines. *Journal of Circuits, Systems and Computers*, 2(1):1–26, 1992.

[18] L. Jóźwiak. General decomposition and its use in digital circuit synthesis. *VLSI Design*, 3(3):225–248, 1995.

[19] L. Jóźwiak and A. Postuła. Genetic engineering versus natural evolution: Genetic algorithms with deterministic operators. *Proc. of Int. Conf. on Artificial Inteligence IGAI'99*, pages 58–64, 1999.

[20] L. Jóźwiak and F. Vankan. Bit full decomposition of sequential machines: Algorithms and results. *Proc. of Canadian Conf. on Electrical and Computer Engineering*, September 1989.

[21] L. Lavagno, S. Malik, R. K. Brayton, and A. Sangiovanni-Vincentelli. MIS-MV: optimization of multi-level logic with multiple-valued inputs. *Proc. of Int. Conf. on CAD ICCAD'92*, pages 560–563, 1992.

[22] I. Lemberski. Modified approach to automata state encoding for lut-fpga implementation. *Proc. of of 24th Euromicro Conf.*, pages 196–199, 1998.

[23] B. Lin and A. R. Newton. Synthesis of multiple level logic from symbolic high-level description languages. *Proc. of IFIP Int. Conf. on VLSI*, pages 187–196, 1989.

[24] M. Martinez, M. J. Avedillo, J. M. Quintana, and J. L. Huertas. A dynamic model for the state assignment problem. *Proc. of DATE Conf.*, pages 835–839, 1998.

[25] J. Monteiro, J. Kukula, S. Devadas, and H. Neto. Bitwise encoding of finite state machines. *Proc. of 7th Conf. on VLSI Design*, pages 379–382, 1994.

[26] C. Sarwary, E. Prado Lopes, L. Burgun, and A. Greiner. Fsm synthesis on fpga architectures. *Proc. of 7th IEEE ASIC Conf.*, pages 178–181, 1994.

[27] J. Shen, Z. Hasan, and M. J. Ciesielski. State assignment for general fsm networks. *Proc. of EDAC*, pages 245–249, 1992.

[28] T. Villa and A. Sangiovanni-Vincentelli. NOVA: state assignment of finite state machines for optimal two-level logic implementation. *IEEE Trans. on CAD*, pages 905–924, 1990.

[29] M. K. Yajnik and M. J. Ciesielski. Finite state machine decomposition using multiway partitioning. *Proc. of Int. Conf. on Computer Design: VLSI in Computers and Processors, ICCD'92*, pages 320–323, 1992.