On Using Speed as the Criteria of State Selection for Minimization of Finite State Machines

Adam Klimowicz(✉)

Bialystok University of Technology, Bialystok, Poland
a.klimowicz@pb.edu.pl

Abstract. This paper presents a heuristic method for minimization of incompletely specified Mealy finite state machines. In this method, such optimization criteria as the speed and possibility of merging other states are taken into account already at the stage of minimizing internal states. Algorithms for the estimation of optimization criteria values are described. The proposed method is based on two states merging. Experimental results for two styles of state encoding and two types of programmable structures are presented. The results show that this approach to minimization of FSM in most of cases is more effective than classical methods in respect of FSM performance.

Keywords: Finite state machine (FSM) · State minimization · High speed

1 Introduction

The problem of increase of performance of electronic equipment becomes especially actual. It is connected to an acceleration of rate of human life, broad implementation of products of electronics to all spheres of human life and complication of the tasks solved by electronic equipment. High-speed performance of electronic projects is important in such spheres as computers, robotics, telecommunication, embedded systems, wired and wireless networks, transport, military and etc.

The high-speed performance of the electronic system directly depends on performance of the main control unit (system clock frequency) and also on the control units of the separate parts of the system. Generally the control unit represents the sequential circuit which mathematical model is the finite state machine (FSM). There are some approaches to increase the performance of the electronic equipment:

- the technological: using the elements with a high-speed performance;
- the system: using a piping and multi-core processors;
- the circuitry: increase of a supply voltage;
- the logical: using the synthesis methods allowing to build FSMs and control units with the maximum high-speed performance, etc.

Today the large number of scientific researches is devoted to the first three directions (especially to the first) while to the fourth direction, according to author, insufficient attention is paid. At the same time, the methods of a logic synthesis can be used

K. Saeed and W. Homenda (Eds.): CISIM 2016, LNCS 9842, pp. 493–503, 2016.
DOI: 10.1007/978-3-319-45378-1_44

for any technological basis, can be applied together with system methods, and do not depend on supply voltage.

The first attempts to combine minimization and state assignment procedures were made in [1–3]. In [1], the problem was considered for asynchronous FSMs so as to minimize the state code length. The method proposed in [2] was applicable only to state machines where the number of states did not exceed 10. In paper [3], a program for concurrent state minimization and state assignment was presented, which made it possible to form incompletely specified state codes.

The problem of the simultaneous minimization of area and signal delay on the critical path is considered in [4–7]. In [4], a structural model of the FSM called MAR model is proposed, which consists of an FSM and a combinational circuit with flip-flops in the feedback loops. In [5], codes with two unities (two-hot) and three unities (three-hot) are used. In [6], the minimization of power consumption and delay is considered for asynchronous FSMs. The concept of a low power semi-synchronous FSM operating on a high frequency is proposed that can be implemented and tested as an ordinary synchronous FSM. In [7], a two-level structural model is proposed to minimize the power consumption, area and delay. The first level of this model consists of sequential units, while the second level consists of combinational units of limited size.

The paper [8] presents a new technique for improving the performance of a synchronous circuit configured as a look-up table based FPGA without changing the initial circuit configuration; only the register location is altered. It improves clock speed and data throughput at the expense of latency. In [9] a new sequential circuit synthesis methodology is discussed that targets FPGAs and reconfigurable SoC platforms. The methodology is based on the information-driven approach to circuit synthesis, general decomposition and the previously developed theory of information relationship measures.

The paper [10] proposes a timing optimization technique for a complex finite state machine that consists of not only random logic but also data operators. The proposed technique, based on the concept of catalyst, adds a functionally redundant block – which includes a piece of combinational logic and several other registers - to the circuits under consideration so that the timing critical paths are divided into stages.

The paper [11] proposes to use the evolutionary methodology to yield optimal evolvable hardware that implements the state machine control component. The evolved hardware requires a minimal hardware area and introduces a minimal propagation delay of the machine output signals. The paper [12] is concerned with the problem of state assignment and logic optimization of high speed finite state machines. The method is designed for PAL-based CPLD implementation.

The analysis of available studies showed that there are no works in which the number of internal states and the speed of FSM are simultaneously minimized. In this paper, we propose an heuristic method for minimization of incompletely specified Mealy FSMs with unspecified values of output variables. This method is based on an operation of two states merging, where such optimization criteria as the speed (critical delay path) and possibility of merging other states are taken into account already at the stage of minimizing internal states. In addition to reduction of internal states this method minimizes the number of FSM transitions and FSM input variables.

2 Preliminaries

An ISFSM can have incompletely specified outputs and incompletely specified transitions. In practice, designers usually redefine the unspecified transitions by transitions to the present state or to the reset state. Sometimes they use transitions to an additional state, where an error signal is generated, what makes it possible to increase the functional reliability of the digital systems. In the presented approach we define the unspecified values only for the output variables and do not change the unspecified transitions.

Let us denote by L the number of FSM input variables of a set $X = \{x_1,...,x_L\}$, by N the number of FSM output variables of a set $Y = \{y_1,...,y_N\}$, by M the number of FSM internal states of a set $A = \{a_1,...,a_M\}$, and by R the minimal number of bits required to encode internal states, where $R = \text{int}\log_2 M$.

A FSM behavior can be described by the *transition list*. The transition list is a table with four columns: a_m, a_s, $X(a_m,a_s)$, and $Y(a_m,a_s)$. Each row of the transition list corresponds to one FSM transition. The column a_m contains a *present state*, the column a_s contains a *next state*, the column $X(a_m,a_s)$ contains a *transition condition* (an *input vector*), and the column $Y(a_m,a_s)$ contains an *output vector*. An ISFSM output vector is represented by ternary vector. For example, $Y(a_m,a_s)$ = "01-0", where 0 denotes zero value, 1 denotes unity value, and dash ("-") denotes a don't care value of the corresponding output variable.

The transition condition may be described in the column $X(a_m,a_s)$ in the form of conjunction of FSM input variables. The transition condition can also be represented by a ternary vector. Since the FSM behavior is deterministic, all the transition conditions from every FSM state should be mutually orthogonal. Two transition conditions are orthogonal if they have different significant values (0 or 1) at least in one position.

Two FSM states a_i and a_j can be merged, i.e. replaced by one state a_{i_j}, if they are equivalent. Equivalency of two FSM states means that FSM behavior does not change when these states are merged in one. FSM behavior does not change after merging, if the transition conditions from the states a_i and a_j that lead to different states are orthogonal. If there are transitions from states a_i and a_j that lead to the same unique state, then the transition conditions for such transitions should be equal. Moreover, the output vectors that are generated at these transitions should be not orthogonal. Note also that in case of two FSM states merging *wait states* can be formed. The detailed conditions of the states merging procedure are precisely described in paper [13].

Under FSM states merging the output vectors with unspecified values can be merged only if they are not orthogonal. Thus the significant values (0 or 1) remain without changes, and the unspecified values are replaced by the corresponding significant values. For example, let $Y(a_i,a_s)$ = "1-0-0" and $Y(a_j,a_s)$ = "−1010", and let the states a_i and a_j assume merging, then the output vector $Y(a_{i\_j},a_s)$ = "11010" will be formed at the transition from the new state a_{i_j} to the state a_s.

3 Main Minimization Algorithm

The main strategy of the proposed method consists in finding the set G of all the pairs of FSM states satisfying the merging conditions. Then for each pair of states from set G the trial merging is performed. Finally a pair (a_i,a_j) is selected for merging in such a

way that leaves the maximal possibilities for other pairs of FSM states merging. This process repeats as long as there exists a possibility of merging for at least one pair of FSM states. The method was described more precisely in paper [13].

In distinction from [13], in the present paper we chose for merging at each step the pair (a_i, a_j) that best satisfies the optimization criteria in terms of speed, and leaves the maximum possibilities for merging other pairs of states. This procedure is repeated while at least one pair of states can be merged.

Let (a_s, a_t) be a pair of states in G, where S_{st} is the estimate of speed (critical delay path), and M_{st} is the estimate of the possibility to merge other states. Regarding to the above considerations, the FSM minimization algorithm can be described as follows.

Algorithm 1 (General Algorithm for FSM Minimization)

1. Using the method described in [13], form the set G of pairs of states that admit merging. If $G = \varnothing$ (no pairs can be merged), go to step 5
2. For each pair of states (a_s, a_t) in G, calculate the estimates S_{st}, and M_{st} of the optimization criteria.
3. Choose a pair of states (a_i, a_j) for merging. Among all the pairs in G, choose a pair (a_i, a_j) for which $S_{ij} = \min$; if there are several such pairs, then choose among them the one for which $M_{ij} = \max$.
4. Merge the pair of states (a_i, a_j). Store the results of minimization (transition list and corresponding S_{st} value). Go to step 1.
5. Among all saved results of minimization select one with minimal S_{st} value.
6. Minimize the number of transitions in the FSM.
7. Minimize the number of input variables in the FSM.
8. Stop.

Merging of the states a_i and a_j (step 4 of Algorithm 1), minimization of the number of transitions (step 6 of Algorithm 1) and minimization of the number of input variables (step 7 of Algorithm 1) are performed as described in [13].

Algorithms of minimization of the number of transition an input variables are based on some observations. Suppose, for instance, that one transition from a state a_1 under condition x_1 leads to a state a_2 and the second transition from a_1 under condition $\bar{x}_1$ leads to another state a_3 and on each of these transitions not orthogonal output vectors are formed ($\bar{x}_1$ is an inversed form of the variable x_1). Suppose that the states a_2 and a_3 can be merged. After merging a_2 and a_3, a new state a_{23} is formed. Now two transitions lead from a_1 to a_{23}, one under condition x_1 and the second under condition $\bar{x}_1$. The latter means that the transition from a_1 to a_{23} is unconditional and two transitions can be replaced by one unconditional transition. Notice that in general transition conditions from a state a_1 can be much more complicated.

At minimization of the number of FSM input variables can be performed at a situation when certain input variables have no impact on the transition conditions. Suppose, for instance, that one transition from a state a_1 under condition x_1 leads to a state a_2 and another transition from a_1 under condition $\bar{x}_1$ leads to a state a_3 and the variable x_1 does not meet anywhere else in transition conditions of the FSM. Suppose that after the states a_2 and a_3 have been merged, the transition from the state a_1 to the

state a_{23} becomes unconditional, i.e. it does not depend on values of input variables. The latter means that the variable x_1 has no impact on any FSM transition and therefore it is redundant.

4 Estimation of Optimization Criteria

To estimate the optimization criteria, all pairs of states in G are considered one after another. For each pair of states (a_s, a_t) in G, a trial merging is performed. For the resultant FSM, its internal states are encoded using one of the available methods that will later be used in the synthesis of the FSM, and the system of Boolean functions corresponding to the combinational part of the FSM is built. Next, for the pair (a_s, a_t), the speed S_{st}, and the possibility of minimizing other states M_{st} are estimated. The optimization criteria for each pair of states (a_s, a_t) in G are estimated at step 2 of Algorithm 1 using the following algorithm.

Algorithm 2 (Estimation of Optimization Criteria)

1. Sequentially consider the elements of the set G.
2. For each pair of states $(a_s, a_t) \in G$, make a trial merging.
3. Perform the internal states assignment using one of the available methods.
4. Build the system of Boolean functions W corresponding to the combinational part of the FSM.
5. Estimate the critical delay path (speed) S_{st}.
6. Estimate the possibility of other states minimization M_{st}.
7. Return to the original FSM (before merging at step 2).
8. Execute steps 2–7 for all pairs of states in G.
9. Stop.

The speed of operation of an FSM is determined by the length of the critical path of its combinational part, which is equal to the number of CPLD macrocells or FPGA logic elements involved in the critical path.

The CPLD architecture is a set of functional units of which each consists of two programmable arrays - AND and OR. The outputs of the array AND are connected to the inputs of the array OR; they are called terms. Typically, the number of inputs of functional blocks in CPLDs is sufficiently large (16–54) and it usually exceeds the number of arguments of the functions implemented by the combinational part of the FSM. If the FSM is implemented on the basis of CPLD, then the cause of the decomposition of the system of Boolean functions corresponding to the combinational part of the FSM can be a large number of minterms in the DNF of a function and a large number of arguments of a function. For that reason, when the FSM is implemented on the basis of CPLD, two critical paths are found, and the longest of them is chosen.

In the case of a large number of minterms in the DNF of a function, the linear decomposition with respect to the minterms is used. The number of inputs of the OR gates is restricted by the parameter q_{max} - the maximum number of terms that can be

connected to one CPLD macrocell (typically, q_{max} for different families of CPLDs is between 12 and 90). In the case of a large number of arguments, linear decomposition of the Boolean function with respect to the arguments is used, where the number n_{FB} of inputs of CPLD functional blocks is used as the restriction (for different families of CPLDs, n_{FB} is between 16 and 54).

In the general case, the architecture of modern FPGAs can be represented as a set of logic elements based on Look-Up Tables (LUT). A feature of LUTs is that they can realize any Boolean function but with a small number of arguments (typically, 4–6 and more often 4). In the case when the number of arguments of functions to be realized exceeds the number of LUT inputs n_L, the Boolean function must be decomposed with respect to the number of arguments [14]. Among the great number of decomposition methods for Boolean functions with respect to the number of arguments, linear decomposition methods are most popular. They are used in the majority of industrial EDA tools. When the FSM is implemented on the basis of FPGA, the length of the critical path depends only on the maximum number of arguments of the realized functions.

Algorithm 3 (for the Estimation of the FSM Speed of Operation)

1. Find the maximum number L_{max} of arguments of the functions realized by the combinational part of the FSM. For each pair of states (a_s, a_t) in G, calculate the estimates S_{st}, and M_{st} of the optimization criteria

$$L_{\max} = \max_{w_i \in W} |L(w_i)|, \tag{1}$$

where $L(w_i)$ is a set of arguments of the function w_i. If the FSM is implemented on the basis of CPLD, then additionally the maximum number of minterms in the DNFs of the functions realized by the combinational part of the FSM is determined:

$$Q_{\max} = \max_{w_i \in W} |Q(w_i)|, \tag{2}$$

where $Q(w_i)$ is a set of minterms in the DNF of the function w_i.

2. Calculate the length of the critical path in the combinational part of the FSM.
   2.1. If the FSM is implemented on the basis of CPLD, then the lengths of two critical paths are found: one (S_L) depending on the maximum number of arguments (]x[is the minimal integer greater than or equal to x):

$$S_{\mathrm{L}} = 1 +](L_{\max} - n_{\mathrm{FB}})/(n_{\mathrm{FB}} - 1)[\tag{3}$$

   and the other (S_Q) depending on the maximum number of terms:

$$S_Q = 1 +](Q_{\max} - q_{\max})/(q_{\max} - 1)[. \tag{4}$$

   Set S_{st} = max(S_L, S_Q).
   2.2. If the FSM is implemented on the basis of FPGA, the length of the critical path is determined only based on the maximum number of arguments:

$$S_{st} = 1 +](L_{\max} - n_{\mathrm{L}})/(n_{\mathrm{L}} - 1)[. \quad (5)$$

3. Stop.
   The estimate M_{st} is determined by the number of pairs of the FSM that can be merged after merging the pair (a_s, a_t). To provide the best possibilities for merging other states, M_{st} should be maximized. With regard to the above considerations, the algorithm for estimating the FSM speed of operation is as follows.

Algorithm 4 (for Estimation of Possibility of Merging Other States)

1. Using the method described in [13], find the set G_{st} of pairs of states that can be merged upon merging the pair (a_s, a_t). For each pair of states $(a_s, a_t) \in G$, make a trial merging.
2. Set $M_{st} = | G_{st} |$.
3. Stop.

5 Experimental Results

The method for minimization was implemented in a program called ZUBR. The ZUBR system is the scientific-industrial software developed for design of digital systems based on programmable logic. To estimate the efficiency of the offered method we used MCNC FSM benchmarks [15]. Each of tested FSM benchmarks was encoded using binary and one-hot encoding. The speed was calculated for the initial FSM, the STAMINA program [16] and the method described in this paper.

Two models of programmable structures were used for experiments. The CPLD device model has following parameters, which can affect synthesis result: $q_{\max} = 5$ and $n_{\mathrm{FB}} = 54$. It corresponds to real CPLD devices, such as XC9500XL from Xilinx. The FPGA device model has only one parameter, which can affect synthesis result: $n_{\mathrm{L}} = 4$. It corresponds to most of real FPGA devices, such as Virtex and Spartan from Xilinx or Cyclone and Stratix from Altera.

The experimental results for binary encoding and synthesis using the CPLD device are presented in Table 1, where M_0 and S_0 are, respectively, the number of internal states and signal delay critical path (the number of logical levels after synthesis) of the initial FSM; M_1 and S_1 are, respectively, the same parameters after minimization using STAMINA and M_2, and S_2 are, respectively the same parameters after minimization using proposed method (with taking in consideration speed of FSM). S_0/S_2 and S_1/S_2 are ratios of the corresponding parameters; and *Mean* is the geometric mean value.

The analysis of Table 1 shows that application of the proposed method using binary encoding allows to reduce the number of internal states of the initial FSM. Similarly, the average reduction of the critical delay path of the FSM makes 1.44 times, and on occasion (example *train11*) 2.5 times. In comparison to STAMINA the number of states is higher in 1 case but the average reduction of the critical delay path of the FSM makes 1.67 times, and on occasion (example *bbsse*) 2.89 times.

The experimental results for one-hot encoding and CPLD device are presented in Table 2, where all parameters have the same meaning as in Table 1. The analysis of

Table 1. The experimental results for binary encoding and CPLD device

Name	M_0	S_0	M_1	S_1	M_2	S_2	S_0/S_2	S_1/S_2
bbara	10	7	7	4	7	4	1,75	1,00
bbsse	16	10	13	26	13	9	1,11	2,89
beecount	7	5	4	4	5	4	1,25	1,00
lion9	14	4	4	3	4	2	2,00	1,50
s27	6	7	5	6	5	5	1,40	1,20
sse	16	10	13	26	13	9	1,11	2,89
tma	20	6	18	16	18	6	1,00	2,67
train11	12	5	4	3	4	2	2,50	1,50
Mean							1,44	1,67

Table 2 shows that application of the proposed method using one-hot encoding allows to reduce the number of internal states of the initial FSM. Similarly, the average increase of the speed of the FSM makes 1.38 times, and on occasion (example train11) 2.5 times. In comparison to STAMINA the number of states is higher in 1 case but the average reduction of the critical delay path of the FSM makes 1.43 times, and on occasion (example tma) 2.17 times.

Table 2. The experimental results for one-hot encoding and CPLD device

Name	M_0	S_0	M_1	S_1	M_2	S_2	S_0/S_2	S_1/S_2
bbara	10	3	7	2	7	2	1.50	1.00
bbsse	16	7	13	12	13	7	1.00	1.71
beecount	7	5	4	4	5	4	1.25	1.00
lion9	14	4	4	3	4	2	2.00	1.50
s27	6	7	5	6	5	5	1.40	1.20
sse	16	7	13	12	13	7	1.00	1.71
tma	20	6	18	13	18	6	1.00	2.17
train11	12	5	4	3	4	2	2.50	1.50
Mean							1.38	1.43

The experimental results for binary encoding and FPGA device are presented in Table 3, where all parameters have the same meaning as in Tables 1 and 2.

The analysis of Table 3 shows that the average reduction of the critical delay path of the FSM makes 1.25 times, and on occasion (examples *lion9* and *train11*) 2 times. In comparison to STAMINA the number of states is higher in one case and speed of FSM for both methods is equal.

The experimental results for one-hot encoding and FPGA device are presented in Table 4, where all parameters have the same meaning as in above tables. The analysis of Table 4 shows that application of the proposed allows to reduce the critical delay path 1.27 times, and on occasion even 2 times. In comparison to STAMINA the number of states is higher in one case (example *beecount*). The average reduction of the critical delay path of the FSM is almost equal (in extreme cases - for *tma* benchmark reduction ratio is equal 1.33, but for *beecount* - 0.67).

Table 3. The experimental results for binary encoding and FPGA device

Name	M_0	S_0	M_1	S_1	M_2	S_2	S_0/S_2	S_1/S_2
bbara	10	3	7	2	7	2	1.50	1.00
bbsse	16	3	13	3	13	3	1.00	1.00
beecount	7	2	4	2	5	2	1.00	1.00
lion9	14	2	4	1	4	1	2.00	1.00
s27	6	2	5	2	5	2	1.00	1.00
sse	16	3	13	3	13	3	1.00	1.00
tma	20	3	18	3	18	3	1.00	1.00
train11	12	2	4	1	4	1	2.00	1.00
Mean							1.25	1.00

It can be noticed that the reduction of states in most of cases leads to critical path reduction of FSMs. But for 3 cases, if using STAMINA minimization program, there was a longer critical delay path than for the FSM without using minimization. These results were obtained when minimization for CPLD was performed.

Table 4. The experimental results for one-hot encoding and FPGA device

Name	M_0	S_0	M_1	S_1	M_2	S_2	S_0/S_2	S_1/S_2
bbara	10	5	7	4	7	4	1.25	1.00
bbsse	16	7	13	6	13	6	1.17	1.00
beecount	7	3	4	2	5	3	1.00	0.67
lion9	14	4	4	2	4	2	2.00	1.00
s27	6	3	5	3	5	3	1.00	1.00
sse	16	7	13	6	13	6	1.17	1.00
tma	20	6	18	8	18	6	1.00	1.33
train11	12	4	4	2	4	2	2.00	1.00
Mean							1.27	0.99

Table 5 presents the average speed for two used encoding styles for CPLD device. S_{AV0}, S_{AV1} and S_{AV2} parameters stand for the average speed of the initial FSM, the FSM after minimization using the STAMINA program and the method from this paper accordingly.

Table 5. Average speed comparison for tested encodings for CPLD device

Encoding	S_{AV0}	S_{AV1}	S_{AV2}
Binary	6.75	11.0	5.125
One-hot	5.5	6.875	4.375

The analysis of the Table 5 shows that results obtained using presented approach are better than results obtained from the STAMINA in all styles of encoding used. Also, the one-hot encoding style was the better than binary in terms of FSM speed for all considered cases for CPLD devices.

Table 6 presents the average speed for two used encoding styles for FPGA device, where all parameters have the same meaning as in Table 5.

Table 6. Average speed comparison for tested encodings for FPGA device

Encoding	S_{AV0}	S_{AV1}	S_{AV2}
Binary	2.5	2.125	2.125
One-hot	4.875	4.125	4.0

The analysis of the Table 6 shows that results obtained using presented approach are equal or better than results obtained from the STAMINA in both of styles of encoding used. Also, the binary encoding style was the better than one-hot in terms of FSM speed for all considered cases for FPGA devices.

It can be noticed, that in terms of speed, the binary encoding is more effective for FPGA devices and the one-hot encoding – for CPLD. It is related to fact, that the key parameters for FPGA devices is the number of arguments of each function, which is often higher than the number of minterms in tested benchmarks. The last mentioned parameter is a key parameter in synthesis on CPLD. To increase speed in FPGA implementation we can use other methods of decomposition, e.g. functional decomposition [17].

6 Conclusion

In this paper an efficient method for ISFSM minimization was presented. In this method, the critical delay path is taken into account already at the stage of the minimization of the number of internal states. The presented method allows to reduce the number of internal states of FSM and additionally it allows to increase a speed of FSMs comparing to STAMINA program. The proposed method also allows to reduce the number of FSM transitions and input variables. The time of execution of presented algorithm do not exceed 20 s on the computer with Intel i3 processor for FSMs having more than 200 states and 1500 transitions.

In future, more experiments with real digital systems will be performed using industrial EDA tools and specific programmable devices. In the offered approach to FSM minimization only two states merging is considered. The presented method can be modified to join a group of states containing more states. Proposed method is only the part of work on the complex minimization method [18], where speed, power consumption and area parameters are taken in consideration. In future, this method will serve to minimize power, cost and increase speed for FSM realization on programmable logic devices.

Acknowledgements. The research was done in the framework of the grant S/WI/1/2013 and financed from the funds for science by MNiSW.

References

1. Hallbauer, G.: Procedures of state reduction and assignment in one step in synthesis of asynchronous sequential circuits. In: 1974 Proceedings of the International IFAC Symposium on Discrete Systems, Riga, Pergamons, pp. 272–282 (1974)
2. Lee, E.B., Perkowski, M.: Concurrent minimization and state assignment of finite state machines. In: 1984 Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Minneapolis. IEEE Computer Society (1984)
3. Avedillo, M.J., Quintana, J.M., Huertas, J.L.: SMAS: a program for concurrent state reduction and state assignment of finite state machines. In: 1991 Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Singapore, pp. 1781–1784. IEEE (1991)
4. Rama Mohan, C., Chakrabarti, P.: A new approach to synthesis of PLA-based FSM's. In: 1994 Proceedings of the 7th International Conference on VLSI Design, Calcutta, India, pp. 373–378. IEEE Computer Society (1994)
5. Gupta, B.N.V.M., Narayanan, H., Desai, M.P.: A state assignment scheme targeting performance and area. In: 1999 Proceedings of the Twelfth International Conference on VLSI Design, Goa, India, pp. 378–383. IEEE Computer Society (1999)
6. Lindholm, C.: High frequency and low power semi-synchronous PFM state machine. In: 2011 Proceedings of the IEEE International Symposium on Digital Object Identifier, Rio de Janeiro, pp. 1868–1871. IEEE Computer Society (2011)
7. Liu, Z., Arslan, T., Erdogan, A.T.: An embedded low power reconfigurable fabric for finite state machine operations. In: 2006 Proceedings of the International Symposium on Circuits and Systems (ISCAS), Island of Kos, Greece, pp. 4374–4377. IEEE Computer Society (2006)
8. Miyazaki, N., Nakada, H., Tsutsui, A., Yamada, K., Ohta, N.: Performance improvement technique for synchronous circuits realized as LUT-Based FPGA's. IEEE Trans. Very Large Scale Integr. VLSI Syst. **3**(3), 455–459 (1995)
9. Jozwiak, L., Slusarczyk, A., Chojnacki, A.: Fast and compact sequential circuits through the information-driven circuit synthesis. In: Proceedings of the Euromicro Symposium on Digital Systems Design, Warsaw, Poland, 4–6 September 2001, pp. 46–53
10. Huang, S.-Y.: On speeding up extended finite state machines using catalyst circuitry. In: Proceedings of the Asia and South Pacific Design Automation Conference (ASAP-DAC), Yokohama, January–February 2001, pp. 583–588
11. Nedjah, N., Mourelle, L.: Evolutionary synthesis of synchronous finite state machines. In: Proceedings of the International Conference on Computer Engineering and Systems, Cairo, Egypt, 5–7 November 2006, pp. 19–24
12. Czerwiński, R., Kania, D.: Synthesis method of high speed finite state machines. Bull. Pol. Acad. Sci. Tech. Sci. **58**(4), 635–644 (2010)
13. Klimowicz, A., Solov'ev, V.V.: Minimization of incompletely specified mealy finite-state machines by merging two internal states. J. Comput. Syst. Sci. Int. **52**(3), 400–409 (2013)
14. Zakrevskij, A.D.: Logic Synthesis of Cascade Circuits. Nauka, Moscow (1981). [in Russian]
15. Yang, S.: Logic synthesis and optimization benchmarks user guide, version 3.0. Technical report, North Carolina, Microelectronics Center of North Carolina (1991)
16. Rho, J.-K., Hachtel, G., Somenzi, F., Jacoby, R.: Exact and heuristic algorithms for the minimization of incompletely specified state machines. IEEE Trans. Comput.-Aid. Des. **13**, 167–177 (1994)
17. Luba, T., Selvaraj, H.: A general approach to boolean function decomposition and its applications in FPGA-based synthesis. VLSI Des. **3**(3-4), 289–300 (1995)
18. Solov'ev, V.V.: Complex minimization method for finite state machines implemented on programmable logic devices. J. Comput. Syst. Sci. Int. **53**(2), 186–194 (2014)