Experiments on Minimization Method of Incompletely Specified Finite State Machines for Low Power Design

Adam Klimowicz(✉) and Valery Solov'ev

Bialystok University of Technology, Bialystok, Poland
{a.klimowicz,v.salauyou}@pb.edu.pl

Abstract. This paper presents a heuristic method for minimization of incompletely specified finite state machine with unspecified values of output variables. The proposed method is based on two states merging. In this method, such optimization criteria as the power consumption and possibility of merging other states are taken into account already at the stage of minimizing internal states. In addition to reduction of the finite state machine (FSM) states, the method also allows reducing the number of FSM transitions and FSM input variables. Experimental results for various styles of state assignment 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 power consumption.

Keywords: Finite State Machine (FSM) · State minimization · Low power design

1 Introduction

A finite state machine (FSM) is a model used in the design of various computation structures like sequential circuits, digital control systems, microprocessor control circuits, digital communication systems, iterative networks, communication protocols, etc. For various reasons the transitions between the FSM states or the FSM outputs may be not completely specified. An incompletely specified finite state machine (ISFSM) is the one where either the next state or the output is not specified for at least one input vector. Minimization of ISFSMs is an important task in the optimal design of sequential circuits.

A general theory for incompletely specified machines was first developed in [1]. The standard approach to solution of the problem of ISFSM's state reduction is based on a generation of sets of compatible states (or compatibles) and finding of a minimal closed cover. The problem of minimization of ISFSMs is an NP-complete problem [2] and has been studied by a number of authors. In [3], a program called STAMINA that runs in exact and heuristic modes and uses explicit enumeration for the solution of state minimization problem is presented. In [4], an exact state minimization algorithm based on mapping of ISFSMs to FSMs tree is presented. In [5], a branch-and-bound search technique for the identification of sets of compatible states is described.

The conventional approach to the synthesis of FSMs includes the following stages, which are executed sequentially: minimization of the number of internal states, state

K. Saeed and W. Homenda (Eds.): CISIM 2015, LNCS 9339, pp. 375–385, 2015.
DOI: 10.1007/978-3-319-24369-6_31

assignment and synthesis of the combinational part of the FSM. However, classical approach often contradicts the FSM optimization goal at the stage of logic synthesis because both the minimization of the number of internal states and their assignment completely ignore the features of the technological base and the requirements of logic synthesis. In [6-8], the implementation cost is minimized simultaneously with the minimization of the power consumption at the stage of state assignment. In the majority of these works, genetic algorithms are used. In [9], 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 [10], 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 analysis of available studies showed that there are no works in which the number of internal states and power consumption are simultaneously minimized. In this paper, we propose a heuristic method for minimization of incompletely specified FSMs with unspecified values of output variables. This method is based on an operation of two states merging. In this method, such optimization criteria as the power consumption 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. Incompletely specified outputs take place when the value of the output variable does not influence on functioning of the controlled object, e.g. when a carry voltage is not applied to the controlled object. Incompletely specified transitions arise when some input vectors never appear on FSM inputs, e.g. the codes of hexadecimal figures A-F at work in a decimal notation.

In practice, designers usually redefine the unspecified transitions by transitions to the present state, or to the reset state, or to an additional state, where an error signal is generated. It makes it possible to increase the functional reliability of the digital systems. In the offered approach we define the unspecified values only of the output variables and do not change the unspecified transitions.

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

A FSM behavior is 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 the state where the transition begins (*a present state*), the column a_s contains the state where the transition ends (*a next state*), the column $X(a_m,a_s)$ contains the set of values of the input variables that

initiates this transition (*a transition condition* or *an input vector*), and the column $Y(a_m,a_s)$ contains the set of values of the output variables that is generated by FSM at this transition (*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. For example, $X(a_m,a_s)$ = "1-10-0" where unit (1) means that the corresponding input variable is included in conjunction in the direct form, zero (0) means that the corresponding input variable is included in conjunction in the inversed form, and dash ("-") means that the value of the input variable does not affect the FSM transition. 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 states a_i and a_j merge, 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 under two FSM states merging *wait states* can be formed.

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.

The main strategy of the offered approach consists in finding the set D of all the pairs of FSM states satisfying the merging conditions. Then for each pair of states from D the trial merging is carried out. Finally a pair (a_i,a_j) for merging is selected in such a way that it leaves the maximal possibilities for other pairs of FSM states merging. The given process repeats as long as there exists a possibility for at least one pair of FSM states merging. The method was described more precisely in paper [11].

In distinction from [11], 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 power consumption, and leaves the maximum possibilities for merging other pairs in G. 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 P_{st} is the estimate of power consumption, and M_{st} is the estimate of the possibility to merge other states. Then, with regard 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 [11], 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 P_{st}, and M_{st} of the optimization criteria.
3. According to the specified order of optimization criteria, 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 P_{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 P_{st} value). Go to step 1.
5. Among all saved results of minimization select one with minimal P_{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 [11].

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 $\overline{x}_1$ leads to another state a_3 and on each of these transitions not orthogonal output vectors are formed ($\overline{x}_1$ 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_{2_3}, one under condition x_1 and the second under condition $\overline{x}_1$. 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 transitions one can arrive 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 $\overline{x}_1$ 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.

3 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), power consumption P_{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. Encode the internal states using one of the available methods.
4. Estimate the power consumption P_{st}.
5. Estimate the possibility of other states minimization M_{st}.
6. Return to the original FSM (before merging at step 2).
7. Execute steps 2-9 for all pairs of states in G.
8. 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. Using the method described in [11], the set G_{st} of pairs of states that can be merged upon merging the pair (a_s, a_t) must me find. After that, the parameter M_{st} can be calculated as the cardinality of the set G_{st} ($M_{st} = | G_{st} |$).

To estimate the power consumption of an FSM, we use the procedure proposed in [12] because it is the most universal and suitable for any hardware components. In addition, this procedure is adapted for the CMOS technology. The procedure described in [12] makes it possible to calculate the dynamic power consumption of an FSM based on the encoding of its internal states and the probability of occurrence of one (zero) at each input of the FSM. The power consumption depends on the encoding of internal states. For performing experiments the binary and one-hot encoding styles are used. In the one-hot encoding only one bit of the code has '1' value and all other bits have '0' value. In addition, the sequential encoding algorithm for low power design [13] is proposed to use.

According to [12], the power consumption of the FSM is determined by the rule:

$$P = \sum_{r=1}^{R} P_r = \frac{1}{2} V_{DD}^2 f C \sum_{r=1}^{R} N_r \tag{1}$$

where P_r is the power consumed by the flip-flop r, V_{DD} is the supply voltage, f is the frequency at which the FSM operates, C is the capacity of flip-flop output, and N_r is the activity of the flip-flop r.

Let k_i be a binary code of a state $a_i \in A$. Denote by k_r^i the value of the bit r in the code k_i of the state a_i. Then, the activity N_r of switching the memory flip-flop r of the FSM satisfies the equation

$$N_r = \sum_{m=1}^{M} \sum_{s=1}^{M} P(a_m \to a_s)\left(k_m^r \oplus k_s^r\right) \tag{2}$$

where $P(a_m \rightarrow a_s)$ is the probability of transiting from the state a_m to the state a_s (a_m, $a_s \in A$) and $\oplus$ is the XOR operation. The FSM must be encoded first to find the activity of each flip-flop.

The probability $P(a_m \rightarrow a_s)$ of transiting from the state a_m to the state a_s (a_m, $a_s \in A$) is given by the rule:

$$P(a_m \rightarrow a_s) = P(a_m)P(X(a_m, a_s)) \tag{3}$$

where $P(a_m)$ is the probability of the FSM to be in the state a_m and $P(X(a_m, a_s)$ is the probability of appearing the vector $X(a_m, a_s)$ initiating the transition from a_m to a_s at the input of the FSM.

The probability $P(X(a_m, a_s)$ of the vector $X(a_m, a_s)$ to appear at the input of the FSM is given by the rule:

$$P(X(a_m, a_s)) = \prod_{b=1}^{L} P(x_b = d) \tag{4}$$

where $d \in \{0, 1, '-'\}$ and $P(x_b = d)$ is the probability that the input variable x_b in the input vector $X(a_m, a_s)$ takes the value d. In this paper, we assume that 0 and 1 appear at each input of the FSM with the same probability; therefore, $P(x_b = 0) = P(x_b = 1) = 0.5$ and $P(x_b = '-') = 1$ (the probability that 0 or 1 appear at each input of the FSM is equal one because symbol '-' means logic zero or logic one and any other values cannot appear at input). For a specific FSM, $P(x_b = 0)$ and $P(x_b = 1)$ may be different; however, it must hold that $P(x_b = 0) + P(x_b = 1) = 1$.

The probability $P(a_i)$ to find the FSM in each state a_i can be determined by solving the system of equations

$$P(a_i) = \sum_{m=1}^{M} P(a_m)P(X(a_m, a_i)),\ i \in [1, M] \tag{5}$$

If there are no transitions between the states a_m and a_i, then we set $P(X(a_m, a_i)) = 0$. If there are several transitions, then $P(X(a_m, a_i))$ is the sum of the probabilities of appearing each input vector initiating the transition from a_m to a_i.

System (5) is a system of M linear equations with M unknowns $P(a_1),\ldots, P(a_M)$, which can be solved by any available method, for example, by the Gauss method. Since the FSM is always in one of its internal states, it holds that

$$\sum_{m=1}^{M} P(a_m) = 1 \tag{6}$$

To simplify the solution of system (5), one equation in (5) can be replaced with (6).

4 Experimental Results

The method for minimization of incompletely specified finite state machines was implemented in a program called ZUBR. To estimate the efficiency of the offered

method we used MCNC FSM benchmarks [14]. Each of tested FSM benchmarks was encoded using binary and one-hot encoding and sequential low power algorithm [13]. A power consumption estimation parameter was calculated using following values: output capacitance C = 3pF, frequency f = 5MHz, supply voltage V_{CC} = 5V, input probability $P(x_i = 1) = 0{,}5$. The power was calculated for the initial FSM, the FSM after minimization using method described in paper [11], the STAMINA program [3] and the method described in this paper.

The experimental results for binary encoding are presented in Table 1, where M_0 and P_0 are, respectively, the number of internal states and dissipated power (in mW) of the initial FSM; M_1 and P_1 are, respectively, the number of internal states and dissipated power (in mW) after minimization without taking in consideration power consumption (method described in [11]); M_2 and P_2 are, respectively, the number of internal states and dissipated power (in mW) after minimization using STAMINA and M_3, and P_3 are, respectively, the number of internal states and dissipated power (in mW) after minimization using proposed method (with taking in consideration power consumption). P_0/P_3, P_1/P_3, and P_2/P_3 are ratios of the corresponding parameters; and *Average* is the geometric mean value.

Table 1. The experimental results for binary encoding

Name	M_0	P_0	M_1	P_1	M_2	P_2	M_3	P_3	P_0/P_3	P_1/P_3	P_2/P_3
bbara	10	62.141	7	62.090	7	57.812	8	61.706	1.01	1.01	0.94
bbsse	16	226.014	13	226.014	13	232.976	13	226.014	1.00	1.00	1.03
beecount	7	113.275	5	91.889	4	75.116	5	91.889	1.23	1.00	0.82
lion9	14	192.569	4	93.750	4	109.375	4	84.375	2.28	1.11	1.30
s27	6	192.751	5	163.078	5	157.300	5	163.078	1.18	1.00	0.96
sse	16	226.014	13	226.014	13	232.976	13	226.014	1.00	1.00	1.03
tma	20	158.348	18	135.475	18	122.372	19	126.884	1.25	1.07	0.96
train11	12	101.902	4	78.125	4	93.750	6	68.452	1.49	1.14	1.37
Average									1.26	1.04	1.04

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 power consumption of the FSM makes 1.26 times, and on occasion (example *lion9*) 2.28 times. In comparison to method from [11] the number of states is higher in 3 cases, but the average reduction of the power consumption of the FSM makes 1.04 times, and on occasion (example *train11*) 1.14 times. In comparison to STAMINA the number of states is higher in 4 cases but the average reduction of the power consumption of the FSM makes 1.04 times, and on occasion (example *train11*) 1.37 times.

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

Table 2. The experimental results for one-hot encoding

Name	M_0	P_0	M_1	P_1	M_2	P_2	M_3	P_3	P_0/P_3	P_1/P_3	P_2/P_3
bbara	10	83.476	7	82.325	7	82.236	7	82.325	1.01	1.00	1.00
bbsse	16	252.442	13	252.442	13	252.443	13	252.442	1.00	1.00	1.00
beecount	7	169.563	5	168.899	4	146.739	5	168.899	1.00	1.00	0.87
lion9	14	129.522	4	140.625	4	156.250	6	128.906	1.00	1.09	1.21
s27	6	255.251	5	226.102	5	226.103	5	226.102	1.13	1.00	1.00
sse	16	252.442	13	252.442	13	252.443	13	252.442	1.00	1.00	1.00
tma	20	130.134	18	140.384	18	99.033	19	119.320	1.09	1.18	0.83
train11	12	124.320	4	109.375	4	136.364	4	109.375	1.14	1.00	1.25
Average									1.05	1.03	1.01

The analysis of 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 reduction of the power consumption of the FSM makes 1.05 times, and on occasion (example *train11*) 1.14 times. In comparison to method from [11] the number of states is higher in two cases, but the average reduction of the power consumption of the FSM makes 1.03 times, and on occasion (example *tma*) 1.18 times. In comparison to STAMINA the number of states is higher in 3 cases but the average reduction of the power consumption of the FSM makes 1.01 times, and on occasion (example *train11*) 1.25 times.

Table 3. The experimental results for encoding using the sequential algorithm

Name	M_0	P_0	M_1	P_1	M_2	P_2	M_3	P_3	P_0/P_3	P_1/P_3	P_2/P_3
bbara	10	52.389	7	52.664	7	52.664	8	52.171	1.00	1.01	1.01
bbsse	16	146.468	13	146.468	13	146.469	13	146.468	1.00	1.00	1.00
beecount	7	89.421	5	91.889	4	75.116	6	89.339	1.00	1.03	0.84
lion9	14	64.762	4	70.313	4	78.125	5	63.919	1.01	1.10	1.22
s27	6	168.330	5	148.634	5	148.634	5	148.634	1.13	1.00	1.00
sse	16	146.468	13	146.468	13	146.469	13	146.468	1.00	1.00	1.00
tma	20	65.250	18	71.117	18	53.106	19	63.970	1.02	1.11	0.83
train11	12	63.519	4	70.313	4	85.227	10	62.840	1.01	1.12	1.36
Average									1.02	1.04	1.02

The experimental results for encoding using the sequential low power algorithm [13] are presented in Table 3, where all parameters have the same meaning as in Table 1 and Table 2.

The analysis of Table 3 shows that application of the proposed method with low power encoding allows to reduce the number of internal states of the initial FSM on the average by 1.24 times, and on occasion (example *lion9*) by 1.75 times. Similarly, the average reduction of the power consumption of the FSM makes 1.02 times, and on occasion (example *s27*) 1.13 times. In comparison to method from [11] the number of

states is 1.28 times higher, occasionally even 2.5 times higher (example *train11*). The average reduction of the power consumption of the FSM makes 1.04 times, and on occasion (example *train11*) 1.12 times. In comparison to STAMINA the number of states is higher, occasionally even 2.5 times higher (example *train11*). The average reduction of the power consumption of the FSM makes 1.02 times, and on occasion (example *train11*) 1.36 times.

It can be noticed that the greater reduction of states in most cases leads to increased power consumption of FSMs (in 62.5% of cases). Only for one example (*s27*) a FSM of lesser power consumption was obtained using the method from [11]. In contrast, using the method, taking into account the criterion of minimizing the power consumption, there are always obtained machines with less or the same power consumption as the initial machines. Additional minimization of the number of transitions in accordance with the method [11] does not cause further reduction in power consumption.

Table 4 presents the average power consumption for all three encoding styles. P_{AV0}, P_{AV1}, P_{AV2} and P_{AV3} parameters stand for the average power consumption of the initial FSM, the FSM after minimization using the method [11], the FSM after minimization using the STAMINA program and the method from this paper accordingly.

Table 4. Average power comparison for all tested encodings

Encoding	P_{AV0}	P_{AV1}	P_{AV2}	P_{AV3}
Binary	159126.75	134554.38	135209.63	131051.50
One-hot	174643.75	171574.25	168951.38	167476.38
Sequential	99575.88	99733.25	98226.25	96726.13

The analysis of the Table 4 shows that results obtained using presented approach are better than results obtained from the STAMINA and method from paper [11] in all styles of encoding used. Also, the one-hot encoding style was the most power consumable for all minimization algorithms used and sequential algorithm was the least power consumable method of encoding for all considered cases.

Table 5 presents ratios of power consumption for sequential encoding method in relation to binary encoding (P_{AVB}/P_{AVS}) and one-hot encoding (P_{AVO}/P_{AVS}) styles for all considered minimization methods, where parameters P_{AVB}, P_{AVO}, P_{AVS} are the average power consumption for binary encoding, one-hot encoding and sequential encoding, accordingly.

Table 5. Average power consumption ratios for all tested methods of minimization

Ratio	Initial FSM	Method [11]	STAMINA	This method
P_{AVB}/P_{AVS}	1.60	1.35	1.38	1.35
P_{AVO}/P_{AVS}	1.75	1.72	1.72	1.73

The analysis of the Table 5 shows that sequential algorithm for low power encoding is the most efficient in comparison to one-hot encoding (similar values for all minimization methods and not minimized FSMs). For binary encoding style, the power consumed by FSM is higher for the initial FSM than for FSMs after minimization.

5 Conclusion

Minimization of incompletely specified finite state machines is an important step in the FSM synthesis. In this paper we presented an efficient method for FSM minimization. In contrast to traditional approaches, the proposed method allows to minimize not only the number of FSM states and consumed power, but also the number of FSM transitions and input variables.

The main goal of this method is not to find the minimal number of states but the such representation of the FSM, which consumes minimal amount of power. Of course we obtain in most cases the worse results for state minimization in comparison to methods [11] and [3], but much better for power consumption. The most important conclusion from experiments is that the FSM with minimal number of states is not in most cases the best solution in respect of power consumption.

In the offered method of FSM minimization only two states merging is considered. The given algorithm can be modified so to merge a group of states containing more than two states. Besides, the further perfection of the presented algorithm can be implemented by consideration of incompletely specified values for the transition functions as additional conditions for merging possibility of FSM internal states.

Performed experiments are only the part of work on the complex minimization method [15], where not only power consumption, but also speed and area parameters are taken in consideration. In future, this method will serve to diminish power and cost and increase speed for FSM realization on programmable logic devices.

References

1. Paull, M., Unger, S.: Minimizing the number of states in incompletely specified state machines. IRE Trans. Electron. Comput. **EC-8**, 356–367 (1959)
2. Pfleeger, C.F.: State reduction in incompletely specified finite state machines. IEEE Trans. Comput. **C-22**, 1099–1102 (1973)
3. Rho, J.-K., Hachtel, G., Somenzi, F., Jacoby, R.: Exact and heuristic algorithms for the minimization of incompletely specified state machines. IEEE Trans. Computer-Aided Design **13**, 167–177 (1994)
4. Pena, J.M., Oliveira, A.L.: A new algorithm for exact reduction of incompletely specified finite state machines. IEEE Trans. Computer-Aided Design **18**, 1619–1632 (1999)
5. Gören, S., Ferguson, F.: On state reduction of incompletely specified finite state machines. Computers and Electrical Engineering **33**(1), 58–69 (2007)
6. Xia, Y., Almaini, A.E.A.: Genetic algorithm based state assignment for power and area optimization. IEE Proc. Comput. Digital Techn. **149**(4), 128–133 (2002)
7. Aiman, M., Sadiq, S.M., Nawaz, K.F.: Finite state machine state assignment for area and power minimization. In: Proc of the IEEE Int Symposium on Circuits and Systems (ISCAS), pp. 5303–5306. IEEE Computer Society (2006)
8. Chaudhury, S., Sistla, K.T., Chattopadhyay, S.: Genetic algorithm-based FSM synthesis with area-power trade-offs. Integration, VLSI J. **42**, 376–384 (2009)
9. Lindholm, C.: High frequency and low power semi-synchronous PFM state machine. In: Proc. of the IEEE Int. Symposium on Digital Object Identifier, pp. 1868–1871. IEEE Computer Society (2011)

10. Liu, Z., Arslan, T., Erdogan A.T.: An embedded low power reconfigurable fabric for finite state machine operations. In: Proc. of the Int. Symposium on Circuits and Systems (ISCAS), pp. 4374–4377. IEEE Computer Society (2006)
11. 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)
12. Tsui, C.-Y., Monteiro, J., Devadas, S., Despain, A.M., Lin, B.: Power estimation methods for sequential logic circuits. IEEE Trans. VLSI Syst. **3**, 404–416 (1995)
13. Grzes, T.N., Solov'ev, V.V.: Sequential algorithm for low-power encoding internal states of finite state machines. J. Comput. Syst. Sci. Int **53**(1), 92–99 (2014)
14. Yang, S.: Logic synthesis and optimization benchmarks user guide. Version 3.0. Technical Report. North Carolina. Microelectronics Center of North Carolina (1991)
15. 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)