Synthesis Method of Finite State Machines Based on State Minimization for Low Power Design

Adam Klimowicz

Synthesis Method of Finite State Machines Based on State Minimization for Low Power Design

Adam Klimowicz

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

Abstract. A new method for the synthesis of finite state machines (FSMs) is proposed. In this method, such optimization criterion as the power consumption is taken into account already at the stage of minimizing internal states. In addition, the proposed method allows one to minimize the number of transitions and input variables of the FSM. The method is based on sequential merging of two internal states. For this purpose, the set of all pairs of states that can be merged is found, and the pair that best satisfies the optimization criteria is chosen for merging. The sequential algorithm is used for low power state encoding. Experimental results show, that the dissipated power is less by 7% comparing to traditional methods.

Keywords: finite state machine (FSM), state minimization, logic synthesis, low power design.

1 Introduction

In recent years digital systems appears in all spheres of human activities. Reducing the power consumption of digital devices has become more important due to especially for battery powered mobile devices. Depending on specific conditions, this parameter can be the major factor to be optimized. There are several approaches to the solution of this problem: technological, logical, system level etc. A some way of solving this problem is to reduce the power consumption of finite state machines (FSMs).

The finite state machine (FSM) provides a mathematical model that is widely used for designing digital systems, which are often designed as sequential circuits. For that reason, the optimization of FSMs with respect to the power consumption parameter is an important task.

The conventional approach to the synthesis of FSMs includes the following stages, which are executed sequentially: minimization of the number of internal states, state assignment and synthesis of the combinational part of the FSM. Under the conventional approach, a developer has only two methods of optimizing the FSM: minimization of the number of internal states and state assignment.

Often, even the exact minimization of the number of internal states does not make it possible to solve the optimization problems at the stage of logic synthesis.

In work [1], the problem of minimization and state assignment was considered for asynchronous FSMs. The method proposed in [2] is applicable only to FSMs with the number of states not exceeding 10. In [3], a program for concurrent state reduction and state encoding was presented, which made it possible to build incompletely specified state codes.

The power consumption of an FSM can be directly reduced by special encoding of internal states [4–6]. In [7-9], 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 [10], 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 [11], the conventional approach to the synthesis of FSMs is considered, under which the number of internal states is first minimized, then the internal states are encoded; next, the program ESPRESSO is used to build disjunctive normal forms (DNFs) of the functions to be realized and, finally, the cost, power consumption, and speed of operation are estimated. A set of algorithms is proposed to select the best state assignment so as to minimize the parameters mentioned above.

The analysis of available studies showed that the number of internal states and power consumption are not simultaneously minimized. The methods that claim to simultaneously take into account several optimization criteria actually reduce to the conventional approach in which several different algorithms are proposed for each stage. In the present paper, we propose a heuristic method for the minimization of incompletely specified FSMs that makes it possible to optimize a power consumption already at the stage of minimization of the number of internal states. In addition the method of synthesis also applies a special state assignment method called sequential encoding algorithm [12] designed for low power optimization. The proposed approach suits well for the implementation of FSMs on programmable logic devices (PLDs).

2 Idea of the proposed approach

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 the present, the column a_s contains the 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*).

The proposed approach is based on the method for the minimization of the number of internal states of incompletely specified FSMs (ISFSM) proposed in [13]. 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 idea of the method [13] is to sequentially merge two states. For this purpose, the set G of all pairs of internal states of the FSM satisfying the merging condition is found at each step. Then, for each pair in G, a trial merging is done. Next, the pair (a_i, a_j) that leaves the maximum possibilities for merging other pairs in G is chosen for real merging.

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 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.

After procedure of state minimization, the sequential algorithm of state assignment[12] is performed to provide encoding which minimizes the power consumption.

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 synthesis algorithm can be described as follows.

Algorithm 1 (general algorithm for FSM synthesis)

1. Using the method described in [13], form the set G of pairs of states that can be merged. If $G = \varnothing$ (no pairs can be merged), go to step 5
2. For each pair of states (a_s, a_t) in set 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. Perform state assignment using sequential algorithm [12].
9. Stop.

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_{2_3}, 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 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 $\bar{x}_1$ 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.

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. Next the internal states are encoded using sequential algorithm 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 sequential algorithm.
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 [13], 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 method given in [14] because it is the quite universal and suitable for any hardware components built using the CMOS technology. The procedure described in [14] makes it possible to calculate the dynamic power consumption of an FSM taking into account the encoding of its internal states and the probability of occurrence of ones at FSM inputs.

According to [14], the power consumption of the FSM is determined by the rule:

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

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

Let k_i be a binary code of a state a_i . 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 element r of the FSM satisfies the following equation

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

where $P(a_m \to a_s)$ is the probability of transition from the state a_m to the state a_s and $\oplus$ is the XOR operation. The FSM has to be encoded first to determine the activity of each trigger.

The probability $P(a_m \to a_s)$ of transition from the state a_m to the state a_s is given by the following equation:

$$P(a_m \to a_s) = P(a_m) P(X(a_m, a_s)) \quad (3)$$

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:

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

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. We make an assumption 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

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

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). With regard to the above considerations, the algorithm for estimating the FSM power consumption is as follows.

Algorithm 3 (estimation of the power consumption).

1. According to (4), for each input vector $X(a_m, a_s)$ $(a_m, a_s \in A)$, calculate the probability $P(X(a_m, a_s))$ of its appearance at the input of the FSM.
2. Solve system of equations (5) to find the probabilities $P(a_i)$ of the FSM to be in each state $a_i \in A$.
3. Using (3), calculate the probabilities of transitions $P(a_m \to a_s)$ of the FSM for a_m, $a_s \in A$.
4. Based on the encoding of internal states, find the activity N_r $(r \in [1, R])$ of each trigger using (3).
5. Using (1), calculate the power consumption P of the FSM for the following values of the parameters: V_{DD} = 5 V, f = 10 MHz, and C = 3 pF (these are typical values for most chips manufactured using the CMOS technology). Set $P_{st} := P$.
6. Stop.

4 State assignment procedure

The encoding of internal states is proposed to be made with the sequential algorithm more precisely described in [12]. In this method assigning the code to the state depends on states assigned earlier. It needs to define the set K^R of the all state codes that can be assigned, where $R = [\text{int } \log_2 M, M]$.

An FSM can be described by a state transition graph (STG), where the states are defined by the vertices and the transitions are defined by the edges. STG is a directed graph but for the power computations purpose we can convert it to an undirected graph because the transition from a_i to a_j causes that the same number of the flip-flops changes the output value as the transition from a_j to a_i. Weights of the edges can be expressed by:

$$w_{i,j} = P(a_i \to a_j) + P(a_j \to a_i) \tag{7}$$

Every state a_i must be assigned a code c_i. Thus internal states set A is connected with states codes set $C = \{c_1, c_2, ..., c_M\}$. Every code must be orthogonal with the all other states codes. The code width R can be any value from the range [int $\log_2 M$, M].

Let c_i^l denote l-th bit of the code of the state a_i. Hamming distance $H(c_i, c_j)$ is defined as the number of bits in the same position with the opposite phase:

$$H(c_i, c_j) = \sum_{i=1}^{R} c_i^l \oplus c_j^l \tag{8}$$

Then, with regard to the above considerations, the sequential encoding algorithm can be described as follows.

Algorithm 4 (sequential low power encoding).

1. Select two states a_i and a_j, for which $w_{i,j}$ is highest.
2. Assign two codes from the K^R, such that Hamming distance $H(c_i, c_j) = 1$. Remove the assigned state codes from K^R.
3. Repeat steps 4-5 until all states are assigned.
4. Select the unassigned state a_i, such that sum of the weights of the edges connected with the already assigned states is highest.
5. Assign the state a_i unassigned code from K^R with the lowest value of the function γ.

$$\gamma(c_i) = \sum_{j=1}^{M} w_{i,j} \cdot H(c_i, c_j) \tag{9}$$

6. Remove the code from the K^R.
7. Stop.

5 Experimental results

The method for synthesis of finite state machines was implemented in a program called ZUBR. To estimate the efficiency of the offered method we used MCNC FSM benchmarks [15] and well-known STAMINA minimization program [16] for comparison. The experiments were performed using Altera Quartus Prime version 16.0 EDA tool. All benchmarks in all three cases (without minimization, minimized with STAMINA and synthesized with proposed method) were implemented using identical design flow optimization parameters. Three parameters were taken from report files for further analysis: Core Dynamic Power (P), Total Logic Elements (C) and Maximum Clock Frequency - Fmax (F). For an implementation author has chosen the EP4CE115F29I8L device – a popular low cost FPGA from the Cyclone IV E family.

The experimental results for Core Dynamic Power 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 (without minimization); M_1 and P_1 are, respectively, the number of internal states and dissipated power (in mW) after minimization using

STAMINA and M_2, and P_2 are, respectively, the number of internal states and dissipated power (in mW) after synthesis using proposed method. P_0/P_2 and P_1/P_2 are ratios of the corresponding parameters. *Average* row contains the mean values.

Table 1. The experimental results for power and number of states

Name	M_0	P_0	M_1	P_1	M_2	P_2	P_0/P_2	P_1/P_2
LION9	9	0.21	4	0.2	5	0.20	1.05	1.00
PLANET	48	0.29	48	0.29	48	0.26	1.12	1.12
S208	18	0.27	18	0.27	18	0.24	1.13	1.13
S27	6	0.21	5	0.2	5	0.20	1.05	1.00
S386	13	0.22	13	0.22	13	0.21	1.05	1.05
S420	18	0.26	18	0.26	18	0.25	1.04	1.04
S820	25	0.35	24	0.38	25	0.34	1.03	1.12
S832	25	0.37	24	0.32	25	0.31	1.19	1.03
SAND	32	0.34	32	0.34	32	0.32	1.06	1.06
TMA	20	0.23	18	0.24	19	0.23	1.00	1.04
Average	21.4	0.28	20.4	0.27	20.8	0.26	1.07	1.06

The analysis of Table 1 shows that application of the proposed method allows to reduce the number of internal states of the initial FSM. Similarly, the average reduction of the power consumption of the FSM makes 1.07 times, and on occasion (example *S832*) 1.19 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 1.06 times, and on occasion (example *S208*) 1.13 times.

The experimental results for cost (Total Logic Elements) and speed (Fmax) are presented in Table 2, where C_0 and F_0 are, respectively, the number used logic element and maximum frequency of the initial FSM (without minimization); C_1 and F_1 are, respectively, the same parameters after minimization using STAMINA[16] and C_2, and F_2 are, respectively, the same parameters after synthesis using proposed method. *Average* row contains the mean values.

The analysis of Table 2 shows that application of the proposed method also allows to reduce the number of used logic elements in 9 of 10 cases in relation to FSMs without any minimization and in 8 of 10 cases in relation to FSMs minimized by STAMINA. In addition the maximum clock frequency in benchmarks realized with proposed method was higher than in base FSMs in 8 of 10 cases and higher than in STAMINA minimized benchmarks in 6 of 10 cases. Of course, there are examples where the cost and the speed were worse in relation to initial machines or FSMs minimized with STAMINA. It is related to fact, that in minimization method with power consumption criterion, the full minimization of states is not performed. There is al-

ways selected a result with lower power dissipation, which is not always the same as one with minimal number of states.

Table 2. The experimental results for cost and speed

Name	C_0	F_0	C_1	F_1	C_2	F_2
LION9	20	490.20	6	587.2	11	504.80
PLANET	134	369.96	134	369.96	126	457.25
S208	150	171.94	150	171.94	115	193.12
S27	22	388.50	18	450.65	18	420.34
S386	46	358.29	46	358.29	44	289.44
S420	135	177.43	135	177.43	127	175.62
S820	227	141.38	238	152.74	209	157.88
S832	247	148.92	228	143.55	204	159.92
SAND	221	164.12	221	164.12	189	194.10
TMA	84	389.41	96	237.76	84	418.41
Average	128.6	280.02	127.2	281.36	112.7	297.09

6 Conclusion

In this paper we presented an efficient method for FSM synthesis. 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 what has an influence the cost and the speed of synthesized circuits. Using the proposed method there are always obtained machines with less or the same power consumption as the initial machines or STAMINA minimized FSMs.

Presented method is the part of future work on the complex minimization method, where not only power consumption, but also speed and area parameters are taken in consideration. In the general case, the problem of choosing the group of states for merging is a multicriteria discrete optimization problem, which can be solved by various algorithms.

In future, the complex synthesis method will serve to minimize power and cost and increase speed for FSM realization on programmable logic devices.

Acknowledgements. The research has been 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: Proc. of the Int. IFAC Symposium on Discrete Systems, pp. 272-282, Riga, Pergamons (1974).
2. Lee E. B., Perkowski M.: Concurrent Minimization and State Assignment of Finite State Machines. In Proc. of the IEEE Int. Conf. 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 Proc. of the IEEE Int. Symposium on Circuits and Systems (ISCAS), pp. 1781-1784. IEEE, Singapore (1991).
4. Benini L., De Micheli G.: State assignment for low power dissipation. IEEE J. Solid State Circuits 30 (3), 259–268 (1995).
5. Chattopadhyay S.: Low power state assignment and flipflop selection for finite state machine synthesis - a genetic algorithmic approach. IEE Proc. Comput. Digital Techn. 148 (45), 147–151 (2001).
6. Koegst M., Franke G., Feske K.: State Assignment for FSM Low Power Design. In Proc. of Conf. on European Design Automation, pp. 28–33. Geneva (2003),.
7. Gören S., Ferguson F.: On state reduction of incompletely specified finite state machines. Computers and Electrical Engineering, 33 (1), 58-69 (2007)
8. 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)
9. 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).
10. 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).
11. Shiue W.-T.: Novel state minimization and state assignment in finite state machine design for low-power portable devices. Integration, VLSI J. 38, 549-570 (2005).
12. 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).
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. 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).
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. Computer-Aided Design, 13, 167–177 (1994).