Valery SALAUYOU
BIALYSTOK UNIVERSITY OF TECHNOLOGY
45A Wiejska St., 15-351 Bialystok

FSM state merging for low power

Abstract

A method of finite state machine (FSM) minimization for low power by merging FSM internal states is considered. The general algorithm for the minimization of FSM power consumption by means of merging two states is presented. The algorithm of the merging possibility of two states and the actual algorithm merging of two states for incompletely specified Mealy FSMs are given. In the conclusions, the possible directions of development of this approach are specified.

Keywords: finite state machine, low power, merging, internal states.

1. Introduction

The wide use of portable and onboard digital systems imposes stringent requirements for the power consumption of developed projects. In the general case, a digital system can be represented by a set of interacting combinational circuits and finite state machines (FSMs); therefore, a possible way for resolving the mentioned problem is the minimization of power consumption of FSMs.

Various approaches to reducing the FSM power consumption have been proposed based on the state assignment [1-3], changing the state encoding length [4], using of special structure models of FSMs [5], splitting internal states [6], and others. A more detail review of the low power FSM synthesis is provided in [6].

In this article for reduction of the FSM power consumption it is offered to use the merging of FSM internal states.

2. Determining the power consumption of an FSM

Let $A = \{a_1,..., a_M\}$ be the set of internal states of an FSM, $X = \{x_1,..., x_L\}$ be the set of input variables, $Y = \{y_1,..., y_N\}$ be the set of output variables, and $D = \{d_1,..., d_R\}$ be the set of transition functions, where R is the number of memory elements (the number of bits in internal state codes), $R \in$ [intlog2$M;M$].

To determine the power consumption of an FSM, we use the method [6], which allows one to find the dynamic power of the FSM based on encoding its internal states and the probability of occurrence of one (zero) at each input of the FSM. According to [6], the power consumption of an FSM is determined by the formula

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

where P_r is the power consumed by the flip-flop r, V_{DD} is the power voltage, f is the FSM operating frequency, C is the capacitance of the output of each flip-flop, and N_r is the switching activity of the flip-flop r ($r = \overline{1,R}$).

Let k_i be a binary code of the state $a_i \in A$. Denote by k_i^r the value of the bit r in the code k_i of state a_i ($r = \overline{1,R}$). Then, the activity N_r of switching the memory flip-flop r can be written as

$$N_r = \sum_{m=1}^{M} \sum_{s=1}^{M} P(a_m \rightarrow a_s) \times (k_m^r \oplus k_s^r), \tag{2}$$

where $\oplus$ is the exclusive OR (XOR), $P(a_m \rightarrow a_s)$ is the probability of the FSM transition from the state a_m to the state a_s (a_m, $a_s \in A$) given by the expression

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

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 the appearance of the vector $X(a_m, a_s)$ at the input of the FSM, which initiates its transition from the state a_m to the state a_s. The last probability is found by the rule

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

where $P(x_b = d)$ is the probability that the input variable x_b in the input vector $X(a_m, a_s)$ takes a value $d \in \{0, 1, \text{'–'}\}$, where '–' is the undefined value. In this paper, we assume that the probability 0 or 1 at each input of the FSM is the same; therefore, $P(x_b=0) = P(x_b=1) = 0,5$, a $P(x_b=\text{'-'}) = 1$.

The probability $P(a_i)$ of the FSM to be in each state a_i ($i = \overline{1,M}$) can be determined by solving the system of equations

$$P(a_i) = \sum_{m=1}^{M} P(a_m) \times P(X(a_m,a_i)), \quad i=\overline{1,M}. \tag{5}$$

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

The system of equations (5) is a system consisting of M linear equations in M unknowns $P(a_1),\dots,P(a_M)$. It can be solved by any available method, for example, by Gauss elimination. Since the FSM always resides in one of its internal states, it holds

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

To solve the system of equations (5), one of the equations in (5) is replaced by equation (6).

With regard to the reasoning above, the algorithm for estimating the power consumption of an FSM is as follows.

Algorithm 1

1. Use (4) to determine, for each input vector $X(a_m, a_s)$ (a_m, $a_s \in A$), the probability $P(X(a_m, a_s))$ of its appearance at the input of the FSM.
2. Solve the system of equations (5) to determine the probabilities $P(a_i)$ of the FSM to be in each state a_i ($a_i \in A$).
3. Use (3) to find the transition probabilities $P(a_m \rightarrow a_s)$ of the FSM (a_m, $a_s \in A$).
4. Based on encoding the internal states, use (2) to determine the activity of each flip-flop N_r ($r = \overline{1,R}$).
5. Use (1) to find the power consumption P of the FSM for the following parameter values: VDD = 5 V, f = 10 MHz, and C = 5 pF (these are typical values for the majority of CMOS circuits).
6. End.

3. The method of FSM minimization for low power

The general algorithm for the minimization of the FSM power consumption by merging internal states is as follows.

Algorithm 2

1. Execute the sequential method [3] of the state assignment for the FSM low power. Find the value P of the FSM power consumption by the Algorithm 1. Set $P_{start} := P; P_{min} := P$.
2. Sequentially consider all the possible pairs (a_i,a_j) of the FSM states, where $i=\overline{1,M-1}$, $j=\overline{i+1,M}$, and M is the number of the FSM internal states. If all the possible pairs have been considered, then go to step 8.
3. Check the conditions for the possibility to merge the states a_i and a_j by the Algorithm 3. If the conditions for the possibility to merge the states a_i and a_j are executed, then go to step 4, otherwise go to step 2 for considering the next pairs of the FSM states.
4. Execute the trial merging of the states a_i and a_j by the Algorithm 4.
5. Execute the sequential method [3] of the state assignment for the new FSM low power. Execute the Algorithm 1 to find the power consumption P for the new FSM. Execute a return to the FSM before merging the states a_i and a_j in step 4.
6. If $P < P_{min}$, then set $P_{min} := P$, the state pair (a_i,a_j) is remembered.
7. Go to step 2 for considering the next pairs of the FSM states.
8. If $P_{min} < P_{start}$ (i.e. the state pair whose merging led to reduction of FSM power consumption was found), then execute merging by the Algorithm 4 the state pair (a_i,a_j) which was remembered in step 6, go to step 1. Otherwise go to step 9.
9. End.

As a result of merging the FSM internal states the value M is decreased. It can lead to reduction in the probability $P(a_m)$ of the FSM to be in state a_m for some states, and also to reduction in the activity N_r of switching the memory elements [6], $r=\overline{1,R}$. Besides, the reduction of the value M leads to an increase in the number of available codes that increases possibilities of the method [3] for minimization of the FSM power consumption. In some cases, the reduction in the number of the internal states M can lead even to the reduction in the value R and as result to the reduction of the number of the memory elements. All the listed above factors promote the reduction in the FSM power consumption.

4. Merging the two internal states

The operation of the FSM is described by a State Transition Table (STT). The STT is a table consisting of four columns: a_m, $X(a_m,a_s)$, a_s and $Y(a_m,a_s)$. Each row in the STT corresponds to one transition of the FSM. The column a_m contains the present state of the FSM, the column $X(a_m,a_s)$ contains the set of values of the FSM input variables (input vector) that initiates this transition, the column a_m contains the next state, and the column $Y(a_m,a_s)$ contains the set of values of the FSM output variables (output vector) formed on this transition.

Let $P(a_i)$ be the set of transitions from the state a_i, $C(a_i)$ be the set of transitions into the state a_i, $A(a_i)$ be the set of states where the transitions from the state a_i end, $Z(a_i)$ be the set of the transition conditions from the state a_i, $W(a_i)$ be the set of the output vectors formed on the transitions from the state a_i, $a_i \in A$.

Consider the following three possible situations:

$$\begin{gathered} A(a_i) \cap A(a_j) = \emptyset, \\ A(a_i) = A(a_j), \\ A(a_i) \neq A(a_j) \text{ and } A(a_i) \cap A(a_j) \neq \emptyset. \end{gathered} \tag{7}$$

Here, Ø is the empty set.

The first condition in (7) corresponds to the situation when all the transitions from a_i and a_j lead to different states. In this case, for the merging of a_i and a_j to be possible, it is necessary and sufficient that the elements of the sets $Z(a_i) \cup Z(a_j)$ are mutually orthogonal.

This condition can be written as

$$\forall\ X(a_i,a_h),\ a_h \in A(a_i),\ \forall\ X(a_j,a_t),\ a_t \in A(a_j) \Rightarrow X(a_i,a_h) \perp X(a_j,a_t) \tag{8}$$

where the symbol «$\perp$» denotes the orthogonality of transition conditions.

The second condition in (7) corresponds to the situation when the transitions from the states a_i and a_j lead to the same set of states. In this case, a necessary and sufficient condition for the possibility of merging the states a_i and a_j (the merging conditions) is the equality of the transition conditions into the same states and the non-orthogonality of the output vectors on the transitions into the same states. The last requirements can be represented as

$$\forall\ a_h,\ a_h \in A(a_i),\ \exists\ X(a_j,a_h) \in Z(a_j) \text{ such that}$$

$$\begin{cases} X(a_i,a_h) = X(a_j,a_h) \\ Y(a_i,a_h) = Y(a_j,a_h) \end{cases} \tag{9}$$

and

$$\forall\ a_t,\ a_t \in A(a_j),\ \exists\ X(a_i,a_t) \in Z(a_i) \text{ such that}$$

$$\begin{cases} X(a_j,a_t) = X(a_i,a_t) \\ Y(a_j,a_t) \neg \perp Y(a_i,a_t) \end{cases} \tag{10}$$

Here, the symbol «$\neg$» denotes the negation.

The third condition in (7) corresponds to the situation when the transitions from the states a_i and a_j can lead both to different and common states. Let $A' = A(a_i) \cap A(a_j)$, $A'(a_i) = A(a_i) \setminus A'$, and $A'(a_j) = A(a_j) \setminus A'$. For the states a_i and a_j to be allowed to merge, it is necessary and sufficient that the transitions from the states belonging to the set A' satisfy conditions (9) and (10), but the transitions from the states $A'(a_i)$ and $A'(a_j)$ satisfy conditions (8).

We will note that at the merging of two states the wait states can be formed. If $X(a_i,a_h) = X(a_j,a_t)$ and $Y(a_i,a_h) \neg \perp Y(a_j,a_t)$, but $a_h \neq a_t$, then the states a_i and a_j may be merged if the next conditions of the emergence of the wait state are satisfied.

$$a_h = a_i \text{ and } a_t = a_j \text{ or } a_h = a_j \text{ and } a_t = a_i. \tag{11}$$

The algorithm to check the merging conditions for the states a_i and a_j used at step 3 of Algorithm 2 can be described as follows.

Algorithm 3

1. Determine the set $Z(a_i)$ and $Z(a_j)$ for the states a_i and a_j.
2. Sequentially consider the all possible pairs of the transition conditions $(X(a_i,a_h), X(a_j,a_t))$ from the sets $Z(a_i)$ and $Z(a_j)$ respectively. If all the possible pairs have been considered (i.e. for states a_i and a_j the merging conditions are satisfied), then go to step 8.
3. Check orthogonality conditions (8) for the transition conditions $X(a_i,a_h)$ and $X(a_j,a_t)$. If these conditions are orthogonal, then go to step 2 to examine the next pair of the transition conditions.
4. If $X(a_i,a_h) \neq X(a_j,a_t)$ (the transition conditions from the states a_i and a_j are different and not orthogonal), then the merging conditions for the states a_i and a_j are satisfied, go to step 8.
5. We have $X(a_i,a_h) = X(a_j,a_t)$. If the output vectors $Y(a_i,a_h)$ and $Y(a_j,a_t)$ are orthogonal, i.e. conditions are violated (9) or (10), then the merging conditions for the states a_i and a_j are not met, go to step 8.

6. We have $X(a_i,a_h) = X(a_j,a_t)$ and $Y(a_i,a_h)\ \neg\perp\ Y(a_j,a_t)$. If $a_h = a_t$, then go to step 2 (the merging conditions are met because the transitions lead to the same states under identical conditions and non-orthogonal output vectors).
7. We have $X(a_i,a_h) = X(a_j,a_t)$ and $Y(a_i,a_h)\ \neg\perp\ Y(a_j,a_t)$, but $a_h \neq a_t$. Check conditions (11) of the emergence of the wait state. If these conditions are met, then go to step 2. Otherwise, the merging conditions for the states a_i and a_j are not met, go to step 8.
8. End.

The algorithm for merging two states a_i and a_j used at steps 4 and 8 of Algorithm 2 is as follows.

Algorithm 4

1. The new state a_{i_j} is defined, and it is set $A(a_{i\_j}) := A(a_i) \cup A(a_j)$.
2. The states a_i and a_j are deleted from the set of FSM states, and the new state a_{i_j} is included into it; it is the set $A := A \setminus \{a_i\}$, $A := A \setminus \{a_j\}$, and $A := A \cup \{a_{i\_j}\}$.
3. The STT is modified to reflect the changes of the states made at step 1 and 2.
   3.1. Make the union $C(a_{i\_j}) = C(a_i) \cup C(a_j)$. Replace the states a_i and a_j with the state a_{i_j} in the column a_s of the STT.
   3.2. Make the union $P(a_{i\_j}) = P(a_i) \cup P(a_j)$. Replace the states a_i and a_j with the state a_{i_j} in the column a_m of the STT.
   3.3. If identical transitions are formed in the STT, only one of them remains, and the others are deleted.
   3.4. If transitions differing only in the unspecified values in the column $Y(a_m,a_s)$ (e.i., «1-» and «-0») are formed, then only the transition with the complete definition of the values of the output variables («10») is left in the STT.
4. End.

Example

As an example we will consider operation of the offered algorithms for the FSM state diagram in Fig. 1.

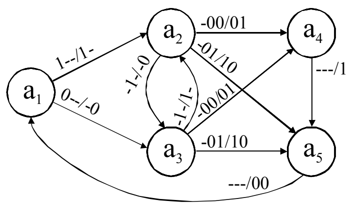

Fig. 1. The FSM state diagram

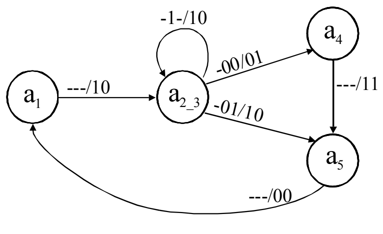

Fig. 2. The FSM state diagram after merging the states a_2 and a_3

Let $\frac{1}{2} \times V_{DD}^2 \times f \times C = 1$. According to the Algorithm 1 we will receive the following value of the FSM power consumption: $P_{start} = 25/18$. For this FSM it is possible to merge the states a_2 and a_3 as shown in Fig. 2.

According to the Algorithm 1 we will receive the new value of the FSM power consumption, which is equal to $P = 8/9 = 16/18$. Thus, by merging the states a_2 and a_3, we reduced the FSM power consumption from 25/18 to 16/18 or by 36%.

5. Conclusions

The offered method can be used after splitting the FSM states for the minimization of power consumption by the method [6]. Besides, both of these methods can be applied cyclically until it is possible to reduce the FSM power consumption by means of splitting and merging the FSM internal states.

The work was supported in part by the Bialystok University of Technology (Poland), grant no. S/WI/1/2013.

6. References

[1] Grzes T., Salauyou V., Bulatova I.: Algorithms of coding the internal states of finite-state machine focused on the reduced power consumption. Radioelectronics and Communications Systems, 2010, Vol. 53, No. 5, pp. 265-273.
[2] Solovev V.V., Grzes T.N.: An iteration algorithm of coding internal states of finite-state machines for minimizing the power consumption. Russian Microelectronics, 2013, Vol. 42, No. 3, pp. 189-195.
[3] Grzes T.N., Solov'ev V.V.: Sequential algorithm for low power encoding internal states of Finite State Machines. Journal of Computer and Systems Sciences International, 2014, Vol. 53, No. 1, pp. 92–99.
[4] Solov'ev V.V.: Changes in the length of internal state codes with the aim at minimizing the power consumption of finite-state machines. Journal of Communications Technology and Electronics, 2012, Vol. 57, No. 6, pp. 642-648.
[5] Klimovicz A.S., Solov'ev V.V.: Structural models of finite-state machines for their implementation on programmable logic devices and systems on chip. Journal of Computer and Systems Sciences International, 2015,Vol. 54, No. 2, pp. 230-242.
[6] Grzes T.N., Solov'ev V.V.: Minimization of power consumption of finite state machines by splitting their internal state. Journal of Computer and Systems Sciences International, 2015,Vol. 54, No. 3, pp. 367-374.

Received: 17.04.2015 *Paper reviewed* *Accepted: 02.06.2015*

Valery SALAUYOU, DSc, PhD, eng.

Valery Salauyou received the Ph.D. and D.Sc. degrees in computer science from the Belarusian State University Informatics and Radioelectronics, in 1986 and 2003, respectively. From 1992 he is Associate Professor at the Bialystok University of Technology. His research interests are in the area of VLSI design, with emphasis on logic synthesis, Embedded Systems, architectures of programmable logic (CPLD/FPGA/SoC), and optimization of FSMs.

e-mail: v.salauyou@pb.edu.pl