FPGA-BASED IMPLEMENTING FSM FOR EMC

Larysa Titarenko, Alexander Barkalov

Institute of Metrology, Electronics and Computer Science, University of Zielona Góra, Zielona Góra, Poland

E-mail: {l.titarenko, a.barkalo}@imei.uz.zgora.pl

Abstract
The method of synthesis and implementation into FPGAs of Mealy FSMs is proposed. Synthesis is based on structural decomposition of initial circuit. FSM states are divided by classes and encoded separately in each class. The states are decoded in the second-level circuit. It leads to implementation of FSM in double-level structure where utilization of both, LUTs and embedded memory blocks, is applied. It leads to balanced usage of hardware resources of an FPGA device. The method targets blocks for electromagnetic compatibility of radiotechnical devices.

1. Introduction

It is very important to provide the electromagnetic compatibility (EMC) for different radiotechnical devices operating in different conditions. For example, it is very important for different embedded systems [1]. It could be done using special hardware block represented as a Mealy finite state machine (FSM) [2]. Nowadays, such blocks could be implemented using field programmable gate arrays (FPGAs) [3]. FPGAs are based on look-up tables (LUTs) [4]. It is known that logic functions of FSMs could have much more arguments than it is manageable by typical LUTs. This imbalance leads to need of functional decomposition of Boolean functions [5]. The negative results of functional decomposition are increasing a number of levels of the logic circuit of FSM and increasing a number of required LUTs for implementation.

Nowadays, FPGAs are also equipped in embedded memory blocks. These blocks can be also used for realization of combinational circuits. The problem is that implementation only with memory blocks also utilize a big number of these blocks and very often exceed the number of available blocks.

One of methods of decreasing a number of FSM functions is structural decomposition of FSM [2]. Such methods apply encoding of some parameters of FSM. It leads to implementation of FSM circuit as a double-level block where reduced number of functions is implemented by the circuit of first level which is implemented with LUTs. The circuits of second level are implemented with embedded memory blocks.

The method of synthesis proposed in this article is based on the encoding of internal states divided into subsets based on a current state or a microinstruction [2]. This encoding allows decreasing a number of functions. The state is decoded in the second level circuit based on the multiple code and the code of a current state or the code of a microinstruction. Because this system is regular it can be implemented with embedded memory blocks. It leads to decrease for the number of LUTs required for implementation of FSM logic circuit and balanced usage of different resources of FPGA device in the block providing the EMC.

2. Preliminary information

Mealy FSM can be set up by a direct structural table (DST) [2] with the columns: a_m, $K(a_m)$, a_s, $K(a_s)$, X_h, Y_h, Φ_h, h. Here a_m is an initial state of an FSM, $a_m \in A$ where $A = \{a_1, \dots, a_M\}$ is a set of states; $K(a_m)$ is a binary code of the state $a_m \in A$ with $R = \lceil \log_2 M \rceil$ bits, the internal variables $q_r \in Q = \{ q_1, \dots, q_R\}$ are used to encode states $a_m \in A$; a_s is a state of transition; $K(a_s)$ is a code of the state $a_s \in A$; X_h is a conjunction of some logic elements from the set $X = \{ x_1, \dots, x_L\}$, it causes the transition $\langle a_m, a_s \rangle$; Y_h is the microinstruction which is formed during the transition $\langle a_m, a_s\rangle$, $Y_h \subseteq Y$, where $Y = \{ y_1, ..., y_N\}$ is a set of microoperations; Φ_h is the set of memory excitation functions that are equal to 1 to switch an FSM memory from $K(a_m)$ to $K(a_s)$, $\Phi_h \subseteq \Phi = \{D_1, ..., D_R\}$ as a rule D flip-flops are used to form a memory; h is a number of the DST line, $h = 1, ..., H$. This table is used as the base to form the system of functions

$$\begin{aligned} Y &= Y(Q, X), \\ \Phi &= \Phi(Q, X). \end{aligned} \qquad (1)$$

This system describes a single-level circuit of Mealy FSM (Fig. 1). As a ruke,it is called P Mealy FSM.

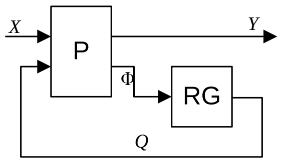

Fig. 1. Structural diagram of P Mealy FSM

Here the circuit P implements system of functions (1), the register RG represents the memory of FSM.

One of the drawbacks of the structure P is a big number of p-functions:

$$n_p(\mathrm{P}) = N + R. \qquad (2)$$

One of the known methods of decreasing this parameter is encoding of microinstructions [2]. Let DST contain T different microinstructions $Y_t \subseteq Y$. Encode each set $Y_t \subseteq Y$ by the binary code $K(Y_t)$ with $R_1 = \lceil \log_2 T \rceil$ bits ($t = 1, \dots, T$). Use variables $z_r \in Z = \{z_1, \dots, z_{R_1}\}$ for encoding of these sets. In this case Mealy FSM can be implemented as double-level circuit (Fig. 2) named as PY Mealy FSM [2]. The register RG is exactly the same as previous. The circuit Y implements the system of functions

$$Y = Y(Z) \qquad (3)$$

and transforms the variables $z_r \in Z$ into microoperations $y_n \in Y$. This circuit can be implemented using embedded memory block. Now the circuit P implements systems

$$\begin{aligned} Z &= Z(Q, X), \\ \Phi &= \Phi(Q, X), \end{aligned} \qquad (4)$$

and the number of p-functions is decreased to:

$$n_p(\mathrm{PY}) = N + R_1. \qquad (5)$$

But this number is still relatively big.

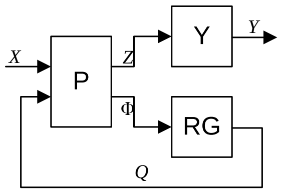

Fig. 2. Structural diagram of PY Mealy FSM

3. Background of proposed methods

The idea of further improvement is to encode also the next state using the code of a microinstruction or the code of a current state as a partial code [3].

Let divide set of internal states $a_s \in A = \{a_1, \dots, a_M\}$ into subsets based on a microinstruction Y_t. It leads into existence of T subsets $A(Y_t) \subseteq A$ and state $a_s \in A(Y_t)$ iff it is the state of transition when microinstruction Y_t is executed. Let $B_t = |A(Y_t)|$ and $B_0 = \max(B_1, \dots, B_T)$. Let encode each internal state $a_s \in A(Y_t)$ by the binary code $K_t(a_s)$ with $R_2 = \lceil \log_2 B_0 \rceil$ bits. The code is represented by variables $\tau_r \in \mathrm{T} = \{\tau_1, \dots, \tau_{R_2}\}$. In this case the code of the internal state $K(a_s)$ is represented by the multiple code of the internal state $K_t(a_s)$ and the code of the microinstruction $K(Y_t)$:

$$K(a_s) = K_t(a_s) * K(Y_t). \tag{6}$$

Digital circuit of FSM with such an encoding can be implemented as a double-level circuit named as PYY Mealy FSM (Fig. 3). The circuit Y implements the same system (4) like for PY Mealy FSM. The circuit P implements system:

$$\begin{aligned} Z &= Z(Q, X), \\ \mathrm{T} &= \mathrm{T}(Q, X), \end{aligned} \tag{7}$$

in this case. There is used additional circuit CC in this structure. It is used for decoding internal states and implements system:

$$\Phi = \Phi(Z, \mathrm{T}). \tag{8}$$

Because this circuit has regular structure it can be implemented using embedded memory blocks.

This structure permits further reduction of the number of p-functions to:

$$n_p(\mathrm{PYY}) = R_2 + R_1 \tag{9}$$

in comparison with the PY Mealy FSM. It makes that also a number of LUTs required for implementation of the circuit P is reduced and both decoders Y and CC can be implemented with memory blocks what makes that FPGA resources are used in balanced way.

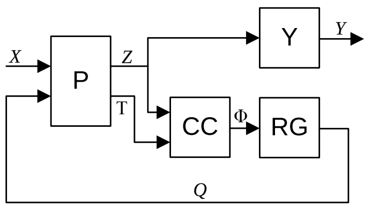

Fig. 3. Structural diagram of PYY FSM

The method where the code of a current state is used as the partial code of an internal state is very similar. In this case the set of internal states $a_s \in A = \{a_1, \ldots, a_M\}$ is divided into subsets based on a current state a_m. It leads into existence of M subsets $A(a_m) \subseteq A$ and the state $a_s \in A(a_m)$ iff it is the state of transition from the state a_m. Now $B'_t = |A(a_m)|$ and $B'_0 = \max(B'_1, \ldots, B'_T)$ and internal states are encoded by the binary code $K_m(a_s)$ with $R_3 = \lceil \log_2 B'_0 \rceil$ bits. In this case the code is represented by variables $\tau_r \in \mathrm{T} = \{\tau_1, \ldots, \tau_{R_3}\}$ and the code of the internal state $K(a_s)$ is represented by the multiple code of the internal state $K_m(a_s)$ and the code of the current state $K(a_m)$:

$$K(a_s) = Km(a_s) * K(a_m). \tag{10}$$

Digital circuit of FSM with this encoding can be implemented as a double-level circuit named as PAY Mealy FSM (Fig. 4). In this structure only the circuit CC implements different system:

$$\Phi = \Phi(Q, \mathrm{T}) \tag{11}$$

in comparison with PYY Mealy FSM.

This structure also permits reduction of the number of p-functions to:

$$n_p(\mathrm{PAY}) = R_3 + R_1 \tag{12}$$

in comparison with the PY Mealy FSM. The rule of realization in FPGA structure is the same as for the structure PYY.

It is very hard to calculate relation between n_p(PYY) and n_p(PAY) it means that structure should be selected individually for each control algorithm.

4. Proposed method of FSM synthesis

The special method of synthesis for new structures is proposed. This method includes following steps:

1. Creation and encoding of microinstructions. This step is based on trivial way of binary encoding.

2. Division of the set of internal states. The set of internal states is divided into *T* or *M* subsets. Each subset consist only states that are states of a transition during executing a *t*-th microinstruction or from a *m*-th state.

3. Multiple encoding of internal states. Internal states are binary encoded separately in each subset.

4. Formation of DST of PAY or PYY Mealy FSM. This table is formed from the original DST by replacing the column Y_h with the column Z_h and and T_h.

5. Formation of microoperations decoder table. This table contains columns $K(Y_t)$, Y_t, t. The column Y_t should be written in a binary format.

6. Formation of internal state code converter table. This table contains columns $K_t(a_s)$ or $K_m(a_s)$, $K(Y_t)$ or $K(a_m)$, $K(a_s)$, i.

7. Formation of logic equations of the circuit P. These equations form systems *Z* and T. They are formed based on the DST of PAY or PYY Mealy FSM.

8. Implementation of the logic circuit of PAY or PYY Mealy FSM. The combinational circuit P and the register RG are implemented with CLBs of FPGA – the circuit P with LUTs and the register RG with D flip-flops. The circuit Y is implemented in memory blocks

The implemented methods of synthesis was tested using benchmarks from *LGSynth91* library [7]. The analysis of all benchmarks shown that the application of the structure PAY reduce the number of required LUTs by 33% in comparison with the structure P and by 30% in comparison with the structure PY in average. The gain for the structure PYY is 19% and 16% respectively. Which method is better depends only on characteristic of implemented control algorithm.

5. Conclusion

The proposed design methods target providing EMC in complex systems. They permit to decrease the number of logic elements required for implementation of combinational circuit of FSM. Using LUTs and EMBs allows utilizing different kinds of FPGA resources.

The results of our investigations show that proposed methods lead to minimizing the number of LUTs required for FSM circuit. Analysis of standard benchmarks shows that these methods give better results in comparison with the known design approaches.

These methods could be applied for implementing hardware blocks providing electromagnetic compatibility in telecommunications. Of course, these blocks should be sequential units represented by models of finite state machines.

References

1. A.Barkalov, L.Titarenko, M.Mazurkiewicz. Foundations of embedded systems. – Berlin: Springer, 2019.
2. S. Baranov. Logic Synthesis for Control Automata. - Boston: Kluwer Academic Publishers, 1994.
3. A. Barkalov, L.Titarenko. Logic synthesis for FSM-based control units. - Berlin: Springer, 2009.
4. V. Sklyarov, I. Skliarova, A. Barkalov, L.Titarenko. Synthesis and Optimization of FPGA-based Systems. – Berlin: Springer, 2014.
5. A.Barkalov, L.Titarenko. Logic synthesis for compositional microprogram control units. - Berlin: Springer, 2008.