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

Performance Targeted Synthesis of ASM Controllers on FPGA

Abstract

Performance-driven synthesis of controller circuits is very important and challenging task in digital systems design. The clock frequency of a synchronous sequential logic circuit is dependent in a large part on the maximum propagation delay through its combinational block. The paper presents a new method for FPGA-based design of high-speed Algorithmic State Machine (ASM) controllers. The proposed approach is based on the introduction of additional states of the state machine in order to simplify transition and output logical functions to implement them in the single-level structures. The proposed technique is applied at the stage of converting the ASM chart to the finite state machine description and allows obtaining such an HDL specification that provides an increase in the designed system speed. Experimental results show that our approach achieves an average performance gain of 22.24% to 29.72% (for various FPGA devices) compared with the conventional synthesis method.

Keywords: Algorithmic State Machine (ASM), ASM chart, Finite State Machine (FSM), Performance optimization.

1. Introduction

Controller circuits form the core of many digital systems. An Algorithmic State Machine (ASM) controller is a control unit whose hardware algorithm is specified by an ASM chart. It is a powerful method for describing the behavior of complex control system designs. ASM controllers are commonly modeled by a finite state machine (FSM).

The main optimization criteria used in FPGA-based controller circuits design are performance, area, and power consumption. Since FPGAs have huge resources of logic elements, the utilized area is not often a critical parameter. However, the performance of digital systems is a significant requirement for many real-time applications. The speed of the FSM design can be evaluated based on the maximum clock frequency, which is determined in a large part by the worst-case delay through its combinational block.

ASM-based synthesis methods have been developed by several authors. The ASM charts for specifying digital designs were first documented in [1]. Various ASM-based synthesis methods were developed in [2]. In [3], the techniques for converting ASM charts into PLA-, PAL-, and multiplexer-based implementations are described. Paper [4] proposes the algorithmic register transfer language for formal text specifying of the ASM charts. In [5], the technique for minimizing the number of ASM vertices, which results in state and logic circuit minimization, is presented. Paper [6] proposes a new notation called "ASM++ diagrams" which improves the ASM charts for more complex designs and allows for the automatic conversion into HDL code, is considered. In [7], synthesis methods for ASM controllers based on ASM charts with linear chains of states are proposed.

In a series of papers [8-14], different techniques for sequential circuits performance optimization were developed. Most of these techniques improve FSM performance by reducing the longest path delay in combinational logic, which results in maximum speed. A similar approach is used in our method, but in contrast to the above studies, we propose to apply our technique at an earlier stage of ASM conversion to the FSM description. As a result, we get such an FSM specification in HDL language, that already provides an increase in the designed system speed.

In this paper, we propose a novel performance-targeted synthesis method for ASM controllers implemented in FPGAs. As a model for the ASM controller, the Moore finite state machine is used. The proposed approach is based on introducing additional states to implement all the transition and output functions in single-level structures with a minimal path delay.

The main contribution of the work is a new technique to minimizing the worst-case delay allowing the design performance optimization at the logic synthesis level.

2. Idea of the proposed technique

An ASM chart is a high-level flowchart notation to specify the hardware algorithms of controller circuits. It represents the flow of operations to be performed. The ASM controller outputs from the set $Y=\{y_1,\dots,y_N\}$ are indicated inside the ASM operator vertices. The ASM controller inputs from the set $X=\{x_1,\dots,x_L\}$ are placed within decision vertices which decide the branching from a given state. An example of an ASM chart with $Y=\{y_1,\dots,y_8\}$ and $X=\{x_1,\dots,x_8\}$ is shown in Fig.1.

When designing the Moore machine from the ASM chart, we have to assign the state labels to the ASM operator vertices as follows: the initial and final vertices are marked with label a_1, and the remaining state labels $a_2,\dots,a_M$ are assigned to the ASM operator vertices. The state labels $a_1,\dots,a_M$ correspond to the FSM internal states.

To specify an FSM from the marked ASM chart, we must describe all transition paths between the identified internal states. We can describe the FSM behavior by the *transition list*. Each element of the transition list corresponds to one FSM transition and contains four items: a_m is the present state, a_s is the next state, $X(a_m, a_s)$ is the transition condition, and $Y(a_m)$ is the set of output variables asserted at state a_m. The logical condition $X(a_m, a_s)$ is specified as a conjunction of input variables that initiate this transition.

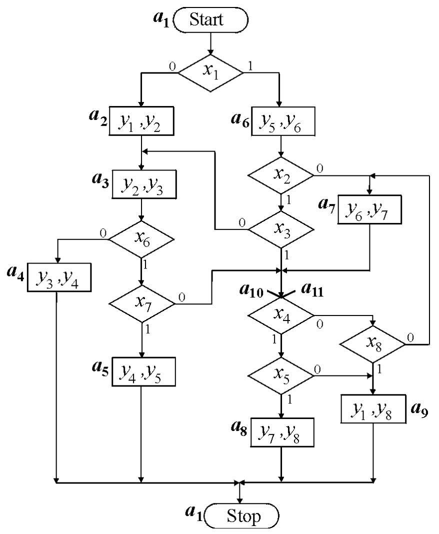

Fig. 1. An ASM example

Our method is targeted at LUT-based FPGA implementation and considers specific features of FPGA logics. The n-input LUT (n-LUT) is a small static RAM with 1-bit output that can implement any logical function of up to n variables. In practice, LUTs have a number of inputs limited to 4-6. A function of more than n variables can be implemented as a multi-level network of LUT components, which causes an increase in output signal delay. For FSM design, the maximum clock frequency of synchronous circuit is determined by the maximum propagation delay through the combinational block. Therefore, a reliable method to increase FSM performance is to reduce the worst-case delay in FSM transition functions logic.

We propose an FSM synthesis technique that ensures single-LUT-level implementation of all transition and output functions, which results in FSM performance improvement. One-hot encoding is the most suited state assignment approach to high-speed FSM design, because it allows simplifying decode logics in transition and output functions. The n-LUT based implementation of transition function from state a_i needs only one LUT input to the current state decoding, and the remaining n-1 inputs of LUT can be used to check the transition condition.

Let $A=\{a_1,\dots,a_M\}$ be the set of main internal states of FSM; $X(a_m, a_s)$ be the set of input variables that affect the transition from a_m to a_s, which we refer as transition condition. If $|X(a_m, a_s)| > n$-1, then it is impossible to implement the transition from a_m to a_s in a single LUT block. Such a transition function can be implemented by a network consisting of more LUTs, but it leads to an increase in signal propagation delay.

In our approach, we propose to introduce additional internal states of FSM to simplify the transition functions and enable their implementation in fast single-LUT-level structures. Additional states should be introduced to satisfy the following condition for all FSM transitions:

$$|X(a_m, a_s)| > n-1, \forall a_m, a_s \in A. \quad (1)$$

If condition (1) is satisfied, each transition function can be implemented in a single LUT and the whole combinational block of the FSM can be realized in a single-logic-level structure. This helps in reducing the worst-case propagation delay and thereby increasing the maximum clock frequency.

3. Algorithms for ASM controller synthesis

The proposed method for high-speed ASM controllers synthesis consists of two stages. In the first stage, the places for entering the additional labels on the ASM chart are determined, and in the second stage, the additional FSM states are introduced. These steps should be repeated until the condition (1) is met for all transitions. The algorithm for determining the place of the addition label on the ASM chart is as follows.

Algorithm 1

1. Build the chains of decision vertices for all ASM transition paths that do not meet the condition (1).
2. Place the SL labels through every n–1 vertex starting from the beginning of the transition chain and the FL labels through every n–1 vertex starting from the end of the chain.
3. For each chain, all vertices labeled with SL and FL and those that are between the corresponding SL and FL labels, are counted among the potential vertices for the additional state labeling.
4. To select one of the vertices as a place for an additional label, create matrix W. The rows of matrix W correspond to the chains defined in step 1. The columns of matrix W correspond to the vertices pointed as potential vertices to introduce additional labels. The entry w_{ij} of matrix W is equal to 1 if the vertex in column j is counted among the potential vertices of the chain in row i.
5. To determine the place of the additional label on the ASM chart, select the matrix column with the maximum number of 1s.
6. Stop.

Let v_i be the decision vertex chosen according to Algorithm 1; $P(v_j)$ be the set of transitions for which vertex v_j has been pointed as a potential place to introduce the additional label; and $p(a_m)$ be the transition from the state a_m. Since transitions from several different states in which the different outputs are generated may lead to vertex v_i, then more than one additional labels may be placed at the input of v_i. The output signals formed in the corresponding additional states are the same as those generated in the initial state of each transition. The algorithm of introducing additional states is presented as follows.

Algorithm 2

1. Determine the set $A^*=\{a_{j1},\dots,a_{jK}\}$ of the initial internal states for transitions of the set $P(v_j)$.
2. Mark the vertex v_j using additional labels $a_{M+1},\dots,a_{M+K}$.
3. Include the states $a_{M+1},\dots,a_{M+K}$ to the FSM transition list.
4. Define the sets $Y(a_{M+k})$ of the output signals in the additional states $a_{M+1},\dots,a_{M+K}$ as $Y(a_{M+k}) = Y(a_{jk})$.
5. Modify the transitions of the set $P(v_j)$ so that each transition $p(a_{jk}) \in P(v_j)$ is ended in the additional state $a_{M+}k$ in which the outputs $Y(a_{jk}) = \Upsilon(a_{M+k})$, are generated.
6. Stop.

To synthesize the Moore FSM, the ASM chart shown in Fig. 1 has been marked with the main state labels $a_1,\dots,a_9$. Let n=4, then there are eight FSM transitions (from states a_3 and a_6) for which the condition (1) is not satisfied. According to Algorithm 1, the vertex with logical condition x_4 was selected for marking with the additional label. Because $A^*=\{a_3, a_6\}$, and $K=|A^*|=2$, two additional states a_{10} and a_{11} will be introduced. These additional states differs only in output signals generated: at the state a_{10} the output set $Y(a_{10}) = Y(a_3) = \{y_2, y_3\}$ corresponding to the state a_3 is generated; but at the state a_{11} the same output set $Y(a_{11}) = Y(a_6) = \{y_5, y_6\}$ as at the state a_6 will be asserted.

4. Experimental results

To estimate the efficiency of the proposed method, we applied it to the ASM examples from the ASM charts benchmark suite of ZUBR design system [15]. The ASM examples are characterized by a large number of input variables (from 9 to 18), this allows an effective applying the proposed technique. FPGA implementation was performed using the Intel Quartus Prime Lite Edition v.17.0 tool with default optimizing options. Targeted FPGAs are 4-LUT-based devices from the MAX and Cyclone series, therefore the parameter n=4 was used in all tested synthesis methods.

The results of comparing the performance of the implemented FSMs are shown in Tables 1 and 2, where F and F^* are the rounded maximum clock frequencies (in MHz) for the conventional method (without the additional states) and the proposed synthesis method, respectively; and "%" is the percentage increase in the maximum clock frequency when using our synthesis method.

Tab. 1. Performance comparison results for MAX series devices

Example	**MAX II**			**MAX 10**		
	F	***F****	*%*	***F***	***F****	*%*
prim2r	330	405	22.73	454	565	24.45
std2r	272	348	27.94	333	477	43.24
gsa5d	234	356	52.14	323	430	33.13
sstr6	269	339	26.02	385	470	22.08
dst7	285	360	26.32	340	466	37.06
mnst	299	354	18.39	359	439	22.28
sstr2r	297	335	12.79	469	585	24.73
gsa10	321	383	19.31	390	510	30.77
mean			25.71			29.72

Tab. 2. Performance comparison results for Cyclone series devices

Example	**Cyclone IV E**			**Cyclone IV GX**		
	F	***F****	*%*	***F***	***F****	*%*
prim2r	634	710	11.99	631	720	14.10
std2r	521	597	14.59	452	659	45.80
gsa5d	473	539	13.95	465	550	18.28
sstr6	524	597	13.93	511	568	11.15
dst7	500	682	36.40	523	590	12.81
mnst	521	620	19.00	532	630	18.42
sstr2r	578	685	18.51	569	708	24.43
gsa10	443	705	59.14	535	711	32.90
mean			23.44			22.24

The results in Tables 1 and 2 show that applying the proposed method can improve performance, on average, by 22.24% to 29.72% for different FPGA series. The maximum performance increase achieved is 59.14% for test example *gsa10* targeted at Cyclone IV E FPGA devices.

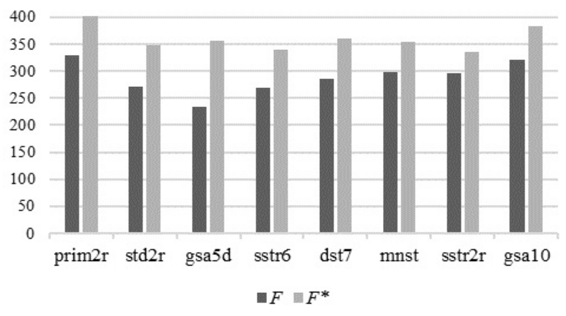

Fig. 2. Performance comparison for MAX II devices

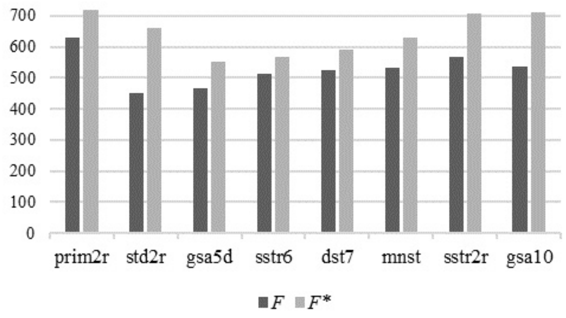

Fig. 3. Performance comparison for Cyclone IV GX devices

Some results from Tables 1 and 2 are illustrated in Fig.2 and Fig.3, which show that introducing the additional states results in performance improvement for all studied examples.

5. Conclusions

In this paper a new method for the design of high-speed ASM controllers was presented. The method is based on the introduction of additional FSM states, which allows to simplify the transition functions logics to implement all the functions in the faster one-LUT-level structures. The proposed approach allows reducing the critical path delay and the minimum clock period, resulting in a maximum speed. Because the method is applied at the early stage of converting the ASM to the state machine HDL description, it can be used with other known optimization methods to improve the performance results.

The work was supported by Bialystok University of Technology grant S/WI/3/2018.

6. References

[1] Clare C.: Designing logic systems using the state machines. McGraw-Hill, New York, 1973.
[2] Baranov S.: Logic Synthesis for Control Automata. Kluwer Academic Publisher, Boston, 1994.
[3] Green D.H., Chughtai M.A.: Use of multiplexers in the direct synthesis of ASM-based designs. IEE Proc. E, Comput. & Digital Tech. Vol. 133, No 4, pp.194–200, 1986.
[4] Lu J.Y., Kim J.D., Chin S.K.: Hardware composition with hardware flowcharts and process algebras. In: 2nd IEEE International Conference on Engineering of Complex Computer Systems Proceedings, IEEE, Montreal, Canada , pp. 352–364, 1996.
[5] Baranov S.: Minimization of algorithmic state machines. In: 24th Euromicro Conference Proceedings, IEEE, Vasteras, Sweden, pp. 176–179, 1998.
[6] De Pablo, S., Cáceres S., Cebrián J.A., Berrocal M.: A proposal for ASM++ diagrams. In: Design and Diagnostics of Electronic Circuits and Systems Proceedings, IEEE, Krakow, Poland, , pp. 1–4, 2007.
[7] Barkalov A., Titarenko L., Bieganowski J.: Logic Synthesis for Finite State Machines Based on Linear Chains of States. Studies in Systems, Decision and Control, Springer, Berlin, 2018.
[8] Hertwig A., Wunderlich H.: Fast controllers for data dominated applications. In: European Design and Test Conference (ED & TC '97) Proceedings, IEEE, Paris, France, pp. 84–89, 1997.
[9] Czerwinski R., Kania D.: State Assignment and Optimization of Ultra-High-Speed FSMs Utilizing Tristate Buffers. ACM Transactions on Design Automation of Electronic Systems (TODAES), Vol. 22, No 1, Article 3, 2016.
[10] Kim E., Lee D., Saito H., Nakamura H., Lee J., Nanya T.: Performance optimization of synchronous control units for datapaths with variable delay arithmetic units. In: ASP-DAC Asia and South Pacific Design Automation Conference Proceedings, IEEE, Kitakyushu, Japan, pp. 816–819, 2003.
[11] Weng S., Kuo Y., Chang S.: Timing Optimization in Sequential Circuit by Exploiting Clock-Gating Logic ACM Transactions on Design Automation of Electronic Systems (TODAES), Vol. 17, No 2, Article 16, 2012.
[12] Bommu S., O'Neill N., Ciesielski M.: Retiming-based factorization for sequential logic optimization. ACM Transactions on Design Automation of Electronic Systems (TODAES), Vol. 5, No 3, pp. 373–398, 2000.
[13] Huang S.: On speeding up extended finite state machines using catalyst circuitry. In: ASP-DAC 2001 Asia and South Pacific Design Automation Conference (DAC '06) Proceedings, IEEE, Yokohama, Japan, pp. 583–588, 2001.
[14] Gupta G.R., Gupta M., Panda P.R.: Rapid estimation of control delay from high-level specifications. In: 43rd Design Automation Conference Proceedings, ACM, San Francisco, USA, pp. 455–458, 2010.
[15] Salauyou V., Klimowicz A., Grzes T., Bulatowa I., Dimitrowa-Grekow T.: Synthesis methods of finite state machines implemented in package ZUBR. In: 6th International Conference Computer-Aided Design of Discrete Devices (CAD DD'7) Proceedings, National Academy of Sciences of Belarus, Minsk, pp. 53–56, 200).

Received: 10.12.2017 *Paper reviewed* *Accepted: 01.02.2018*

Valery SALAUYOU, DSc, PhD, eng.

He received the PhD and DSc degrees in computer science from the Belarusian State University of Informatics and Radioelectronics, in 1986 and 2003, respectively. Since 1992 he is an 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: valsol@mail.ru

Irena BUŁATOWA, PhD, eng.

She received the PhD degree in computer science from the Belarusian State University of Informatics and Radioelectronics in 1992. Currently she is a senior lecturer at the Computer Science Department at the Bialystok University of Technology. Her research interests include control units design and optimization.

e-mail: i.bulatowa@pb.edu.pl