Marian ADAMSKI, Monika WIŚNIEWSKA, Remigiusz WIŚNIEWSKI, Łukasz STEFANOWICZ

Uniwersytet Zielonogórski, Instytut Informatyki i Elektroniki

Application of hypergraphs to the reduction of the memory size in the Microprogrammed Controllers with Address Converter

Streszczenie. *W artykule zaproponowano metodę redukcji pojemności pamięci sterowników mikroprogramowanych z konwerterem adresów. Metoda bazuje na dwupoziomowej redukcji pamięci sterownika. Pierwszy etap to usprawnienie kodowania mikroinstrukcji, poprzez wprowadzenie konwertera adresów do struktury układu mikroprogramowanego. Właściwa redukcja pamięci sterownika stanowi drugi krok proponowanej metody. W tym celu zastosowano teorię hipergrafów oraz specyficzne własności tych struktur. (**Zastosowanie hipergrafów w redukcji pojemności pamięci w sterownikach mikroprogramowanych z konwerterem adresów**).*

***Abstract**. In the article the method of reduction of the control memory size in the microprogrammed controllers with address converter is proposed. The idea is based on the two-level reduction of the memory size. The first step includes application of an additional block (address converter) in the structure of the controller. Such a module improves the addressing of microinstructions to reduce the volume of the control memory. Next, the hypergraph theory is applied for further reduction of the control memory, where concurrent microoperations are encoded together.*

Słowa kluczowe: hipergrafy, sterowniki mikroprogramowane, redukcja, pamięć, konwerter adresów.
Keywords: hypergraphs, microprogrammed controllers, reduction, memory, address converter.

Introduction

A control unit (CU) is one of the most important parts of a digital system [5,7,8,10,11]. The most popular method of CU realization bases on a finite state machine, FSM [2,8, 9,11]. However, it is known that in the case of the linear flow-chart, the microprogrammed controller (compositional microprogram control unit) requires less amount of hardware than control unit based on the traditional FSM model [1,3,14].

It is known that in case of linear flow-chart the microprogrammed controller (also called "compositional microprogram control unit") requires less amount of hardware than traditional FSM model [1,2,3,15]. In a microprogrammed controller, the control unit is decomposed into two main parts. The first one is responsible for microinstructions addressing [1,3], while the second part holds and generates proper microinstruction [3,15]. Such a solution permits to reduce the number of logic elements that are used for implementation of CU [15]. Furthermore, wider areas of the target device can be applied for other modules of the prototyped system. In the traditional implementation of the controller with sharing codes each additional bit in the microinstruction address doubles the total volume of the memory [3,15]. As the volume of memories in embedded systems are limited [3,7,9,10] it causes the serious problems during the implementation of the controller and the memory ought to be decomposed.

In the paper we propose the two-level reduction method of the control memory. Firstly, an additional block (address converter) is applied to the structure of the controller. Such a module improves the addressing of microinstruction. Next, the reduction of microinstruction length is performed. The method bases on the hypergraph theory, where concurrent microoperations are encoded together [13,14].

Microprogrammed controllers

In the microprogrammed controller, the control unit is decomposed into two main parts [1,3]. The first one is in response of microinstructions addressing. It is a simplified finite state machine. The second part holds and generates adequate microinstructions. Such an implementation permits to minimize the number of logic elements in the destination programmable device (like FPGA) [14].

One of the most popular implementation is microprogrammed controller with sharing codes (U_{SC}). Figure 1 presents the idea of the microprogrammed controller with sharing codes.

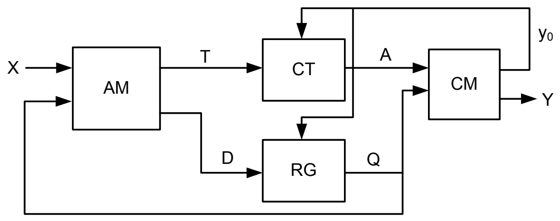

Fig.1. The microprogrammed controller with sharing codes

The controller can be divided into four units: addressing module AM, counter CT, register RG and control memory CM. The addressing module generates excitation functions for the counter and for the register:

(1) $T = f(X,Q)$,
(2) $D = f(X,Q)$.

The RG and CT is in charge of holding the code of the current state of the controller. Additionally, RG generates an upper part of the microinstruction address and the CT determines the lower part of the microinstruction address.

The main benefit of the realization of the control unit as the microprogrammed controller with sharing codes is a possibility of implementation of the circuit CM with embedded memories [15]. Other blocks of the prototyping system are implemented with the logic blocks (flip-flops and LUT elements) of the FPGA [1,3,15]. Such an idea leads to the reduction of the number of logic blocks in comparison with the realization of the controller as a traditional FSM and thus, the designer can allocate wider area of the FPGA for other blocks of the prototyping system.

In the traditional implementation of the microprogrammed controller with sharing codes each additional bit in the microinstruction address doubles the total volume of the memory [3,14]. The volume of memories in embedded systems are limited, thus it can cause the serious problems during the implementation of the controller and the memory ought to be decomposed [3].

Microprogrammed controller with address converter

The method of sharing codes makes sense only if the size of codes generated by the register (R_1) and by the counter CT (R_2) is equal to the width of the microinstruction address (R_3) [3,15]. Then the following condition is fulfilled:

(3) $$R_1 = R_2 + R_3$$

In most cases the total number of bits generated by the register and by the counter exceeds the width of the microinstruction address. The condition (3) is violated because $R_1 + R_2 > R_3$ and the volume of the control memory grows drastically. Each additional bit in the microinstruction address doubles the total volume of the memory.

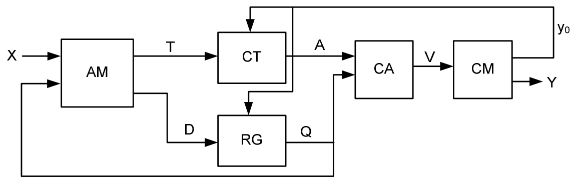

Fig.2. The microprogrammed controller with address converter

The idea of microprogrammed controller with address converter (U_{CA}) is shown in the Figure 2. An additional, block address converter (AM) permits to keep the minimal possible addressing of the microinstruction even if the condition (3) is violated. The module AM converts addresses generated by RG and CT:

(4) $$V = V(A,Q),$$

where $V=\{v_1,\ldots,v_{R3}\}$ is the set of addresses of the control memory.

Application of the hypergraph theory

In the second stage of the proposed method a hypergraph theory is applied [4,6,13]. The aim of this part is reduction the volume of the memory. The process can be divided into the following steps:

1. Formation of the set of compatibility classes for microoperations that are kept in the control memory. Two microoperations are compatible if they are not executed concurrently [13,14].
2. Calculation of the weight of each compatibility class. A weight of a class is equal to the minimum number of bits that are required for its encoding [14].
3. Formation of the compatibility hypergraph [14]. Such a structure represents relations between compatibility classes and microoperations.
4. Computation of the minimum transversal (vertex covering) of a hypergraph [4,6]. This is an essential step of the algorithm. Let us point out that results can be retrieved via any known method of minimum transversal calculation.

   The detailed analysis of the effectiveness of the particular method application was discussed in [14]. Four different algorithms of minimum transversal computation were presented and compared:
   - fast reduction algorithm (traditional method);
   - backtracking algorithm;
   - greedy algorithm;
   - mix (fast & greedy algorithms).

   The results of experiments showed, that the best results were gained after application of the greedy algorithm. Of course, for relatively small memories (20 microinstructions x 20 microoperations) backtracking method returns the best result, however it was unable to calculate the covering for bigger memories [14]. Therefore, experiments presented in this article were performed with greedy algorithm.
5. Calculation of the total weight of each minimum transversal (if exist more than one). The transversal with the lowest weight is selected for the further analysis.
6. Encoding of compatibility classes that belong to the selected transversal.
7. Formation of the new content of the memory. Each new microinstruction is formed as a concatenation of codes achieved in the previous step.

The reduction of the memory can be easily calculated as:

(5) $$t = \left(1 - \frac{\sum_{i=1}^{|S|} L_i}{N} \right) \cdot 100\ \%$$

where t means the percentage gain achieved during reduction;
$|S|$ is the number of compatibility classes that belong to the minimum transversal; L_i is the weight of the i-th class that belongs to the transversal; N is the original size of microinstruction (equal to the total number of microoperations) before reduction [14].

Results of experiments

Proposed method has been verified experimentally. The library of benchmarks consists of over 100 test modules. Test vectors include both, real and hypothetical devices. Each module was prepared in a three ways:
- as a traditional controller with sharing codes;
- as a device with address converter;
- as a device, where both: address converter and hypergraph theory were applied (will be denoted in the future as *proposed method*).

Table 1. Results of experiments (1)

Benchmark	Volume of memory (sharing codes)	Volume of memory (proposed method)	Percentage reduction of the memory
big_mem	227328	33280	85%
contr_14	172032	6656	96%
test_014	448	160	64%
test_033	155648	26112	83%
traffic_07	86016	8192	90%
traffic_light	2304	384	83%
testAW_05	180224	8192	95%
Average	**59834**	**5517**	**91%**

Tables 1 and 2 shows the results of performed experiments. There are only representative benchmarks presented, however the summary shows the average results of all benchmarks in the test library. To clarify the presentation, the results are shown in two tables. The first one illustrates the comparison between controller with sharing codes and controller with proposed method (where address converter and hypergraph theory were applied). This is an essential result. However, to see the benefit of hypergraphs application, we also show the second table. It presents the comparison between controller with address converter and the controller with proposed method, after the reduction of memory based on the hypergraph theory.

Table 2. Results of experiments (2)

Benchmark	Volume of memory (address converter)	Volume of memory (proposed method)	Percentage reduction of the memory
big_mem	56832	33280	41%
contr_14	10752	6656	38%
test_014	224	160	29%
test_033	38912	26112	33%
traffic_07	10752	8192	24%
traffic_light	576	384	33%
testAW_05	11264	8192	27%
Average	**11145**	**5517**	**50%**

From above tables we can see that application of the proposed method permits to reduce the volume of the memory on **average by 90%**. Moreover, it highly reduces the volume of the memory in comparison to the controller with address converter (by 50%).

The gain is especially high in case of controllers with bigger memories (e.g. big_mem, contr_14, testAW_05). Performed investigations also show, that proposed method gives always better results than simple realization of the control unit as a microprogrammed controller with address converter.

Conclusions

In the paper a method of reduction of microprogrammed controllers memory was presented. The idea was based on the two-stage reduction. At the beginning an additional block (address converter) is applied to the structure of a controller. Next, the hypergraph theory is used for further reduction of the memory.

The results of experiments proves very high effectiveness of the proposed method. The volume of the memory is reduced by 90% compared to the traditional realization of a controller with sharing codes. Moreover, application of the hypergraph theory permits to reduce the volume of the memory by 50% in comparison to the control unit with address converter.

REFERENCES

[1] Adamski M., Barkalov A., Architectural and Sequential Synthesis of Digital Devices, *University of Zielona Góra Press*, Zielona Góra, 2006

[2] Baranov S. I., Logic Synthesis for Control Automata, *Kluwer Academic Publishers*, Boston, MA, 1994

[3] Barkalov A., Titarenko L., Logic synthesis for FSM-based control units, *Lecture Notes in Electrical Engineering, Springer-Verlag*, Berlin, 53 (2009)

[4] Berge C., Graphs and Hypergraph, *North-Hols.r Mathematical Library*, Amsterdam (1976)

[5] De Micheli G., Synthesis and Optimization of Digital Circuits, *McGraw-Hill*, New York, NY, 1994

[6] Eiter T., Gottlob G., Identifying the Minimal Transversals of a Hypergraph and Related Problems, *SIAM Journal on Computing*, Vol.24, 1278-1304

[7] Gajski D., Principles of Digital Design, *Prentice Hall*, Upper Saddle River, NJ, 1996

[8] Kania D., The Logic Synthesis for the PAL-based Complex Programmable Logic Devices, *Lecture Notes of the Silesian University of Technology* (in Polish), Gliwice, 2004

[9] Łuba T., Synthesis of Logic Devices, *Warsaw University of Technology Press* (in Polish), Warsaw, 2005

[10] Maxfield C., The Design Warrior's Guide to FPGAs, *Academic Press*, Inc., Orlando, FL, 2004

[11] Sentovich E.M., Sequential Circuit Synthesis at the Gate Level, Ph.D. thesis. Chair-Robert K. Brayton, 1993

[12] Węgrzyn A., Węgrzyn M., A new approach to simulation of concurrent controllers, In Design of embedded control systems, ISBN: 0-387-23630-9, 95–108. Springer, New York, 2005

[13] Wiśniewska M., Application of hypergraphs to the decomposition of the discrete-systems, PhD thesis, University of Zielona Góra, 2011 (in Polish)

[14] Wiśniewska M., Wiśniewski R., Adamski M., Reduction of the microinstruction lenght in the designing process of microprogrammed controllers, Electrical Review, 85(7), 203–206, 2009

[15] Wiśniewski R., Synthesis of compositional microprogram control units for programmable devices, Lecture Notes in Control and Computer Science, 14 (2009), *University of Zielona Góra Press*, Zielona Góra

***Autorzy**: prof. dr hab. inż. Marian Adamski, Uniwersytet Zielonogórski, Instytut Informatyki i Elektroniki, ul. Licealna 9, 65-417 Zielona Góra, E-mail: M.Adamski@iie.uz.zgora.pl; dr inż. Monika Wiśniewska, Instytut Informatyki i Elektroniki, ul. Licealna 9, 65-417 Zielona Góra, E-mail: M.Wisniewska@weit.uz.zgora.pl; dr inż. Remigiusz Wiśniewski, Instytut Informatyki i Elektroniki, ul. Licealna 9, 65-417 Zielona Góra, E-mail: R.Wisniewski@iie.uz.zgora.pl; mgr inż. Łukasz Stefanowicz, Instytut Informatyki i Elektroniki, ul. Licealna 9, 65-417 Zielona Góra, E-mail: L.Stefanowicz@weit.uz.zgora.pl.*