Marian ADAMSKI, Alexander BARKALOV, Remigiusz WIŚNIEWSKI, Jakub LIPIŃSKI

Uniwersytet Zielonogórski

Reduction of the Memory Size in the Microprogrammed Controllers

Abstract. *The method of reduction of the control memory size in the microprogrammed controllers is proposed in the article. The idea is based on the hypergraph theory. The concurrent microoperations are encoded together thus the total volume of the memory is reduced. In order to receive the proper microinstruction, an additional module – microinstruction decoder is also prepared. The idea of the proposed method is illustrated by an example. Moreover, the result of performed experimental investigations is presented, as well.*

Streszczenie. *W artykule zaproponowano metodę redukcji pojemności pamięci sterowników mikroprogramowanych. Metoda bazuje na teorii hipergrafów. Mikrooperacje parami kompatybilne są kodowane wspólnie, dzięki czemu redukcji ulega całkowita pojemność pamięci sterownika mikroprogramowanego. Do struktury układu wprowadzono dodatkowy moduł, dekodera mikroinstrukcji. Jednostka ta jest odpowiedzialna za odkodowanie pierwotnych danych. Idea proponowanej metody zilustrowano przykładem. Ponadto, przeprowadzono także badania eksperymentalne, których celem była weryfikacja skuteczności proponowanej metody. Wyniki badań pokazują, że pierwotna pamięć sterownika jest redukowana średnio o 21%. (**Redukcja pojemności pamięci w sterownikach mikroprogramowanych**).*

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

Introduction

A control unit (CU) is important part of a digital system [1,2,3,4,5,6]. Usually, the control unit is realized as a finite state machine (FSM) [4,6,7,8]. However, in the case of the linear flow-chart, the microprogrammed controller may require less amount of hardware than control unit based on the traditional FSM model [9,10,11]. In case of microprogrammed controllers, the control unit is decomposed into two main parts. The first one addresses microinstructions [7,9], while the second is in response of holding and generating the proper microinstruction [10,12].

Such a solution leads to the reduction of the number of logic elements that are required for implementation of the controller [12]. Typically, the control memory is implemented as a ROM or RAM memory. Thus, wider areas of the destination device can be used for other modules of the prototyped system [1,4,5,7,10,13].

Most of controllers (especially realized as a Complex Instruction Set Computers, CISC) have a long microinstruction width what influences on the memory size [3,5]. Such a situation causes serious problems in the prototyping process. In case of System-On-Programmable-Chip (SoPC), the memory can be implemented with dedicated memory blocks of the Field Programmable Gate Arrays (FPGA). However, if the microinstruction length exceeds the total length of the dedicated memory block of an FPGA, the memory has to be decomposed. On the other hand, in case of controllers implemented as a System-On-Chip (SoC), the memory is designed as an independent module. This means that each additional bit in the microinstruction width increases the total size of the memory and increases the cost of the whole device.

In the paper we propose the method of the control memory reduction. The idea is based on the reduction of the microinstruction length by encoding the concurrent microoperations together. To achieve it, the hypergraph theory is applied [11,14]. Moreover, the particular stages of the reduction process also are performed with hypergraphs. Finally, the initial memory is reduced, while the proper microinstructions are decoded by an additional block of microinstruction decoder.

Microprogrammed controllers

The typical microprogrammed controller is presented in the Figure 1. The controller can be divided into three main

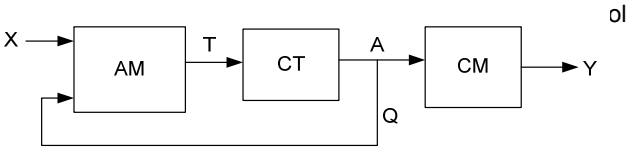

Fig. 1. The microprogrammed controller

The addressing module generates the proper excitation function for the counter:

$$T = f(X, Q) \tag{1}$$

The counter is in charge of holding the code of the current state of the controller. Additionally, it generates the microinstruction address. The main benefit of the realization of the control unit as the microprogrammed controller is a possibility of implementation of the circuit CM with embedded memories [12]. Other blocks of the prototyping system are implemented with the logic blocks (flip-flops and LUT elements) of the programmable device (like FPGA) [1,10,12]. Such an idea permits to reduce 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.

Presented system can be easily implemented with embedded devices as a System-On-a-Chip or System-On-a-Programmable-Chip. However, the size of the control memory may cause serious problems during the prototyping process. Most controllers have long microinstruction length, which influences on the total size of the memory.

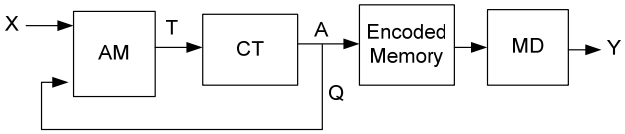

Fig. 2. An idea of the proposed method

Idea of the proposed method

To reduce the memory size, microinstructions that are pairwise compatible (can be executed concurrently) will be encoded together. Such an idea permits to reduce the total

memory size. However, an additional block, Microinstruction Decoder (MD) is also required. Figure 2 illustrates the idea of the proposed method.
The reduction method bases on the hypergraph theory, initially presented in [11,14,15]. The reduction idea was enhanced and adapted to the microprogrammed controllers. The whole process can be divided into the following steps:

1. Formation of the set of compatibility classes. Microoperations are compatible if they are not executed concurrently [14,16]. The result of this operation is the set $Cc=\{C_1,...,C_K\}$, that contains all compatibility classes. This problem is related to the hypergraph complement. The initial memory is formed as a hypergraph H, which vertices refer to microoperations, and hyperedges represent microinstructions of the control memory. The main problem of the compatibility classes formation is computational complexity, which can be exponential in case of exact algorithms [14,16]. Therefore, very often approximate methods are applied [14,16]. The calculation of the hypergraph complement leads to the new hypergraph H_C. It represents relations between compatibility classes and microoperations. Each hyperedge refer to the compatibility class, while vertices represent microoperations.

2. Determination of the weight (cost) of each compatibility class. A weight (cost) of a compatibility class C_i is equal to the minimum number of bits required for its encoding [14,11]. The compatibility classes are encoded with the natural binary code:

$$L_i = \lceil \log_2(|C_i|+1) \rceil \tag{2}$$

An additional bit is required for representation the state where no microoperation is executed.

3. Calculation of the hypergraph dual to the hypergraph H_C. In this step the dual hypergraph H_D is formed. In practice this operation can be very easily done by transposition of the incidence matrix of an initial hypergraph. Vertices of hypergraph H_D describe compatibility classes and hyperedges refer to microoperations of the initial memory.

4. Determination of the minimum vertex covering (transversal) of hypergraph H_D. At this stage the smallest transversal τ is calculated. Since this problem can be exponential for exact algorithms, approximate methods ought to be applied [11,13,14,16]. There is a possibility of obtaining more than one smallest transversal [14]. Therefore, calculation of the smallest weight is also required, to choose the best solution.

5. Calculation of the total cost of each minimum covering and reduction of redundant microoperations. To determine the best solution, for each minimum transversal the total covering cost is calculated:

$$W_S = \sum_{1}^{I} L_i \tag{3}$$

where W_S denotes the total cost of transversal τ_S and it is equal to the sum of weights L_i of all compatibility classes that belong to this cover. If any microoperation belongs to more than one class, it ought to be reduced. This operation is executed primarily for classes, which total weights can be reduced. Finally, each microoperation belongs to one compatibility class. For further analysis, the transversal of the smallest total cost is selected.

6. Encoding compatibility classes which realize minimal transversal. At this stage the compatibility classes are encoded with variables $Q=\{q_1,...,q_{Ws}\}$. The number of required $|Q|$ bits is equal to the total weight of transversal τ_S.

7. Determination of a new content of memory with encoded compatibility classes. The content of the memory is determined through concatenation of all variables obtained in the previous stage. Each new microinstruction is formed as a concatenation of Q codes of encoded classes. Therefore, the width of a new microinstruction is equal to W_S, while the reduction of the control memory size can be calculated as [16]:

$$t = \left(1 - \frac{\sum_{i=1}^{I} L_i}{N}\right) \cdot 100\% \tag{4}$$

where: t – percentage reduction of the memory, I – number of classes realized smallest transversal, L_i – weight of the i-th class that belongs to the transversal, N – primary size of a microinstruction.

8. Formation of the module of Microinstruction Decoder. In the final step the module MD is formed. Such a realization is trivial and may be easily done. An example of microinstruction decoder realization in Verilog hardware description language is shown in the next section.

Finally, the whole system can be designed. Depending on the designers demands, proper modules can be implemented in various ways [3,4,12,17,18]. For example the addressing module, the counter and the microinstruction decoder may be realized with Look-Up Tables (LUTs), while Encoded Memory can be implemented with dedicated memory blocks of a destination reprogrammable device [5,14].

Example of the proposed method

The presented idea will be illustrated by an example. Let's assume the hypothetical microcontroller, that executes N=6 microoperations, that are grouped into M=4 microinstructions. The total size of the control memory equals to V_{init}=6*4=24 bits. Table 1 presents the content of the control memory Mem_1.

Table 1. Exemplary memory Mem_1

Microinstruction (μ_M)	Microoperation					
	y_0	y_1	y_2	y_3	y_4	y_5
μ_1	0	1	0	0	0	1
μ_2	0	1	0	1	0	0
μ_3	1	0	0	0	1	0
μ_4	0	0	1	0	1	0

In the first step, the set of compatibility classes is calculated. To achieve it, the initial memory is formed as a hypergraph. Its vertices refer to microoperations, while its edges – to microinstructions. Now, the set of compatibility classes can be obtained via computation of the hypergraphs complement H_C. Table 2 shows the result of such an operation. There are K=4 compatibility classes. The first one contains elements y_0, y_1 and y_2, which means that those operations do not occur concurrently in any of four microinstructions. The second class contains microoperations y_1 and y_4, third y_0, y_2, y_3, y_5 while the last one y_3, y_4 and y_5.

Table 2. The obtained set of compatibility classes for the memory *MemInit*

Compatibility Class (C_K)	Microoperation					
	y_0	y_1	y_2	y_3	y_4	y_5
C_1	1	1	1	0	0	0
C_2	0	1	0	0	1	0
C_3	1	0	1	1	0	1
C_4	0	0	0	1	1	1

Next, weights for all compatibility classes ought to be computed. In the presented example the particular classes take the following costs:

- $L_1 = \lceil \log_2(3+1) \rceil = 2,$
- $L_2 = \lceil \log_2(2+1) \rceil = 2,$
- $L_3 = \lceil \log_2(4+1) \rceil = 3,$
- $L_4 = \lceil \log_2(3+1) \rceil = 2,$

At the third stage, the hypergraph H_D dual to the H_C is calculated. The edges of H_D correspond to the vertices of H_C, while its vertices refer to the edges of H_C (Tab. 3).

Table 3. The incidence matrix of hypergraph H_C

Microoperation	Compatibility Class			
	C_1	C_2	C_3	C_4
y_0	1	0	1	0
y_1	1	1	0	0
y_2	1	0	1	0
y_3	0	0	1	1
y_4	0	1	0	1
y_5	0	0	1	1

Next, the minimum vertex covering is calculated. For the hypergraph H_C there are two solutions. The first one τ_1 consists of the sets C_1 and C_4, while the second transversal τ_2 realizes the cover with the use of C_2 and C_3. To select the best solution, the total cost for each minimum covering has to be obtained. For the first transversal it is equal to $W_1=L_1+L_4=2+2=4$, while for the second one the total weight equals to $W_2=L_1+L_4=5$. There are no redundant microoperations, because in both transversals each microoperation belongs to only one class. Therefore, transversal τ_1 is selected for the further analysis.

At the subsequent stage, the compatibility classes are encoded. Since the total weight for the transversal equals to $W_1=4$, there are |Q|=4 variables required. Table 4 presents the result of encoding, where natural binary code was used.

Table 4. Encoding of compatibility classes

Class C_1			Code		Class C_4			Code	
y_0	y_1	y_2	q_1	q_2	y_3	y_4	y_5	q_3	q_4
0	1	0	0	0	0	0	1	0	0
0	1	0	0	0	1	0	0	0	1
1	0	0	0	1	0	1	0	1	0
0	0	1	1	0	0	1	0	1	0

Based on the encoding presented in the Table 4, the content of the reduced control memory is determined. Table 5 shows the content of the Mem_1 after the reduction.

Table 5. Content of the memory Mem_1 after the reduction

Microinstruction (μ_M)	Microinstruction			
	q_1	q_2	q_3	q_4
μ_1	0	0	0	0
μ_2	0	0	0	1
μ_3	0	1	1	0
μ_4	1	0	1	0

At the last stage, the microinstruction decoder has to be prototyped. An exemplary description in Verilog language of such a module is presented in the Figure 3.

```
module MicroinstructionDecoder (y,q);
   output [1:6] y;
   input [1:4] q;

   //decoding of microinstructions:
   assign y[1]=~q[1]&q[2];
   assign y[2]=~q[1]&~q[2];
   assign y[3]=q[1]&~q[2];
   assign y[4]=~q[3]&q[4];
   assign y[5]=q[3]&~q[4];
   assign y[6]=~q[3]&~q[4];
endmodule
```

Fig. 3. Przykładowy opis dekodera mikroinstrukcji w języku Verilog

Finally, the whole system can be implemented. Since the encoded microinstruction consists of |Q|=4 variables (encoded microoperations). Therefore, the total size of the new control memory is equal to $V_{reduced}$=16 bits. It means, that the original size of the control memory was reduced by t=33%.

The results of experiments

The effectiveness of the discussed method has been verified experimentally. The library of test modules consists of over 50 benchmarks. All of examined memories were real memory systems of logic controllers and they were taken from specifications and benchmarks presented in [7,9,10,11,12,14,19].

For each benchmark, the full reduction process was performed. First, the memory was described with the hypergraph. Next, compatibility classes were determined via calculation of the hypergraph complement. In the subsequent steps, weight of each class was calculated, dual hypergraph obtained and its minimal covering computed. If any of microoperations belong to more than one classes that comprise the transversal, they are reduced to achieve the lowest total cost. After selection of the transversal with the lowest cost, initial memory was encoded. To verify the functional correctness of the reduced memory, ten randomly selected benchmarks were designed with Verilog language. Next, the functional simulation was executed. For all examined benchmarks, either initial and reduced memory have the proper results, which confirms the correctness of the performed reduction.

Finally, the size of the encoded memory was compared with the initial one. Table 6 shows the results of experiments, where representative benchmarks (memories) are presented.

Particular columns contain the following data:

- benchmark – specifies the name of test module;
- size of the initial memory – the initial size of the test module memory;
- size of the reduced memory – the capacity of the memory after the reduction;

- density of the memory – the number of microoperations that are equal to “1” in relation to the total size of the memory (expressed in percentage terms);
- reduction of the memory – the size of the memory after reduction in relation to the initial capacity memory (expressed in percentage terms.

The average results refer to all benchmarks in the library tests.

Table 6. Results of experiments

Benchmark	Size of the initial memory [bits]	Size of the reduced memory [bits]	Density of the memory	Reduction of the memory [%]
Test022	264	154	19,31%	42%
Test021	176	112	20,45%	36%
TestMK_06	234	162	28,63%	31%
TestMK_10	180	135	27,22%	25%
Test016	25	20	28,00%	20%
Test032	168	140	41,66%	17%
Test012	60	55	71,66%	8%
Average	**72,93**	**57,58**	**35,66%**	**21%**

From above tables we can see that application of the proposed method permits to reduce the volume of the memory on average by 21%. All of the examined memories were reduced (the lowest reduction was equal to 8%, while the highest – 42%). It means, that the size of reduced memory in microprogrammed controllers is always smaller than the initial one.

It is worth to notice, that effectiveness of the reduction process strongly depends on the density of the initial memory. Results of experiments have shown that reduction is especially high in case of density lower than 30% of the initial memory. For example the memory of *Test022* can be reduced over than 40%. On the other hand, there are controllers which memories are hard to reduce with higher density, and the reduction may be less than 10%.

Conclusions

The method of reduction of microprogrammed controllers memory was presented in the paper. The idea was based on the hypergraph theory. An initial memory is represented by a hypergraph. Further computations lead to the reduced memory, where microoperations are encoded. The results of experiments have shown, that effectiveness of the proposed method strongly depends on the density of the initial memory.

REFERENCES

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

[2] Doligalski M., Adamski M., Hierarchical configurable Petri net modeling in VHDL, *International Journal of Electronics and Telecommunications*, 58 (4), (2012), pp. 397-402

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

[4] 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)

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

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

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

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

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

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

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

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

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

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

[15] Adamski M., Wiśniewska M., Wiśniewski R., Stefanowicz Ł., Application of hypergraphs to the reduction of the memory size in the Microprogrammed Controllers with Address Converter, *Electrical Review*, 8 (2012), 134–136

[16] Robertson E.L., Microcode bit optimization is NP-complete, *IEEE Trans. Comput.*, C-28 (1979), pp. 316–319

[17] Grobelna I., Formal verification of embedded logic controller specification with computer deduction in temporal logic, *Electrical Review*, 12a (2011), 40-43

[18] Milik A., Hrynkiewicz E., Synthesis and implementation of reconfigurable PLC on FPGA platform*, International Journal of Electronics and Telecommunications*, 58 (1) (2012), 85–94

[19] Kołopieńczyk M., Application of adress converter for decreasing memory size of CMCU with code sharing, *Lecture Notes in Control and Computer Science*, Vol. 12. *University of Zielona Góra Press*, Zielona Góra, (2008)

***Authors**:*

prof. dr hab. inż. Marian Adamski
Profesor zwyczajny, zatrudniony w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego. Zainteresowania badawcze obejmują projektowanie systemów cyfrowych realizowanych w postaci mikrosystemów cyfrowych oraz formalnych metod programowania sterowników logicznych.
e-mail: M.Adamski@iie.uz.zgora.pl

Prof. dr hab. inż. Alexander Barkalov
Prof. Alexander A. Barkalov w latach 1976-1996 był pracownikiem Instytutu Cybernetyki im. V.M. Glushkova w Kijowie, gdzie uzyskał tytuł doktora habilitowanego ze specjalnością informatyka. W latach 1996-2003 pracował jako profesor w Instytucie Informatyki Narodowej Politechniki Donieckiej. Od 2004 pracuje jako profesor w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego.
e-mail: a.barkalov@iie.uz.zgora.pl

dr inż. Remigiusz Wiśniewski
Absolwent Uniwersytetu Zielonogórskiego, pracę doktorską obronił w 2008 roku. W latach 2000-2001 dwukrotnie odbył przemysłową praktykę studencką w firmie Aldec Inc. w Stanach Zjednoczonych. Aktualnie pracuje jako adiunkt (Uniwersytet Zielonogórski). Zainteresowania badawcze obejmują zagadnienia z zakresu teorii grafów i hipergrafów, bezpieczeństwa danych i kryptologii oraz metodologii projektowania i implementacji systemów cyfrowych.
e-mail: R.Wisniewski@iie.uz.zgora.pl

mgr inż. Jakub Lipiński
Absolwent Uniwersytetu Zielonogórskiego, pracę magisterską obronił w 2009 roku. Jest słuchaczem studiów doktoranckich, specjalność informatyka. Aktualnie pracuje w Szpitalu Wojewódzkim SPZOZ w Zielonej Górze. Zainteresowania badawcze obejmują zagadnienia z zakresu teorii grafów i hipergrafów.
e-mail: J.Lipinski@weit.uz.zgora.pl