The Pseudo LRU Hardware Complexity Decreasing for Associative Cache Memory and Translation Look-a-Side Buffer

Vadim Puidenko, Vyacheslav Kharchenko

***Abstract*—** the synthesis of the synchronous digital automaton with the logic of the pseudo LRU algorithm is carried out taking into the account events of hits or misses inside the associative cache memory or translation look-a-side buffer. The synchronous digital automaton model with the controlling logic for management substitution of data elements at the full reliability of the selected data's multitude is described. The minimizing for the switching functions that are simple completely defined and composite not completely defined are carried out. They are switched as: $\boldsymbol{L = \lambda(B)}$ selecting multiple among reliable values and $\boldsymbol{B^+ = f\left(B, \lambda(B)\right)}$ forming the values of bits for unit LRU considering the previous state. As a result the minimum discrete realizations have been obtained by suggested hardware solutions substitution policy for the algorithm of pseudo LRU of associative cache memory and associative translation look-a-side buffer.

***Keywords*—**algorithm pseudo LRU, controlling logic, associative memory cache, associative translation look-a-side buffer, LRU unit.

I. INTRODUCTION

The processor core contains of microarchitecture components such as internal associative cache and associative look-a-side buffer. Each of these components of the microarchitecture processor core has the specialized unit called LRU (Least Recently Used), in which is embedded the hardware of the substitution policy algorithm for data unit elements. This substitution policy algorithm called pseudo LRU and represents the logic for replacement an element in a multitude of data unit. The LRU unit has a significant impact on characteristics of high-speed performance, complexity and energy consumption for integrated devices. There are plural research works have description on replacement of elements according policy substitution LRU, including algorithm pseudo LRU with defined conclusions about advantages of this substitution policy, namely: the best productivity as compared to the productivity of algorithms of casual substitution FIFO and LRU [1,2], high efficiency realization of LRU [3], the optimal policy of substituting for pseudo LRU [4-7] accompanied with general algorithm, structural, functional schemes and program VHDL codes for the unit of LRU controller, where attention is not paid for simplifying and evaluating hardware for algorithm realization.

This paper adds previous work [8]. The objective is to describe hardware solutions and results of minimizing both the associative memory cash and the associative translation look-a-side buffer and assessing of complexity.

II. ARCHITECTURE OF THE ASSOCIATIVE MEMORY CACHE AND TRANSLATION LOOK-A-SIDE BUFFER

A. Architecture of the associative memory cache

Internal cache memory of level L1 [1] (Fig. 1) stores the copies of the instructions, operands or data after ending cached package cycles of processor core. It is possible to apply to cache memory at every synchronization time. It works with physical addresses, that minimizes a number of times clearing of cache memory. Cache memory has 8 directed associative by a

V. O. Puidenko Kharkiv Radio Technical College, Kharkiv, Ukraine (e-mail: VAPuydenko@gmail.com).
V. S. Kharchenko National Aerospace University «Kharkiv Aviation Institute», Kharkiv, Ukraine (e-mail: V.Kharchenko@csn.khai.edu.).

multitude organization. The unit of data of 32 Kbytes is divided into eight directions (Fig. 1) each of them has 64 64-bytes multitudes or lines of cache memory. Addressing of cache memory is carried out by a division low 36 bits of physical address on three parts. Six bits of the field of index determine the number of multitude from 64. High 24 bits are the field to the tag (signs): these bits are compared to the tags of each line of the index multitude and show whether the 64-bytes line of cache memory is kept according to the physical address. The low six bits of physical address elect a byte inside of the line of cache memory. The field of six bits, that is located in the unit of reliability, shows whether the data are present in current time period according to certain physical address, they are cached.

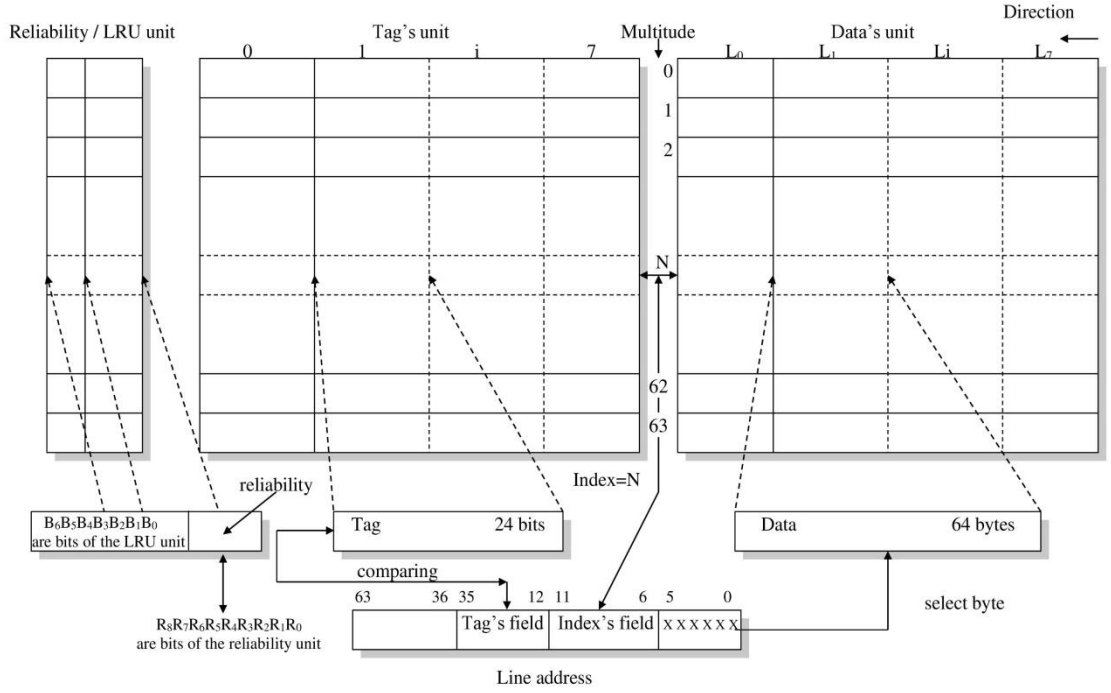

Fig. 1 Architecture of the internal associative memory cache

B. Architecture of the associative translation look-a-side buffer

The device of paging translation provides access to the structures of the data, they are kept partly in basic memory and partly on a disk. The associative buffer of TLB (Translation Look-a-side Buffer) includes the device of paging translation, it keeps 32 page table entry elements, that have long been used. The structures of data of buffer of TLB are shown in Fig. 2. The unit of paging translations looks over linear addresses in TLB. If it does not find a linear address in TLB, a unit forms a request for loading in TLB of the correct physical address from the table of pages in basic memory.

Only when a correct element is in TLB, the bus cycle will be initiated. When the unit of paging translation represents a page from linear address space on a page in physical memory, it changes only low 20 bits of linear address. Low 12 bits of physical address are taken invariable from the linear address. In the typical systems TLB satisfies to 99% requests on access to the tables of pages. As strategy of substitution the same algorithm of pseudo LRU is used in the buffer of TLB, as well as in an internal cache memory. The buffer of TLB is cleared up at loading a basic register CR3 of the catalogue pages.

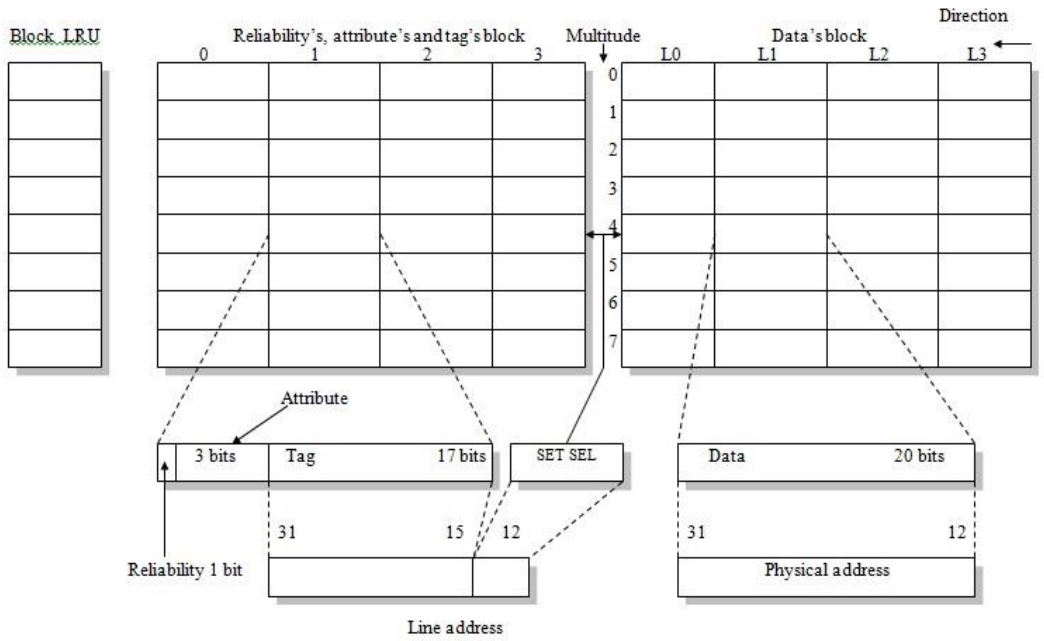

Fig. 2 Architecture of the associative translation look-a-side buffer

III. SYNTHESIS OF THE AUTOMATA FOR SUBSTITUTION POLICY BY THE ALGORITHM PSEUDO LRU

A. The algorithm pseudo LRU of the substitution policy for the associative translation look-a-side buffer to 4 directions

Let's identify lines in multitude through L_0, L_1, L_2 and L_3. Every multitude in the unit LRU corresponds to three bits of B_0, B_1 and B_2, that are modified at the every hits or miss as follows (Fig. 3) :

- if the last access in multitude is to the line of L_0 or L_1, then bit is B_0=1, else to the line of L_2 or L_3 - B_0=0;
- if last access in the pair of L_0 - L_1 is to the line of L_0, then bit is B_1=1, else to the line of L_1 – B_1=0;
- if last access in the pair of L_2 - L_3 is to the line of L_2, then bit is B_2=1, else to the line of L_3 – B_2=0;

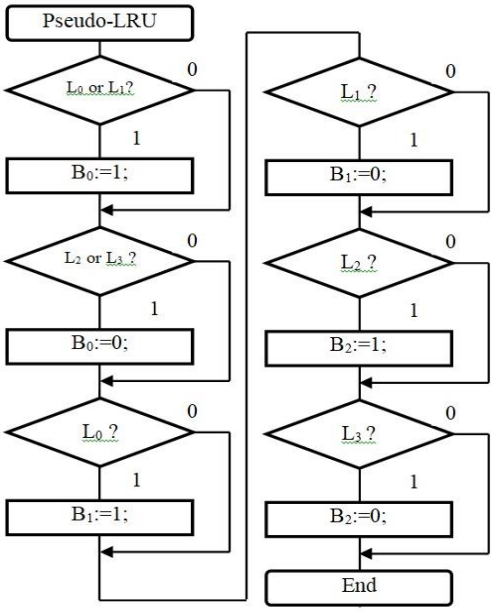

Fig. 3 The algorithm pseudo LRU of the substitution policy for the associative translation look-a-side buffer to 4 directions

B. The logical model

The minimal logical equations (1) – (7) of simple completely defined and composite not completely defined switching functions $\boldsymbol{L = \lambda(B)}$ and $\boldsymbol{B^+ = f(B, \lambda(B))}$ are describe the discrete mathematical model of the LRU unit for all multitudes of a data unit:

$$L_0 = \overline{B}_1 \& \overline{B}_0 \quad (1)$$

$$L_1 = B_1 \& \overline{B}_0 \quad (2)$$

$$L_2 = \overline{B}_2 \& B_0 \quad (3)$$

$$L_3 = B_2 \& B_0 \quad (4)$$

$$B_0^+ = \overline{\overline{\overline{L}_3 \& \overline{L}_2 \& B_0} \& \overline{L}_1 \& \overline{L}_0} \quad (5)$$

$$B_1^+ = \overline{\overline{\overline{L}_1 \& B_1 \& \overline{L}_0}} \quad (6)$$

$$B_2^+ = \overline{\overline{L}_3 \& B_2 \& \overline{L}_2} \quad (7)$$

C. The hardware solution

The hardware solution for the substitution policy implementation by the algorithm pseudo LRU is presented on Fig.4.

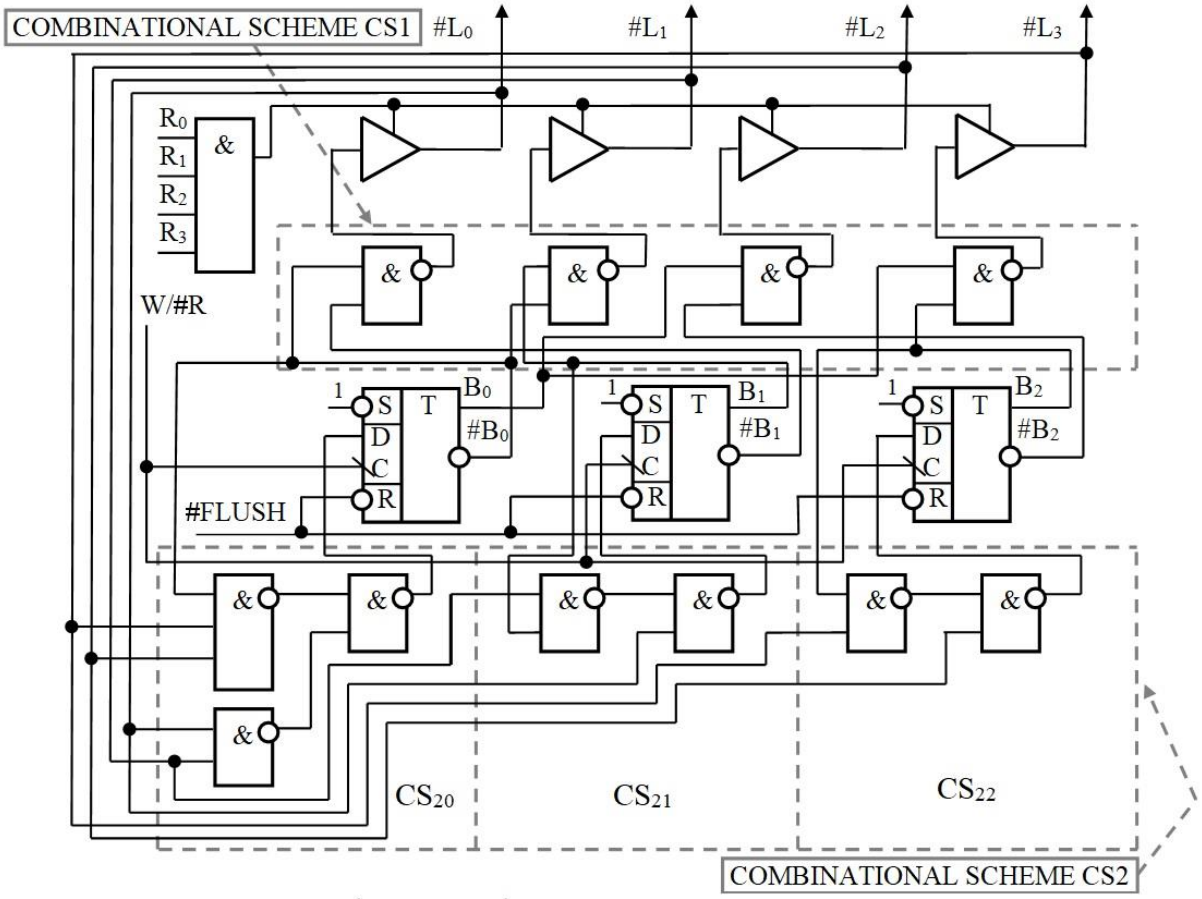

Fig. 4 The algorithm pseudo LRU of the substitution policy for the associative translation look-a-side buffer to 4 directions

D. The computer model

The computer model is built to produce the minimized hardware for further research in the environment of computer modelling (Fig.5):

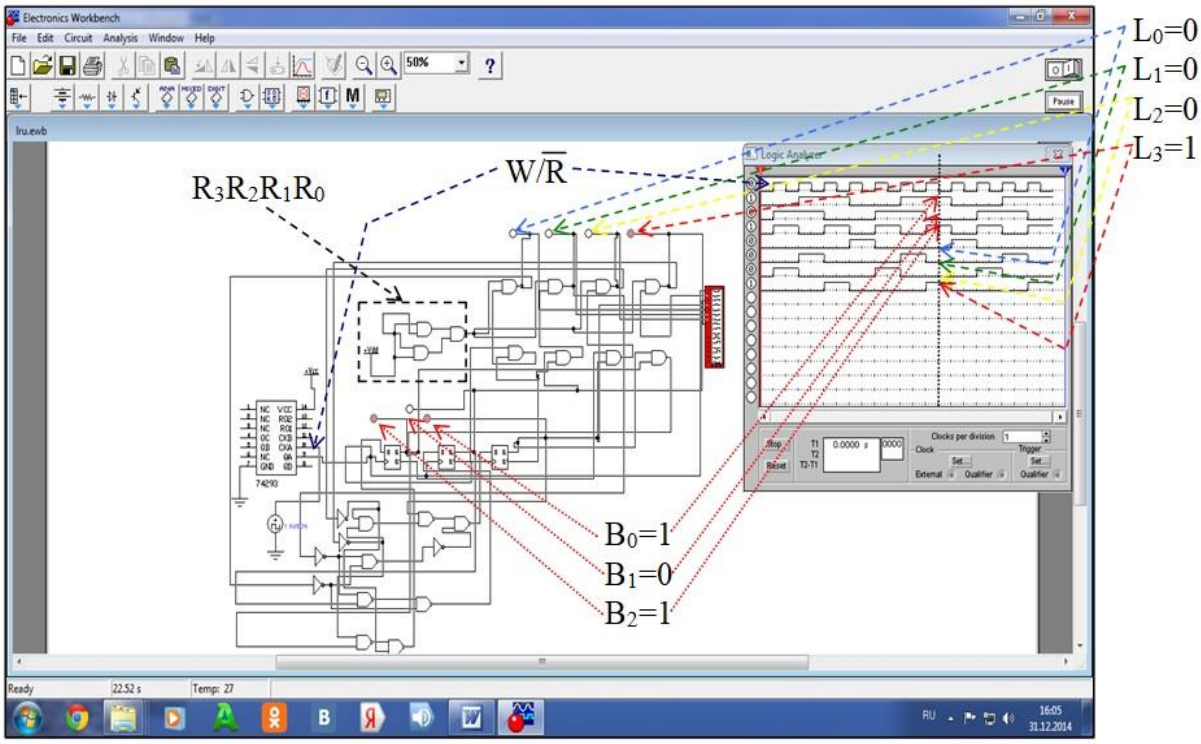

Fig. 5 The computer model for the minimum hardware solution of the substitution policy implementation by the algorithm pseudo LRU for the LRU unit of associative translation look-a-side buffer to 4 directions

The researching diagrams of time and values of the logic analyzers of the computer model are shown the sequence of bit's values transitions B_2, B_1, B_0: 000, 011, 110, 101 as corresponding to the algorithm pseudo LRU.

E. The algorithm pseudo-LRU of the substitution policy for the internal associative cache memory to 8 directions

Let's identify lines in multitude through L_0, L_1, L_2, L_3, L_4, L_5, L_6 and L_7. Every multitude in the Unit LRU corresponds to seven bits of B_0, B_1, B_2, B_3, B_4, B_5 and B_6, that are modified at every miss or filling as follows (Fig. 6):

- if the last access in multitude is to the line of L_0 or L_1 or L_2 or L_3, then bit is B_0=1, else to the line of L_4 or L_5 or L_6 or L_7 - B_0=0;
- if the last access in multitude is to the line of L_0 or L_1, then bit is B_1=1, else to the line of L_2 or L_3 - B_1=0;
- if the last access in multitude is to the line of L_4 or L_5, then bit is B_2=1, else to the line of L_6 or L_7 - B_2=0;
- if last access in the pair of L_0 - L_1 is to the line of L_0, then bit is B_3=1, else to the line of L_1 - B_3=0;
- if last access in the pair of L_2 - L_3 is to the line of L_2, then bit is B_4=1, else to the line of L_3 - B_4=0;
- if last access in the pair of L_4 - L_5 is to the line of L_4, then bit is B_5=1, else to the line of L_5 - B_5=0;
- if last access in the pair of L_6 - L_7 is to the line of L_6, then bit is B_6=1, else to the line of L_7 - B_6=0;

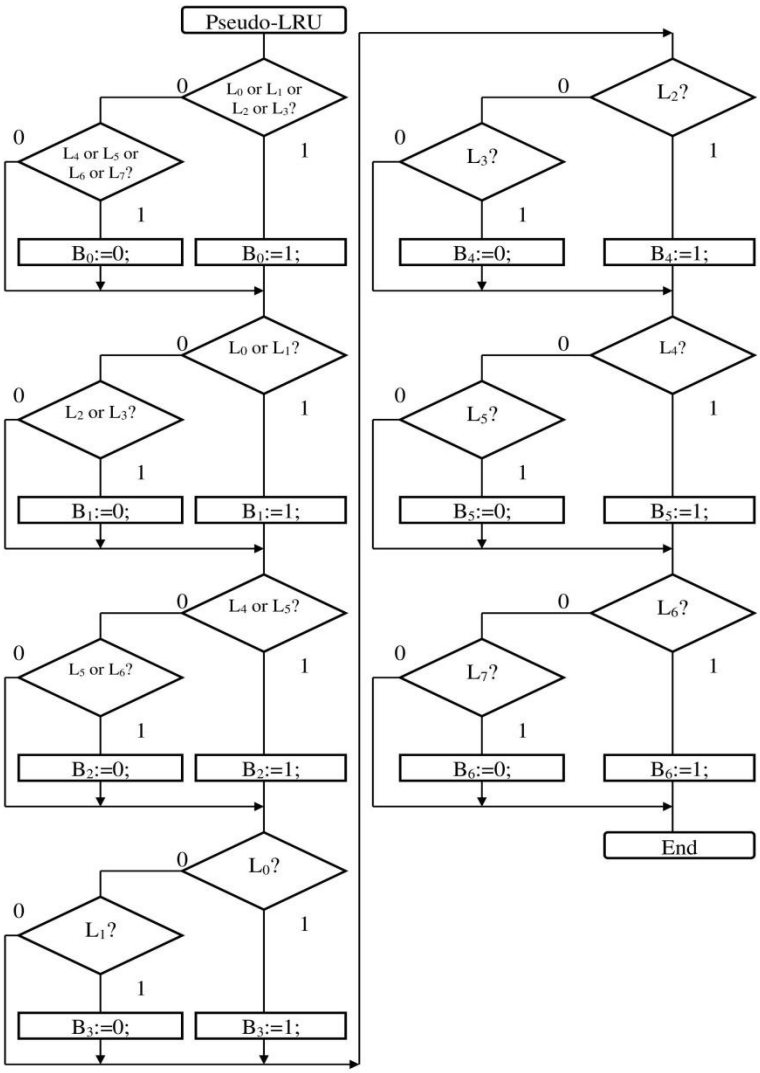

Fig. 6 The algorithm pseudo LRU of the substitution policy for the internal associative cache memory to 8 directions

F. The logical model

The minimal logical equations (8) – (22) of simple completely defined and composite not completely defined switching functions $\boldsymbol{L} = \boldsymbol{\lambda}(\boldsymbol{B})$ and $\boldsymbol{B}^{+} = \boldsymbol{f}\left(\boldsymbol{B}, \boldsymbol{\lambda}(\boldsymbol{B})\right)$ are describe the discrete mathematical model of the LRU unit for all multitudes of a data unit:

$$L_0 = \overline{B}_1 \& \overline{B}_0 \& \overline{B}_0 \quad (8)$$

$$L_1 = B_3 \& \overline{B}_1 \& \overline{B}_0 \quad (9)$$

$$L_2 = \overline{B}_4 \& B_1 \& B_0 \quad (10)$$

$$L_3 = B_4 \& B_1 \& \overline{B}_0 \quad (11)$$

$$L_4 = \overline{B}_5 \& \overline{B}_2 \& B_0 \quad (12)$$

$$L_5 = B_5 \& \overline{B}_2 \& B_0 \quad (13)$$

$$L_6 = \overline{B}_6 \& B_2 \& B_0 \quad (14)$$

$$L_7 = B_6 \& B_2 \& B_0 \quad (15)$$

$$B_0^{+} = \overline{\overline{L}_5 \& \overline{\overline{L}_7 \& \overline{L}_6 \& \overline{L}_4 \& B_0} \& \overline{L}_3 \& \overline{L}_2 \& \overline{L}_1 \& \overline{L}_0} \quad (16)$$

$$B_1^{+} = \overline{\overline{L}_3 \& \overline{L}_2 \& B_1 \& \overline{L}_1 \& \overline{L}_0} \quad (17)$$

$$B_2^{+} = \overline{\overline{L}_7 \& \overline{L}_6 \& B_2 \& \overline{L}_5 \& \overline{L}_4} \quad (18)$$

$$B_3^+ = \overline{\overline{\overline{L_1}\&B_3}\&\overline{L_0}} \quad (19)$$

$$B_4^+ = \overline{\overline{\overline{L_3}\&B_4}\&\overline{L_2}} \quad (20)$$

$$B_5^+ = \overline{\overline{\overline{L_5}\&B_5}\&\overline{L_4}} \quad (21)$$

$$B_6^+ = \overline{\overline{\overline{L_7}\&B_6}\&\overline{L_6}} \quad (22)$$

G. The hardware solution

The hardware solution for the substitution policy implementation by the algorithm pseudo LRU is presented on Fig. 7:

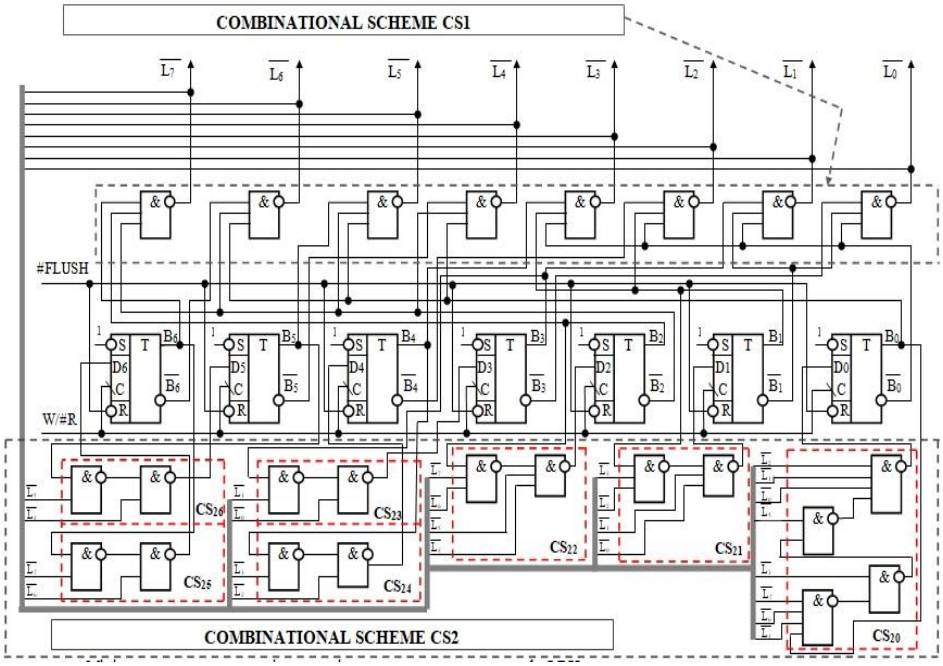

Fig. 7 The minimum hardware solution of the substitution policy implementation by the algorithm pseudo LRU for the LRU unit of associative internal cache memory to 8 directions

H. The computer model

The computer model is developed to produce the minimized hardware for further research in the environment of computer modelling (Fig. 8):

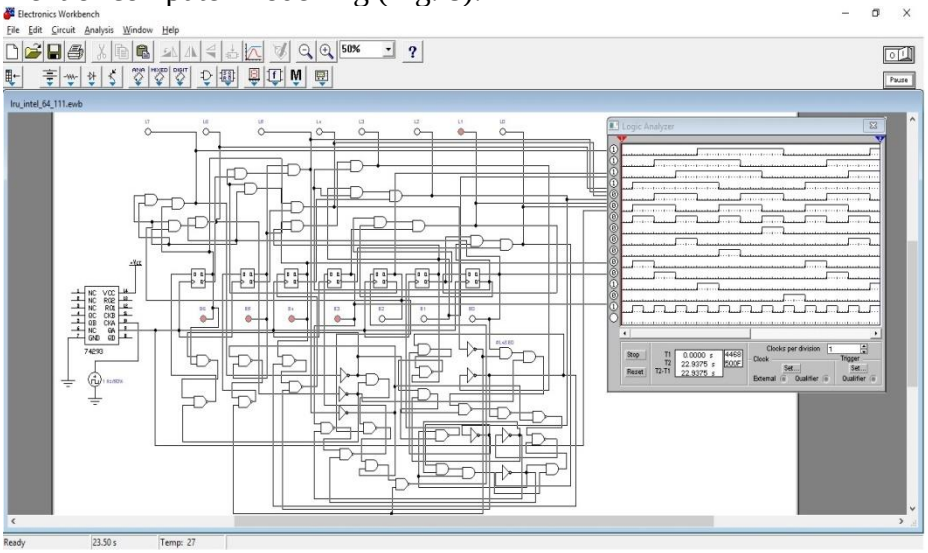

Fig. 8 The computer model for the minimal hardware solution of the substitution policy implementation by the algorithm pseudo-LRU for the LRU unit of associative cache memory to 8 directions

The researching diagrams of time and values of the logic analyzers of the computer model are shown the sequence of bit's values transitions B_6, B_5, B_4, B_3, B_2, B_1, B_0: 0000000, 0001011, 0101110, 0111101, 1111000, 1110011, 1010110, 1000101 as corresponding to the algorithm pseudo LRU.

IV. Assessment of hardware complexity

At the building of combinational schemes, the "method by Quine" gives an opportunity exactly to estimate corresponding expenses. It gives the total estimate of inputs number for all logical elements of digital combinational schemes built in the corresponding base:

$$K = \sum_{i=1}^{N} E_i \quad (23)$$

where: N —number of inputs according to the scheme that is estimated; $E_i=1$ — the direct input; $E_i=2$ — the inverse input.

A. Assessment of hardware complexity for LRU unit of associative translation look-a-side buffer

The combinational scheme CS_1 (Fig. 4) contains 4 logical elements of "and-not" for 3 inputs. The combinational scheme CS_2 consists of the combinational schemes CS_{20}, CS_{21}, CS_{22} (Fig. 4). The combinational schemes CS_{20}, CS_{21}, CS_{22} form a value for the informative inputs D of the synchronous D-triggers D_0, D_1, D_2 forming the composite switching functions of the renewal bits B_2, B_1, B_0. In the first approximation, the total formula of the Quine's estimation is as follows (24):

$$K = K_{CS1} + D + K_{CS2} \quad (24)$$

where: K_{CS1} — the Quine's estimation of the combinational scheme CS_1, D — the Quine's estimation of the memory's elements by type of the synchronous D-trigger, K_{CS2} — the Quine's estimation of the combinational scheme CS_2.

The general view of the Quine's estimation for the combinational schemes CS_1 is the following:

$$K_{CS1} = n * 2^n + 4 \quad (25)$$

where: n=2— number of inputs, 4 — accounting the inverse inputs-outputs.

The combinational scheme CS_{20} consists of consistently included the logical elements of base "and- not" for 2 and 3 inputs. Thus, the general view of the Quine's estimation for the combinational schemes CS_{20} will be the following:

$$K_{CS20} = n * 2^{n-1} + 2 + m * 2^{n-1} + 2 \quad (26)$$

where: n=2— number of inputs, m=3— number of inputs, 4 — accounting the inverse inputs-outputs.

The combinational schemes CS_{21} and CS_{22} are consist of the consistently included logical elements of base "and- not" for 3 inputs. Thus, the general view of Quine's estimation of combinational schemes CS_{21} and CS_{22} will be such:

$$K_{CS2[12]} = 2 * n * 2^{n-1} + 6 = n * 2^{n-1} + 6 \quad (27)$$

where: n=2— number of inputs, 6 — accounting the inverse inputs-outputs.

The general view of the Quine's estimation for the memory's elements by type of the synchronous D-trigger will be such:

$$D = m * 2^n + 3 = 2^p - 1 \quad (28)$$

where: p=4— number of inputs, 3 — accounting the inverse inputs-outputs.

Thus, the total Quine's estimation of the minimum hardware solution is the following:

$$K = n * (2^{n+1} + 2^{n-1}) + (m * 2^{m-3} + (m-1) * 2^m) + 2^p - 1 \quad (29)$$

The expressions (23) and (29) are identical and they produce the same result by the Quine's estimation. With expressions (23) the combinational schemes CS1 (Fig. 5) consist of the consistently including 4 logical elements of base "and-not" for 2 inputs, additional outputs – inputs of 4 units taking into account inverters. Thus, the Quine's estimation of the combinational scheme CS1 will be such:

$$K_{CS1} = 2 * 4 + 4 = 12$$

The combinational scheme CS2 (Fig. 4) consist next the logical elements of basis «and-not»: the logical elements of 6 units for 2 inputs, additional outputs – inputs of 8 units taking into account inverters, logical elements of 1 units for 3 inputs, additional outputs-inputs of 2 units taking into account inverters. Thus, the Quine's estimation of the combinational scheme CS2 will be such:

$$K_{CS2} = (2 * 6 + 8) + (3 * 1 + 2) = 25$$

The Quine's estimation for the memory's elements by type of the D-triggers with 3 units for 4 inputs, additional outputs-inputs of 3 units taking into account inverters will be such:

$$D = 4 * 3 + 3 = 15$$

The total Quine's estimation of the hardware solution of the substitution policy by the algorithm pseudo LRU will be such:

$$K = 12 + 25 + 15 = 52$$

The expression (29) has such argument as binary digit. On the basis of expression (29) the diagram on Fig. 9 allows to see increasing complexity in dependence from the arguments n=2, m=3 and see decreasing complexity in dependence from the arguments p=4

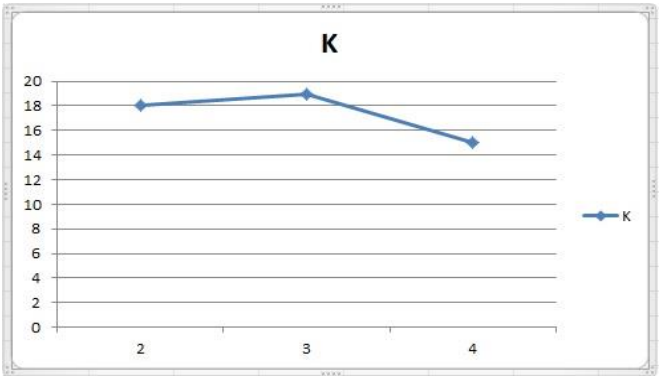

Fig. 9 Changing of the complexity depending on the arguments n=2, m=3, p=4

B. Assessment of hardware complexity for LRU unit of associative cache memory

The combinational scheme CS_1 (Fig. 7) contains 8 logical elements of "and-not" for 3 inputs. The combinational scheme CS_2 consists of the combinational schemes CS_{20}, CS_{21}, CS_{22}, CS_{23}, CS_{24}, CS_{25}, and CS_{26} (Fig. 7). The combinational schemes CS_{20}, CS_{21}, CS_{22}, CS_{23}, CS_{24}, CS_{25}, CS_{26} form a value for the informative inputs D of the synchronous D-triggers D_0, D_1, D_2, D_3,

D_4, D_5, D_6, forming the composite switching functions $\boldsymbol{B}^+ = \boldsymbol{f}\left(\boldsymbol{B}, \boldsymbol{\lambda}(\boldsymbol{B})\right)$ of the renewal bits B_6, B_5, B_4, B_3, B_2, B_1, B_0. The combinational schemes CS_{23}, CS_{24}, CS_{25}, CS_{26} consist of the consistently including the logical elements of base "and-not" for 2 inputs. In the first approximation, the total formula of the Quine's estimation is as follows (30):

$$K = K_{CS1} + D + K_{CS2} \tag{30}$$

where: K_{cs1} — the Quine's estimation of the combinational scheme CS_1, D — the Quine's estimation of the memory's elements by type of the synchronous D-trigger, K_{cs2} — the Quine's estimation of the combinational scheme CS_2

The general view of the Quine's estimation for the combinational schemes CS_1 is the following:

$$K_{CS1} = p * 2^p \tag{31}$$

where: p=3— number of inputs.

The general view of the Quine's estimation for combinational schemes CS_{23}, CS_{24}, CS_{25}, CS_{26} is the following:

$$K_{CS2[3456]} = n * 2^{2*n-1} + 4 \tag{32}$$

where: n=2— number of inputs, 4 — accounting the inverse inputs-outputs.

The general view of the Quine's estimation for the memory's elements by type of the synchronous D-trigger will be such:

$$D = 2^m + (m-1) * 2^{m-2} \tag{33}$$

where: m = 4 — number of inputs.

The combinational schemes CS_{21} and CS_{22} are consist of the consistently included logical elements of base "and- not" for 3 inputs. Thus, the general view of Quine's estimation of combinational schemes CS_{21} and CS_{22} will be such:

$$K_{CS2[12]} = p * 2^{p-1} + 2 \tag{34}$$

where: p=3 — number of inputs, 2 — accounting the inverse inputs-outputs.

The combinational scheme CS_{20} consists of consistently included the logical elements of base "and- not" for 2 and 4 inputs. Thus, the general view of the Quine's estimation for the combinational schemes CS_{20} will be the following:

$$K_{CS20} = n * 2^{n-1} + 3 + m * 2^{m-3} + 7 \tag{35}$$

where: n=2 — number of inputs, m=4— number of inputs, 3 — accounting the inverse inputs-outputs, 7 —accounting the inverse inputs-outputs.

Thus, the total Quine's estimation of the minimum hardware solution is the following:

$$K = [p * (2^p + 2^{p-1} + 1) + 2^p - (p-1)] + [n * (2^{2*n-1} + 2^{n-1}) + (2^{n+1} - 1)] + [m * 2^{m-3} + (2^{m-1} - 1) + (2^m + (m-1) * 2^{m-2})] \tag{36}$$

The expressions (30) and (36) are identical and they produce the same result by the Quine's estimation. With expressions (30) the combinational schemes CS1 (Fig. 7) consist of the consistently including 8 logical elements of base "and-not" for 3 inputs, additional outputs –

inputs of 7 units taking into account inverters. Thus, the Quine's estimation of the combinational scheme CS1 will be such:

$$K_{CS1} = 3 * 8 + 7 = 31$$

The combinational scheme CS2 (Fig. 7) consist next the logical elements of basis «and-not»: the logical elements of 10 units for 2 inputs, additional outputs – inputs of 7 units taking into account inverters, logical elements of 4 units for 3 inputs, additional outputs-inputs of 2 units taking into account inverters, the logical elements of 2 units for 4 inputs, additional outputs-inputs of 7 units taking into account inverters.

The Quine's estimation for the memory's elements by type of the D-triggers with 7 units for 4 inputs will be such:

$$D = 4 * 7 = 28$$

Thus, the Quine's estimation of the combinational scheme CS2 will be such:

$$K_{CS2} = (2 * 10 + 7) + (3 * 4 + 2) + (4 * 2 + 7) = 56$$

The total Quine's estimation of the hardware solution of the substitution policy by the algorithm pseudo LRU will be such:

$$K = 31 + 56 + 28 = 115$$

The expression (36) has such argument as binary digit. On the basis of expression (36) the diagram on Fig. 10 allows to see increasing complexity in dependence from the arguments n = 2, p = 3, m = 4

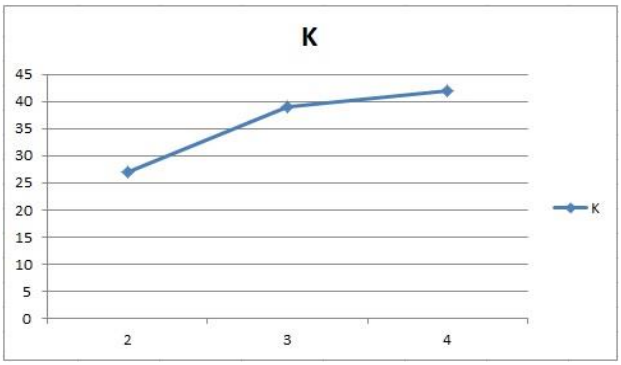

Fig. 10 Changing of the complexity depending on the arguments *n*=2, *m*=3, *p*=4

V. Conclusion

The minimized hardware solutions satisfies to the substitution policy with the pseudo LRU algorithm was researched in in the environment of the computer modelling, where received time diagrams are acknowledge correctness of functional work and therefore the success of the minimization synthesis.

The presented work describes synthesis of the minimal hardware for pseudo LRU algorithm, and advantage over program's VHDL code [2] according by the productivity:

- updating B bits in 6τ and one more τ for selecting corresponding the line of the multitude's data unit to 4 directions, where τ is the switching transition's process of the logical element base's "and-not".

- updating B bits in 8τ and one more τ for selecting corresponding the line of the multitude's data unit to 8 directions, where τ is the switching transition's process of the logical element base's "and-not".

The synthesized hardware of the pseudo LRU algorithm in basis "and-not" is considerably more simple unlike the realization hardware solutions LRU [3], where in particular certain such

complex digital devices are additionally involved, as counters, registers, comparators and they have negative effects on such features, as: simplistic implementation, productivity, reliability and the Quine's estimation. As for the Quine's estimation of the hardware solution, the diagram on Fig. 9 shows that the complexity of the hardware solution is linearly changing in dependence from the parameter of binary digit. As for the Quine's estimation of the hardware solution, the diagram on Fig. 10 shows that with increasing parameter of binary digit, the complexity of the hardware solution will increase linearly.

The following research will be dedicated to development of hardware for other algorithms such as MRU, LFU, ARC and their combining.

REFERENCES

[1] Vadym Puydenko, "The computer's model of the memory cache LRU unit of the processor's core of the architecture IA-32". 2018 International Conference Information Technologies and Computer modelling», Ivano-Frankivsk, Ukraine, May, 18-23, pp. 363-365.

[2] Safaa S. Omran, Ibrahim A. Amory, "Implementation of LRU Replacement Policy for Reconfigurable Cache Memory Using FPGA", 2018 International Conference on Advanced Science and Engineering, Kurdistan Region, Iraq, November, 12-14, pp.13-18.

[3] T.S.B. Sudarshan, Rahil Abbas Mir, S.Vijayalakshmi, "Highly Efficient LRU Implementations for High Associativity Cache Memory". Birla Institute of Technology and Science, Pilani, Rajasthan 330331 INDIA, 2017.

[4] Burhan Ul Islam Khan, Rashidah F. Olanrewaju, Roohie Naaz Mir, Abdul Raouf Khan, S. H. Yusoff A Computationally, "Efficient P-LRU based Optimal Cache Heap Object Replacement Policy". International Journal of Advanced Computer Science and Applications, Vol. 8, No. 1, 2017.

[5] Swadhesh Kumar and P K Singh. "An Overview of Modern Cache Memory and Performance Analysis of Replacement Policies". 2016 2nd IEEE International Conference on Engineering and Technology, India, pp. 4145-4148.

[6] Jaafar Alghazo, Adil Akaaboune, Nazeih Botros, "Cache Replacement Algorithm Records". 2004 International Workshop on Memory Technology, Design and Testing, Illinois, USA, August, pp. 19-24.

[7] J. Reineke, D. Grund, C. Berg, and R. Wilhelm, "Timing predictability of cache replacement policies," Real-Time Syst., vol. 37, no. 2, Nov. 2007, pp. 99–122.

[8] Vadym Puidenko, Vyacheslav Kharchenko "The Minimizating of Hardware for Implementation of Pseudo LRU Algorithm for Cache Memory" The 11th IEEE International Conference on Dependable Systems, Services and Technologies, DESSERT'2020, 14-18 May, 2020, Kyiv, Ukraine pp.63-71