Implementations of Hybrid FPGA Microwave Format Extension as a Control Device

Ahmad AbdulQadir AlRababah

Faculty of Computing and Information Technology, King Abdulaziz University, Rabigh 21911, KSA.

Summary
The article proposes a method for synthesizing a composite microprogram device control in the basis of hybrid FPGA. The use of class codes of pseudo-equivalent operator linear circuits to reduce the hardware costs in the scheme and reduce the total cost implementation for control devices. An application case of the proposed method is specified. Also It is exposed that the savings of hardware resources reaches 30% while maintaining the time characteristics device. The research task is the development of a method for the synthesis of CMDM, which reduces the amount of macro cells PLA in the scheme of functions formation of excitation of memory. In this case, the control algorithm is formed as a graph-scheme algorithm (GSA).

Keywords:
composite microprogram control device, format extension micro-commands, LUT, hybrid FPGA, hardware costs reduction, pseudo equivalent OLC.

1. Introduction

Composite microprogram devices management (CMDM) is effective realization of linear control algorithms. When implementation of CMDM schemes, the task of reducing hardware costs, which is relevant for synthesis of any control devices[1]. Methods of solving this task largely depend on the specific conditions of elements basis[2],[3]. At the present time, the Proactive Logical Integrated Circuits are the type of FPGA (Field-Programmable Gate Array)[4]. Such FPGAs include table type (LUT, look-up table) and built-in blocks programmable logic assessments (PLA), programmable logic array (PLA)[5]. An example is the APEX20K chips, which include PLA blocks with 32 inputs (S = 32), 16 outputs (t = 16) and 32 terms (q = 32). Such FPGAs do not include built-in memory blocks, on which a system of output functions of the CMDM is realized. In this paper we have proposed synthesis methods of CMDM in the basis of hybrid FPGAs[2],[6].
The aim of the study is to reduce the CMDM scheme when it is implemented in the hybrid FPGA based on the introduction to the format micro commands of codes in modules of pseudo equivalent operator linear circuits (OLC) [7]. The research task is the development of a method for the synthesis of CMDM, which reduces the amount of macro cells PLA in the scheme of functions formation of excitation of memory. In this case, the control algorithm is formed as a graph-scheme algorithm (GSA) [8],[9].

2. FPGA Nature and Architecture

Field Programmable Gfateway (FPGA) is an integrated circuit that is configured to be configurable by the customer or designer and after manufacture can be programmed by the term. An FPGA configuration is usually used to determine hardware descriptor language (HDL), which is similar to using an application-specific integrated circuit (ASIC). Circuit is used previously as indicating configuration, but this is becoming less common because of the advent of electronic design automation tools. FPGAs include a number of programmable logical blocks and hierarchies of configurable linkages that allow you to connect the blocks with many logical different configurations. Logical blocks can be configured to perform complex combination functions or simple logical elements such as AND, XOR. In most FPGAs, logical blocks also include memory elements, which can be simple triggers or complete memory blocks. [1] Numerous FPGAs can be reprogrammed to perform various logic operations [2] allowing flexible configurable calculations made in computer programs.
Modern programmable gate layout (FPGA) shown on figure (1) provides significant resources for logic gates, and RAM blocks implement complex digital calculations. [2] Since FPGA design purposes very fast, I/O ratios and bidirectional data bus become a problem to verify the validity of the actual data synchronization of the setup time and drain time. Soil planning allows the allocation of resources within the FPGA to meet these deadlines. The FPGA can be used to execute any logic function that can be performed by an ASIC. The ability to update the functionality of the delivery, the partial reconfiguration of part of the structure [3] and the irregular low technical cost compared to ASIC design that provides many benefits for many applications. [1]
It has some FPGA analog functions besides digital services. The most common analogue function is the programmable voltage which is allowing an engineer to operate at a low speed with a slightly loaded pin which

Manuscript received November 5, 2018
Manuscript revised November 20, 2018

can otherwise be called or connected unacceptable and to set a fast-paced pin for high-speed channels that are otherwise too slow [4] [5] Quartz Oscillators, Built-in Condenser Oscillators, and Built-in Voltage Controlled Oscillators are used for triggering and controlling the clock speed and the high speed parallel-serial clock (SERDES) and clock speed recovery are widely used. Receivers Input terminals have fairly common differential comparators designed to connect to differential signal channels. Several "FPGA mixed signals" have been built with a peripheral analogue-to-digital converter (ADC) and a digital-to-analogue converter (DAC) analog signal conditioning device that enables them to operate on the chip. [6] Such devices blur the line between the FPGA, which carries digital and zeroes, the internal, programmable connective fabric and the field programmable analog array (FPAA) carrying the analog values of the internal, programmable connective fabric.

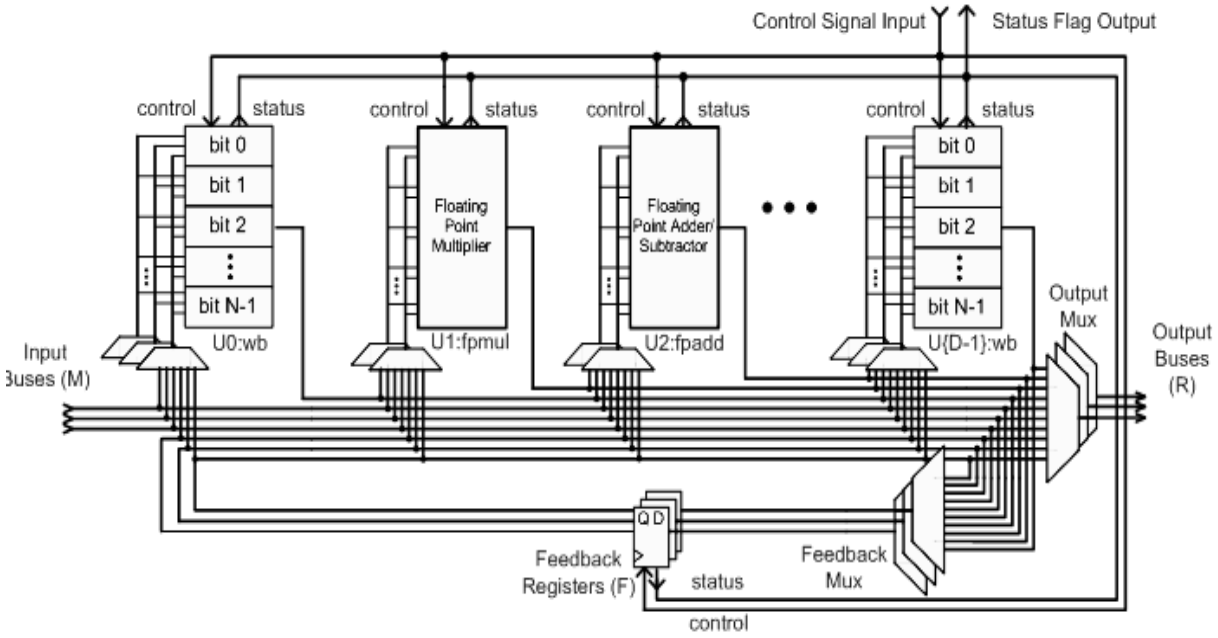

Fig. 1 FPGA Architecture

The latest FPGA-based project to hit Kickstarter, the LOGi FPGA Dev Board for Raspberry Pi and BeagleBone Black from Valent F(x), The Valent F(x) Kickstarter project actually encompasses two development boards. There's one version of the Valent F(x) board called the LOGi-Pi that's designed for the RaspBerry Pi—based on a Broadcom BCM2835 (Figure 2). And there's a second version called the LOGi-Bone for the BeagleBone Black development board, which is based on a TI AM335x microprocessor (Figure 3)

Fig. 2 LOGi-Pi for the Raspberry Pi

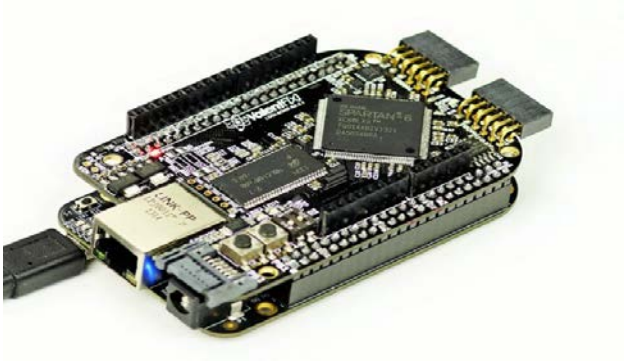

Fig. 3 LOGi-Bone for the Beaglebone Black

3. Peculiarities of CMDM with Separation Codes

We denote in some set of vertices B = {b0, bE}∪B1∪B2 and the set of arches E connecting these vertices[10]. We denote the initial vertex by b0, bE, the set of operator vertices is B1, the set of the vertex vertices is B2. The operator vertex bq ∈ B1 contains a set of micro operations Y (bq) ⊆ Y, where Y = {y1, ..., yN} is the set of micro-operations (output signals) generated by the control device. Us - The vertex bp ∈ B2 contains one component of the set, the logical conditions X = {x1, ..., xL} (the input signals fishing) [11],[12]. A linear is understood to be GSA, in which at least 75% of the overall number of vertices are operator ones. The operator linear chain (OLC) is a graph of the algorithm. We form the set of the OLC C = {α1, ..., αG}, When each pair of neighboring vertices of the OLC αg ∈C is connected, this is determined by the arc ei ∈ E. Each OLC αg ∈ C has different number of inputs Ikg and only one Og yield. For The main definitions of the OLC, their inputs and outputs are considered. Note that each vertex bq ∈ B1 corresponds to the MIq microinstruction saved in the control memory by the address Aq [13]. For addressing micro commands are sufficient

$$R = \lceil \log2(M) \rceil \text{ bit}, \quad (1)$$

where M = ⌈B1⌈. Suppose that every OLC αg ∈ C consists of Fg operator vertices, and let Fmax = max (F1, ..., FG) [14]. We associate with each OLC αg ∈ C a binary code K(αg) digits

$$R1 = \lceil \log2(G) \rceil, \quad (2)$$

and every component bq ∈ B1 - binary code K(bq) digits

$$R2 = \lceil \log2 (Fmax) \rceil. \quad (3)$$

To encode the OLC, we use the elements τr∈τ, and for the encoding of their components, the elements Tr ∈ T, with this is ⌈τ⌉ = R1 and ⌈T⌉ = R2[15]. Components coding is in a natural order as

$$K[(bg)i] = K[(bg)i-1] + 1, \quad (4)$$

where g = 1, G; i = 1, FG. If for GSA the condition catch

$$R1 + R2 = R, \quad (5)$$

Then for its interpretation, a (CMDM) with code partitioning can be used (Figure 4), which expressed as U1 [16],[17].
In CMDM U1, the micro-instruction circuit of addresses implements the excitation system counter CT (6) and the register RG (7):

$$\Phi = \Phi(\tau, X) \quad (6)$$

$$\Psi = \Psi(\tau, X) \quad (7)$$

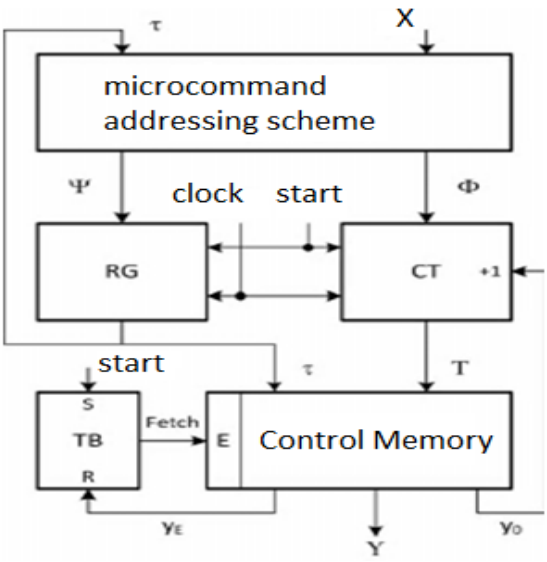

Fig. 4 Structural view of CMDM U1

With this approach, the address of the MIq micro command is defined by:

$$A(bq) = K(\alpha g) * K(bq) \quad (8)$$

where the vertex bq enters the OLC $\alpha g \in C$, and the symbol "*" denotes the concatenation operation[18].
On the Start signal, the starting address of the firmware is entered in the RG and CT, and the select trigger is set to a single state, allowing the selection of the commands[19]. If the read microinstruction doesn't correspond to the OLC outputs, then simultaneously with the microprocesses Y(bq), a signal y0 is generated, along which one is added to the CT content, forming the address of the components to be next of the current OLC[20],[21]. If the micro-command corresponds to the output of the OLC, the signal y0 is not generated. The address of the next OLC input is formed by the CAM scheme. Upon reaching the end yE signal is created, the flip-flop resets, and the selection of the commands stops[22],[23].
The total of rapports in the CAM scheme might be reduced by introducing a converter of OLC codes into class codes of pseudo equivalent OLCs. OLC $\{\alpha i, \alpha j\} \subseteq C$ are called pseudo equivalent in case if their outputs and inputs are connected to the same vertex[24],[25]. It should be noted, however, that the implementation of such a converter requires additional hardware resources of the FPGA chip[26].
In this paper, we have proposed a method for synthesizing CMDM in a hybrid FPGA basis, the main purpose of which is to reduce the hardware costs in the control device circuit[27].

4. The Proposed Method

Let OLC $\alpha g \in C1$ if $\alpha g \in C1$ and its output is not connected with the input of a finite vertex. We find the partition BC = {B1, ..., BI} of the set C1 into classes. We code the classes' $Bi \in BC$ with the binary codes K (Bi) digits

$$RI = \lceil \log_2(I) \rceil. \quad (9)$$

It is proposed in previous works to introduce the field K (Bi) into the format of microinstructions. In this case, the control memory is implemented on the built-in memory blocks[28]. However, in the case of hybrid FPGAs, such blocks are not available. Function blocks of the built-in memory can perform LUT elements, which can be considered as a memory block with SL inputs and one output. Obviously, the LUT element has 2SL memory cells.
Suppose that for a given GSA the implemented relation

$$SL \geq R. \quad (10)$$

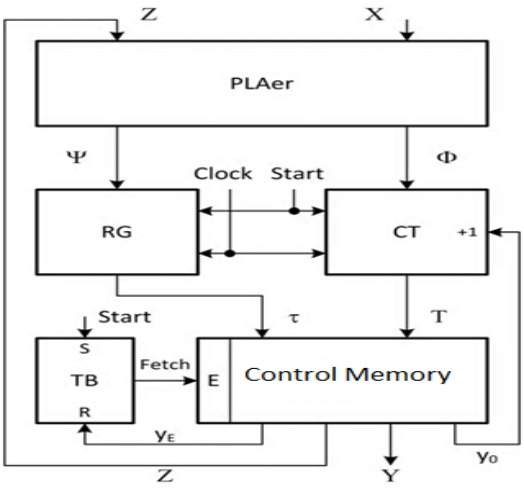

Fig. 5. Structural diagram of CMDM U_2

In the CMDM U2 on (Figure 5), the CAM scheme is realized as a cumulative PLA units, called PLAer. The UE scheme is realized from a set of LUT elements, called LUTer. Blocks CT and RG are also implemented on LUT elements[29].
The classes $Bi \in BC$ are encoded by the variables zr that form the set Z = {z1, ..., zRI}. The PLAer block applies the functions

$$\Phi = \Phi(Z, X); \quad (11)$$

$$\Psi = \Psi(Z, X). \quad (12)$$

The LUTer block implements the functions

$$Y = Y(\tau, T); \quad (13)$$

$$Z = Z(\tau, T) \quad (14)$$

$$y0 = y0(\tau, T). \quad (15)$$

In this study we have proposed a combination method for the CMDM U2, which has the bellowed phases:
1. Sets formation of C, C1 and BC
2. Coding of OLC, it's modules and classes $Bi \in BC$.
3. Formation the contents of the LUTer block.

4. Formation of the PLAer block table.
5. Synthesis the logic circuit of CMDM

5. Development of CMDM Model

Consider the VHDL language implementation of each from the listed blocks. Feature of these descriptions is the uses of so-called generic-constants that allow parameterize the projected modules[30].
Addressing scheme, this block forms the code for the next OLC Ψ and the code of the input of the OLC F on the basis of the code current OLC and logical conditions signals[31].
The external description of the block looks like this:

```
entity CA is  -- Circuit of addressing
generic (
R_Tau: natural;
R_LC:  natural;
R_T: natural);
port (X: in bit_vector (1 to R_LC);
      T: in bit_vector (1 to R_Tau);
      Phi: out bit_vector (1 to R_T);
      Ksi: out bit_vector (1 to R_Tau));
end entity;
```

And the internal block description:

```
architecture CA_A of CA is
begin
process (X, T)

variable nX: bit_vector (1 to R_LC);

variable nT: bit_vector (1 to R_Tau);

variable D1: bit_vector (1 to R_Tau);

variable D2: bit_vector (1 to R_T);

begin
       nX := not X;
       nT := not T;


D1(1) := nT(1) and T(2) and nT(3) and nT(4);
D2(1) := '0';


Ksi <= D1(1 to R_Tau);
Phi <= D2(1 to R_T);
end process;
end architecture CA_A;
```

On the signal of controlled synchronization, from the cache module, in the case of
y0 = 1, the counter loads the value from the schema addressing. In the case of y0 = 0, the counter increments internal content[24].
External description of the counter:

```
entity CT is
  generic (R_T: natural);
  port (D: in bit_vector (1 to R_T);
        Int_Clk: in bit;
        R: in bit;
        y0:  in bit;
        O: out bit_vector (1 to R_T));
end entity CT;
```

Internal description of the counter:

```
architecture CT_A of CT is
begin
process (Int_Clk, R)
variable contents: bit_vector (1 to R_T);
variable i: natural;
variable carry: bit;
begin
 if r='1' then
    contents:=(others=>'0');
  else
-- ------------------------------
  if y0='0' and Int_Clk='1' then
     carry:='1';
f1:  for i in R_T downto 1 loop
  if carry='1' then
     if contents(i)='1' then
        contents(i):='0';
        carry:='1';
     else
        contents(i):='1';
        carry:='0';
     end if;
  end if;
end loop f1;
  end if;
  if y0='1' and Int_Clk='1' then
     contents := D;
   end if;
-- ------------------------------
   end if;
  O <= contents;
-- ------------------------------
end process;
end architecture CT_A;
```

6. Implemented Example of the Proposed Method

Let GSA contain G = 7 OLC: C = {α1, ..., α7}, where α7 ∉ C1. In the set C1, I = 3 classes are distinguished. COLLECT: BC = {B1, B2, B3}, where B1 = {α1}, B2 = {α2, α3}, B3 = {α4, α5, α6}. The OLC αi are formed from the following sequences of operator vertices:
α1 = b1, b2, b3 α2 = b4, b5, b6, b7 α3 = b8, b9 α4 = b10, b11, b12
α5 = b13, b14, b15, b16 α6 = b17, b18 α7 = b19, b20.
According to (2), R1 = 3 variables in a set τ = {τ1, τ2, τ3} are sufficient for encoding the OLC. Maximum number of components Fmax = 4, for their coding according to (3) R2 = 2 variables of a set T = {T1, T2} are sufficient. In general, to encode M = 20 per vertex vertices, according to (1), R = 5 binary digits are sufficient; therefore, condition (5) is satisfied, and the application of the method of code separation is expedient. In this case, to encode I = 3 classes, according to (9), RI = 2 variables are necessary, which form the set Z = {z1, z2}[32].

We code the OLC αg ∈ C and their classes in the random mode: K (α1) = 000, ..., K(α7) = 110; K (B1) = 00, ..., K (B3) = 10. To satisfy condition (4), assign the first element of each OLC αg ∈ C to 00, the second to 01, the third to 10, and the fourth to 11. This will determine the addresses) A(bq) microcomputers CMDM U2, shown in Table 1. Here and below, the notation Ui denotes the CMDM Ui.
From Table.1 we have, for example, A(b6)=00110, A(b18) = 10101, and so on [33].
The format of CMDM U2 micro commands includes the fields y0, yE, FY, FB, where the field FY holds the code of the set of micro operations, and the field FB is the code of the class Bi ∈ BC. If y0 = 1, then the substances of the FB field are ignored [11], [40].
The contents of the LUTer block CMDM U2 are shown in Table. 2. The principle of forming the content of the LUTer block is trivial[34]. A set of micro operations Y(bq) is written to the string with the address A(bq). If the vertex bq ∈ B1 is not an OLC exit αg ∈ C, then the micro projection y0 is written in the string address A(bq). Otherwise, the code K(Bi) is written in this line, where αg ∈ Bi. If a vertex bq ∈ B1 is linked to the last vertex, and then the micro operation yE is inscribed to the string with the address A(bq) [31],[35].
In Table. 2, the symbolic contents of the LUTer block are given, and the transition to bit lines is not difficult. Let the transitions from the OLC outputs αg ∈ C1 be characterized the next system of generalized transition formulas:

$$B_1 \to x_1 b_4 \vee \overline{x_1} x_2 b_6 \vee \overline{x_1 x_2} b_8;$$
$$B_2 \to x_3 x_4 b_{10} \vee x_3 \overline{x_4} b_{13} \vee \overline{x_3} x_5 b_{19} \vee \overline{x_3 x_5} b_{16};$$
$$B_3 \to x_5 b_{11} \vee \overline{x_5} x_3 b_{17} \vee \overline{x_5 x_3} b_8. \qquad (16)$$

Such a system is the basis for forming the board of the PLAer block with columns Bi, K (Bi), bq, A (bq), Xh, Ψh, Φh, h. The purpose of the columns is clear from Table.3, which defines the transitions for the class B3∈BC[36]. The addresses of micro commands are taken from Table1. Note that Ψ = {D1, D2, D3}, Φ = {D4, D5}. The overall number of rows H2 in the CMDM block table U2 coincides with the quantity of rapports the systems of generalized alteration formulas [37].

Table 1: Addresses of micro commands CMDM U2

T_1T_2 \ $\tau_1\tau_2\tau_3$	000	001	010	011	100	101	110
00	b_1	b_4	b_8	b_{10}	b_{13}	b_{17}	b_{19}
01	b_2	b_5	b_9	b_{11}	b_{14}	b_{18}	b_{20}
10	b_3	b_6	*	b_{12}	b_{15}	*	*
11	*	b_7	*	*	b_{16}	*	*

Table 2: Content of the LUTer block CMDM U2

T_1T_2 \ $\tau_1\tau_2\tau_3$	000	001	010	011	100	101	110
00	$y_0\, Y(b_1)$	$y_0\, Y(b_4)$	$y_0\, Y(b_8)$	$y_0\, Y(b_{10})$	$y_0\, Y(b_{13})$	$y_0\, Y(b_{17})$	$y_0\, Y(b_{19})$
01	$y_0\, Y(b_2)$	$y_0\, Y(b_5)$	$z_2\, Y(b_9)$	$y_0\, Y(b_{11})$	$y_0\, Y(b_{14})$	$z_1\, Y(b_{18})$	$y_E\, Y(b_{20})$
10	$Y(b_3)$	$y_0\, Y(b_6)$	*	$z_1\, Y(b_{12})$	$y_0\, Y(b_{15})$	*	*
11	*	$z_2\, Y(b_7)$	*	*	$z_1\, Y(b_{16})$	*	*

Table 3: Portion of the PLAer table of CMDM U2

B_i	$K(B_i)$		b_q	$A(b_q)$					X_h	Ψ_h	Φ_h	h
	z_1	z_2		τ_1	τ_2	τ_3	T_1	T_2				
B_3	1	0	b_{11}	0	1	1	0	1	x_5	$D_2\ D_3$	D_5	1
			b_{17}	1	0	1	0	0	$\overline{x_5} x_3$	$D_1\ D_3$	–	2
			b_8	0	1	0	0	0	$\overline{x_5 x_3}$	D_2	–	3

In our example, H2 = 10. Note that H1 = 20, where Hi denotes the quantity of rows for the table of the PLAer block of CMDM Ui. Systems (11) - (12) are formed according to the transition table [38]. So, from the table 3, it is possible to construct fragments:

$$D_1 = z_1 \overline{z_2} x_5 x_3;$$
$$D_2 = z_1 \overline{z_2} x_5 \vee z_1 z_2 x_5 x_3;$$
$$D_3 = z_1 \overline{z_2} x_5 \vee z_1 \overline{z_2} x_5 x_3. \quad (17)$$

If the conditions

$$S \geq L + R1 + R3; \quad (18)$$

$$t \geq R1 + R2; \quad (19)$$

$$q \geq H2, \quad (20)$$

The PLAer block is trivially implemented on one PLA macro cell. If these relationships are violated, then several macro cells are required. To decrease the quantity of PLA macro cells in the PLAer block diagram, known methods (11).

When condition (10) is satisfied, each function of systems (12) - (14) is realized on one element of LUT. Such a solution is optimal. In this case, the LUTer block table is treated as a truth table functions (13) - (15). Note that if the conditions (18) - (20) are violated, the use of the above approach is impossible, and the structure of CMDM and the corresponding synthesis method need to be modified and further directions of research.

7. Conclusion

The proposed method for expanding the format of micro commands due to the introduction of a field with the code of the class of pseudo equivalent OLC is focused on reducing the number of macro cells PLA in the scheme for generating the address of microinstructions. In this case, the number of cycles of the interpretation of the control algorithm coincides with the corresponding value for the basic structure of the CMDM U1 with the code division. A decrease in the quantity of rapports in memory excitation occupations can lead to a decrease in the level numbers of the combinational fragment of the CMDM. This in turn leads to an increase in the speed of the digital system as a whole. The examples examined by us showed that the number of PLA macro cells depending on their parameters. The value of the parameters decreases by a value of up to 30% in comparison with CMDM U1. We recall that the application of this method is expedient only for linear approach when condition (5) is satisfied.

The scientific novelty of the proposed method is the using classes of pseudo equivalent OLCs to decrease the quantity of macro cells PLA in the addressing scheme of microinstructions. The practical importance of the method is to reduce the number of macro cells in the CMDM scheme implementation, which makes it possible to obtain schemes that have a lower cost than the known analogs.

References

[1] Archer, C.J. and G.R. Ricard, Administering registered virtual addresses in a hybrid computing environment including maintaining a cache of ranges of currently registered virtual addresses. 2016, Google Patents.

[2] Shreejith, S., B. Anshuman, and S.A. Fahmy. Accelerated artificial neural networks on FPGA for fault detection in automotive systems. in Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016. 2016. IEEE.

[3] AlRababah, A., Digital Image Encryption Implementations Based on AES Algorithm. VAWKUM Transactions on Computer Sciences, 2017. 13(1): p. 1-9.

[4] Gupta, P. Accelerating datacenter workloads. in 26th International Conference on Field Programmable Logic and Applications (FPL). 2016.

[5] Chen, R. and V.K. Prasanna. Accelerating Equi-join on a CPU-FPGA heterogeneous platform. in Field-Programmable Custom Computing Machines (FCCM), 2016 IEEE 24th Annual International Symposium on. 2016. IEEE.

[6] Jain, A.K., D.L. Maskell, and S.A. Fahmy. Are coarse-grained overlays ready for general purpose application acceleration on fpgas? in Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), 2016 IEEE 14th Intl C. 2016. IEEE.

[7] Dangjuan, L. and W. Shenjiang. Cultivating Engineering Innovation Ability Based on Optoelectronic Experimental Platform. in Education and Training in Optics and Photonics. 2017. Optical Society of America.

[8] Sansyzbaevich, I.S., et al. Development of algorithm flow graph, mealy automaton graph and mathematical models of microprogram control mealy automaton for microprocessor control device. in Control and Communications (SIBCON), 2017 International Siberian Conference on. 2017. IEEE.

[9] AlRABABAH, A.A., IMPLEMENTATION OF SOFTWARE SYSTEMS PACKAGES IN VISUAL INTERNAL STRUCTURES. Journal of Theoretical & Applied Information Technology, 2017. 95(19).

[10] Aslam, M.H., et al., Exploring the effect of LUT size on the area and power consumption of a novel memristor-transistor hybrid FPGA architecture. Arabian Journal for Science and Engineering, 2016. 41(8): p. 3035-3049.

[11] Miroshkin, A., EXTENSION OF MICROINSTRUCTION FORMAT FOR CONTROL UNIT IMPLEMENTATION

ON HYBRID FPGA. Radio Electronics, Computer Science, Control, 2014(1).
[12] Al-Rababah, A.A. and R. Biswas, Rough vague sets in an approximation space. 2008.
[13] Osann Jr, R., FPGA with hybrid interconnect. 2010, Google Patents.
[14] Dong, X. and X. Zhang. A high performance FPGA implementation of 256-bit Modular multiplication processor over GF (p). in Computer and Communications (ICCC), 2016 2nd IEEE International Conference on. 2016. IEEE.
[15] Agiakatsikas, D., E. Cetin, and O. Diessel. FMER: A hybrid configuration memory error recovery scheme for highly reliable FPGA SoCs. in Field Programmable Logic and Applications (FPL), 2016 26th International Conference on. 2016. IEEE.
[16] AlRababah, A.A., Lempel-Ziv Implementation for a Compression System Model with Sliding Window Buffer.
[17] Kim, S. and R. Lu, The Pseudo-Equivalent Groups Approach as an Alternative to Common-Item Equating. ETS Research Report Series, 2018.
[18] Kaushansky, D., et al., Programmable test instrument. 2017, Google Patents.
[19] Chin, S.A., et al., Hybrid lut/multiplexer fpga logic architectures. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2016. 24(4): p. 1280-1292.
[20] Fiessler, A., et al. HyPaFilter: A versatile hybrid FPGA packet filter. in Proceedings of the 2016 Symposium on Architectures for Networking and Communications Systems. 2016. ACM.
[21] Moustafa, A., et al., A New Dynamic Model for Software Testing Quality. Research Journal of Applied Sciences, Engineering and Technology, 2014. 7(1): p. 191-197.
[22] Vasumathi, B. and S. Moorthi, Implementation of hybrid ANN–PSO algorithm on FPGA for harmonic estimation. Engineering Applications of Artificial Intelligence, 2012. 25(3): p. 476-483.
[23] Al-Rababah Ahmad, A., UML–Models Implementations in Software Engineering System Equipments Representations. International Journal of Soft Computing Applications, 2009(4): p. 25-34.
[24] Cabillic, G. and J.-P. Lesot, Selective compiling method, device, and corresponding computer program product. 2017, Google Patents.
[25] Al Ofeishat, H.A. and A.A. Al-Rababah, Real-time programming platforms in the mainstream environments. IJCSNS, 2009. 9(1): p. 197.
[26] Kastensmidt, F. and P. Rech, FPGAs and Parallel Architectures for Aerospace Applications. Soft Errors and Fault-Tolerant Design, 2016.
[27] Venkateswaran, V., F. Pivit, and L. Guan, Hybrid RF and digital beamformer for cellular networks: Algorithms, microwave architectures, and measurements. IEEE Transactions on Microwave Theory and Techniques, 2016. 64(7): p. 2226-2243.
[28] AlRababah, A.A.Q., On the associative memory utilization in English-Arabic natural language processing. International Journal of Advanced and Applied Sciences, 2017. 4(8): p. 14-18.
[29] Barkalov A., Kovalev A., Nikolaenko D., Modelling of compositional microprogram control unit with division of codes and memory of microinstructions
[30] Choi, J. and R.A. Rutenbar, Video-rate stereo matching using Markov random field TRW-S inference on a hybrid CPU+ FPGA computing platform. IEEE Transactions on Circuits and Systems for Video Technology, 2016. 26(2): p. 385-398.
[31] Wiśniewski, R., Implementation of Concurrent Control Systems in FPGA, in Prototyping of Concurrent Control Systems Implemented in FPGA Devices. 2017, Springer. p. 139-165.
[32] Durand, Y., et al. A Programmable Inbound Transfer Processor for Active Messages in Embedded Multicore Systems. in 2017 Euromicro Conference on Digital System Design (DSD). 2017. IEEE.
[33] Wiśniewski, R., Modelling of Concurrent Systems in Hardware Languages, in Prototyping of Concurrent Control Systems Implemented in FPGA Devices. 2017, Springer. p. 117-137.
[34] Juang, J.-G., W.-K. Liu, and R.-W. Lin, A hybrid intelligent controller for a twin rotor MIMO system and its hardware implementation. ISA transactions, 2011. 50(4): p. 609-619.
[35] Maeda, T. and R. Matsubara, Storage apparatus and failure location identifying method. 2017, Google Patents.
[36] Lu, R. and H. Guo, A Simulation Study to Compare Nonequivalent Groups With Anchor Test Equating and Pseudo-Equivalent Group Linking. ETS Research Report Series, 2018.
[37] Wiśniewski, R., Prototyping of Concurrent Control Systems, in Prototyping of Concurrent Control Systems Implemented in FPGA Devices. 2017, Springer. p. 99-116.
[38] de Rochemont, L.P. and A.J. Kovacs, Hybrid computing module. 2016, Google Patents.
[39] Takeuchi, K., A. Tanabe, and K. Manabe, Semiconductor memory device including rewriting operation for improving the long-term reliability of the resistance variable element. 2017, Google Patents.
[40] Poole, V., et al., On the Anomalous Balmer Line Strengths in Globular Clusters. The Astronomical Journal, 2010. 139(3): p. 809.

Ahmad AbdulQadir Al Rababah received Phd degree in 1998 in computer Engineering, now he is an associate professor at king Abdulaziz university(KSA), he has around 20 experience years of teaching and research in different fields of computing technology and engineering, his research interest areas are: information systems, software engineering, artificial intelligence and others.