Розглянуто математичні моделі обчислювальної схеми у вигляді орієнтованого ациклічного графа для побудови паралельних суматорів з паралельним способом перенесення. Продемонстровано зв'язок між обчислювальними кроками орієнтованого ациклічного графа та процесом перенесення одиниці у схемі багаторозрядного суматора, що дозволяє визначати оптимальне число перенесень у схемі багаторозрядного паралельного суматора з паралельним способом перенесення у теоретико-числовому базисі Радемахера. Процес додавання двійкових чисел у схемі суматора використовує алгоритм логарифмічного підсумовування

Ключові слова: суматор, каскадна схема, направлений ациклічний граф, ТЧБ Радемахера

Рассмотрены математические модели вычислительной схемы в виде ориентированного ациклического графа для построения параллельных сумматоров с параллельным способом переноса. Продемонстрирована связь между вычислительными шагами ориентированного ациклического графа и процессом переноса единицы в схеме многоразрядного сумматора, что позволяет определять оптимальное число переноса в схеме многоразрядного параллельного сумматора с параллельным способом переноса в теоретико-числовом базисе Радемахера. Процесс суммирования двоичных чисел в схеме сумматора использует алгоритм логарифмического суммирования

Ключевые слова: сумматор, каскадная схема, направленный ациклический граф, ТЧБ Радемахера

UDC 681.325
DOI: 10.15587/1729-4061.2016.70355

STUDY OF CARRY OPTIMIZATION WHILE ADDING BINARY NUMBERS IN THE RADEMACHER NUMBER-THEORETIC BASIS

M. Solomko
Candidate of Technical Sciences, Associate Professor*
E-mail: doctrinas@ukr.net

B. Krulikovskyi
Candidate of Technical Sciences, Associate Professor*
E-mail: kboris@ukr.net

*Department of Computer Engineering
National University of Water and Environmental Engineering
Soborna str., 11, Rivne, Ukraine, 33028

1. Introduction

The technology of computing in the Rademacher number-theoretic basis (NTB) has existed for a long time, the information has been accumulated, mathematical and software resources have been developed. The binary system is one of the simplest positional numeral systems and, consequently, electronic equipment, made on its basis is the most reliable and versatile.

The unary numeral system, which in the 60s has been used for developing the «Promin» computer is a simpler numeral system than binary. But it can hardly be attributed to positional numeral systems, and speed is low.

The range of addition of numbers in the Rademacher NTB is:

$$x_D + y_D =< 2^k - 1, \tag{1}$$

where k is the number of bits. The number of addition options for the multi-bit adder with the full adder in the first bit is:

$$c = 2 \times 2^k \times 2^k, \tag{2}$$

where k is the number of bits. The number of addition options for the half-adder in the first bit of the multi-bit adder is:

$$b = \sum_{n\quad 0}^{2\quad 1} 2 \quad - n, \tag{3}$$

or

$$b = \sum_{n=1}^{2^k} n, \tag{4}$$

where k is the number of bits.

Arithmetic devices with the characteristics (1)–(4) for a number of examples do not require additional hardware capacity, so the cost of them will be kept low, and this gives premise to further developments in this area.

The methods of arithmetic operations are implemented by gate circuits of functional elements in the bases consisting of the functions of the algebra of logic. Minimization of complexity and depth of logic circuits is one of the major and practically important problems in this theory. Currently, the circuit optimization is to some extent implemented by software (for example, the Logic Friday program performs circuit optimization in a set of bases – from a range of available elements). However, in general, the problem of optimization of the logic circuit (including adder circuits) today is solved empirically. In this regard, the need for mathematical research of the outcome formation process in the adder circuit is urgent, which allows the circuit control in the design.

2. Literature review and problem statement

The binary system has many undeniable advantages, but it also has a number of essential drawbacks:

1. The binary system allows the sign-and-magnitude representation only of positive numbers. Representation of negative numbers needs the use of special codes – two's complement and one's complement. This complicates the arithmetic structure of computers and reduces their speed.

2. Zero redundancy. All computer circuits and devices are exposed to numerous external and internal influences, resulting in failures and faults. Such errors in the binary system cannot be detected, since all binary code combinations are allowed. The lack of code redundancy, which would allow detecting errors directly in the computing process, is a principal drawback of the classical binary number system.

3. «Long carry». The maximum time of the addition operation is reached when the carry that occurred in the first bit passes all the other bits (e. g., when adding codes 11...11 and 00...01), which is a principal factor of computer speed reduction.

In [1] states that the main advantage of Fibonacci microprocessors is noise immunity and the possibility to obtain reliable data at the microprocessor output.

It should be assumed that the carry in the addition operations in the Rademacher NTB is not related to the main problem, as evidenced by, for example, the following publications: [2] presents the structure of the carry select adder to perform parallel computing in the FPGA applications. Optimum structures for the FPGA implementation are shown. The simulation results are used for verification of the theory; [3] presents the 180 nm CMOS technology of the computing process in the carry select adder circuit on the basis of the parallel ripple carry adder. The proposed approach provides a reasonable compromise between the computing cost and performance. Note that the CLA adder (carry look ahead adder) occupies a larger area on the chip and needs more power for computing; [4] developed the mathematical model to estimate the time attributes of the carry self-timing for the Carry Select Adder structure, presented pilot projects of the given approach; [5] presents the 128-bit Carry Select Adder structure for the VLSI system design. The architecture, based on the modification of 16, 32 and 64-bit Carry Select Adder (CSLA) is designed, which is promising for reducing the adder area and carry delay.

Reduction of hardware complexity of the electronic adder by introducing the information redundancy on the basis of modeling of the binary-ternary redundant numeral system adder is considered in [6].

In [7], efficient use of the cascade computing circuit to support dynamic data querying in the cloud is shown and experimentally proved.

In [8], it is demonstrated that the operation of vector-matrix multiplication, which is used for solving the problems of processing, analysis of signals and images and pattern recognition, is implemented by pairwise product formation and multi-operand summation.

Modification of the method of computing of the group summation operator for real-time neural networks based on vertical and multi-operand approaches is considered in [9].

The use of nucleic acids (NA) for logic gates and computing in biotechnology and biomedicine is examined in [10]. In particular, the logic gate that uses the cascade circuit for the DNA logic implementation is shown. Nucleic acid-based logic devices were first introduced in 1994. Since then, science has seen the emergence of new logic systems for modeling mathematical functions, diagnosing diseases and even those that simulate biological systems.

Molecular logic circuits with multiple components, placed in several layers, can be synthesized using DNA. The issue whether the circuit depth, which is a standard measure of computing time-complexity in digital electronic circuits is an appropriate measure of time-complexity for the chemical circuit is considered in [11]. To this end, the quantitative analysis of how the computing time is associated with the circuit size and cascade architecture is presented. The results of the work can be used for the synthesis of efficient and more reliable molecular circuits.

Unlike [7–11], the present paper deals with the application of the cascade circuit for the operation of addition of binary numbers using the parallel carry adder in the Rademacher NTB. The use of the mathematical model that justifies the addition operation in the parallel carry adder circuit clarifies the method and allows the complexity formation of the computing algorithm.

3. Research goal and objectives

The goal of the paper is to find the relationship between computing steps of the directed acyclic graph and the unit carry process in the multi-bit adder circuit using the mathematical model of computing circuits, and thus determine the optimum number of carries in the multi-bit parallel carry adder circuit in the Rademacher number-theoretic basis.

To achieve this goal, it is necessary to solve the following problems:

1. To identify the correct accordance in the arrangement of the process of adding numbers by the cascade circuit and the process of adding bits of similar bits of binary numbers.

2. To determine the adequacy of the mathematical model in the form of the directed acyclic graph and the process of adding binary numbers in the parallel carry adder, to assign the parameters for describing the characteristics of the mathematical model.

3. To derive logic equations of the optimized adder, built taking into account the number of computing steps of the corresponding directed acyclic graph.

4. To determine the complexity of the algorithm for computing the sum and carry signals of the optimized parallel carry adder in the Rademacher NTB.

5. To make the computing protocol of the process of adding binary numbers and to test the synthesized adder for compliance of the results of adding binary numbers and the protocol.

4. Carry save

To demonstrate the carry process in arithmetic addition of multi-bit numbers, we use a tabular arrangement of the summation process (Table 1). In the top rows of Table 1, we write two 4-bit numbers – A and B.

The similar number bits are recorded in the common column in Table 1. A tabular arrangement of the summation process involves computing cycles for each bit of the number. The number of cycles is the number of bits of binary numbers. Thus, for this example, the number of computing cycles equals four. Each cycle in Table 1 is represented by two rows. The result of the bitwise addition is recorded in the first row, the result of carry – in the second row.

Table 1

The carry save process in the addition of binary numbers

Number – A			a_3	a_2	a_1	a_0
Number – B			b_3	b_2	b_1	b_0
№		Overflow	4th bit	3rd bit	2nd bit	1st bit
1	1	–	$a_3 \oplus b_3 = S_0^3$	$a_2 \oplus b_2 = S_0^2$	$a_1 \oplus b_1 = S_0^1$	$a_0 \oplus b_0 = S_0$
	2	$a_3 \wedge b_3 = I_0^4$	$a_2 \wedge b_2 = I_0^3$	$a_1 \wedge b_1 = I_0^2$	$a_0 \wedge b_0 = I_0^1$	–
2	1	–	$S_0^3 \oplus I_0^3 = S_1^3$	$S_0^2 \oplus I_0^2 = S_1^2$	$S_0^1 \oplus I_0^1 = S_1$	–
	2	$S_0^3 \wedge I_0^3 = I_1^4$	$S_0^2 \wedge I_0^2 = I_1^3$	$S_0^1 \wedge I_0^1 = I_1^2$	–	–
3	1	–	$S_1^3 \oplus I_1^3 = S_2^3$	$S_1^2 \oplus I_1^2 = S_2$	–	–
	2	$S_1^3 \wedge I_1^3 = I_2^4$	$S_1^2 \wedge I_1^2 = I_2^3$	–	–	–
4	1	–	$S_2^3 \oplus I_2^3 = S_3$	–	–	–
	2	$S_2^3 \wedge I_2^3 = I_3^4$	–	–	–	–
		$S_4 = I_0^4 \vee I_1^4 \vee I_2^4 \vee I_3^4$	S_3	S_2	S_1	S_0

The arithmetic procedure of addition consists of two operations – XOR and AND. Summation of numbers begins with LSB. XOR – $a_0 \oplus b_0 = S_0$ – of the LSB is recorded in the same column in the first row of the first computing cycle. Carry of the LSB (AND result – $a_0 \wedge b_0 = I_0^1$) is recorded in the second row of the first computing cycle in the column shifted one bit to the left. Similar actions in the first computing cycle are performed with the second bits of numbers – XOR – $a_1 \oplus b_1 = S_0^1$ – of second bits is recorded in the same column in the first row of the first computing cycle, carry of the second bits (AND result – $a_1 \wedge b_1 = I_0^2$) is recorded in the second row of the first computing cycle in column shifted one bit to the left. After finishing the considered addition operations for the third and fourth bits, computing in the first cycle is completed.

The second cycle repeats computing of the first cycle with the difference that the computing is carried out with the results obtained in the first cycle. So, XOR – $S_0^1 \oplus I_0^1 = S_1$ – of the results of the first computing cycle, the second bits is recorded in the same column in the first row of the second computing cycle. Carry of the computing results in the first cycle, the second bits (AND result – $S_0^1 \wedge I_0^1 = I_1^2$) is recorded in the second row of the second computing cycle, the column shifted one bit to the left. Similar actions in the second computing cycle are performed with the third bits of numbers – XOR – $S_0^2 \oplus I_0^2 = S_1^2$ – of the third bits is recorded in the same column in the first row of the second computing cycle, carry of the third bits (AND result – $S_0^2 \wedge I_0^2 = I_1^3$) is recorded in the second row of the second computing cycle in the column shifted one bit to the left. After finishing the considered addition operations for the fourth bits, computing in the second cycle is completed.

The third cycle uses the computing results of the second cycle, and the fourth cycle uses the computing results of the third cycle (Table 1). In the end, the sum will be – S_3, S_2, S_1, S_0 of the set binary numbers – a_4, a_3, a_2, a_1 and b_4, b_3, b_2, b_1.

The considered tabular arrangement of the process of addition of binary numbers (Table 1) coincides with the carry-save addition method (carry-save adder). For this method of addition, the solution to the problem is reduced to the use of redundant encoding of the result when the result in each position is the value that exceeds the radix.

The record of the sum (5) for the LSB in Table 1 is redundant, since for a_0=1 and b_0=1, the sum S_0 is given as S_0=10 or S_0=2. The latter notation of the sum in the binary system is unconventional, but the result is still clear.

a_1	a_0
b_1	b_0
2nd bit	1st bit
–	$a_0 \oplus b_0 = S_0$
$a_0 \wedge b_0 = I_0^1$	–

(5)

A similar redundancy of record is also present for the sums of other bits.

In the example in Table 1, the carry from the LSB to the MSB is not used. This process is replaced with an extension of the set of significant figures in each bit of the result. However, this procedure cannot be performed indefinitely. Eventually, to return to the standard representation of numbers, all carries need to be performed as shown in Table 1.

The use of carry-save adders can significantly speed up the chained addition, which, for example, is necessary for the multiplication operation.

Example 1. Using the tabular arrangement of the addition process, to compute the sum of 4-bit numbers 0001 and 0111 (Table 2).

Table 2

Addition of binary numbers 0001 and 0111 according to the computing circuit, determined by the tabular arrangement of the process

Number – A		0	0	0	1
Number – B		0	1	1	1
1	–	0	1	1	0
	0	0	0	1	–
2	–	0	1	0	–
	0	0	1	–	–
3	–	0	0	–	–
	0	1	–	–	–
4	–	1	–	–	–
	0	–	–	–	–
–	–	1	0	0	0

Example 2. Using the tabular arrangement of the addition process, to compute the sum of 4-bit numbers 0011 and 0011 (Table 3):

Table 3

Addition of binary numbers 0011 and 0011 according to the computing circuit, determined by the tabular arrangement of the process

Number – A		0	0	1	1
Number – B		0	0	1	1
1	–	0	0	0	0
	0	0	1	1	–
2	–	0	1	1	–
	0	0	0	–	–
3	–	0	1	–	–
	0	0	–	–	–
4	–	0	–	–	–
	0	–	–	–	–
–	–	0	1	1	0

Using Table 1, we write the logic equations of the 4-bit adder with the tabular arrangement of the addition of binary numbers.

$$S_0 = a_0 \oplus b_0,$$

$$S_1 = (a_1 \oplus b_1) \oplus (a_0 \wedge b_0),$$

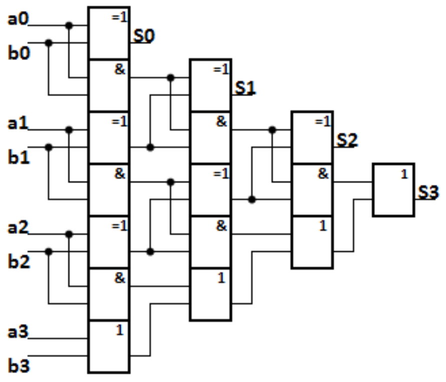
$$S_2 = ((a_2 \oplus b_2) \oplus (a_1 \wedge b_1)) \oplus ((a_1 \oplus b_1) \wedge (a_0 \wedge b_0)),$$

$$S_3 = (((a_3 \oplus b_3) \oplus (a_2 \wedge b_2)) \oplus ((a_2 \oplus b_2) \wedge (a_1 \wedge b_1))) \oplus$$
$$\oplus (((a_2 \oplus b_2) \oplus (a_1 \wedge b_1)) \wedge ((a_1 \oplus b_1) \wedge (a_0 \wedge b_0))).$$

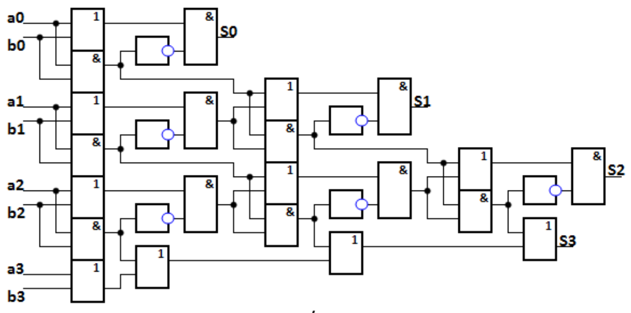
In the case where the range of numbers, in which the adder operates is fixed, the possibility of increasing the number of bits of the adder circuit is eliminated. Then, the logical OR can be used in the last bit of the adder, which will simplify the adder structure. The equation for S_3 will be of the form:

$$S_3 = (a_3 \vee b_3) \vee (a_2 \wedge b_2) \vee ((a_2 \oplus b_2) \wedge (a_1 \wedge b_1)) \vee$$
$$\vee (((a_2 \oplus b_2) \oplus (a_1 \wedge b_1)) \wedge ((a_1 \oplus b_1) \wedge (a_0 \wedge b_0))). \quad (6)$$

Fig. 1 shows the logical structure of the 4-bit adder, which implements the tabular arrangement of the process of addition of binary numbers.



a



b

Fig. 1. The logical structure of the 4-bit adder with the tabular arrangement of the process of addition of binary numbers: *a* – the adder circuit with pictorial symbol of the logical XOR, *b* – the adder circuit with one of the options of the open structure of the logical XOR

According to the classification, the adder with the tabular arrangement of the addition of binary numbers should be attributed to the parallel carry adders. The feature of the logical structure of the adder in Fig. 1 is the absence of the carry look-ahead circuits, which eliminates technical problems when increasing the number of bits of the adder.

5. The model of parallel addition of binary numbers of the Rademacher NTB

The bitwise addition of binary numbers, with a few differences, is similar to the doubling algorithm in the multi-operand summation, when adjacent pairs of terms, and then their sums are added (Table 4).

Table 4

Doubling algorithm ($n=2^3=8$)

Steps	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
1	e_1+e_2		e_3+e_4		e_5+e_6		e_7+e_8	
2	$e_1+e_2+e_3+e_4$				$e_5+e_6+e_7+e_8$			
3	$e_1+e_2+e_3+e_4+e_5+e_6+e_7+e_8$							

If $n=2^k$, where n is the number of terms, the doubling algorithm consists of k steps (cycles), n/2 additions are performed in the first step, n/4 – in the second, ..., one addition – in the last. The number of steps k is determined by the formula:

$$k = \log_2 n. \quad (7)$$

This option of the multi-operand addition is implemented according to the cascade circuit («pyramid») [12–15] (Fig. 2). There is also a recursive summation algorithm [14].

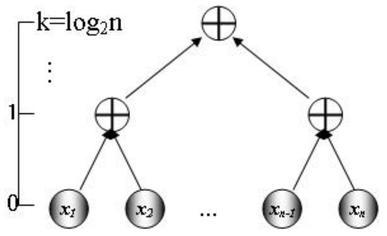


Fig. 2. The cascade circuit of the multi-operand addition algorithm (logarithmic summation)

Using the procedure of the multi-operand addition according to the cascade circuit, it is clear that for the process of parallel addition of binary numbers, bits of similar bits are pairs of data, for each, the sum is computed.

Further, similar to the procedure of the multi-operand addition, all sums of pairs of similar bit pairs of binary numbers, with their characteristics, are also divided into pairs and addition of values of pairs is performed again, and so on.

As a result, the value of the MSB of the sum of binary numbers can be compared with the value of the total sum in multi-operand addition. Apart from the sum of the MSB, intermediate results in the form of the values of sums of previous bits of binary numbers are also used in the parallel addition of binary numbers.

The computing circuit of parallel addition of binary numbers can be determined by the directed acyclic graph (Fig. 3), which is a binary tree, which, in particular, adopted the following parameters: k – the number of steps in time; ω – the total number of the algorithm operations; τ – the time of one step; T=τ·k – the time of the algorithm; L – the number of types of operations, and so on.

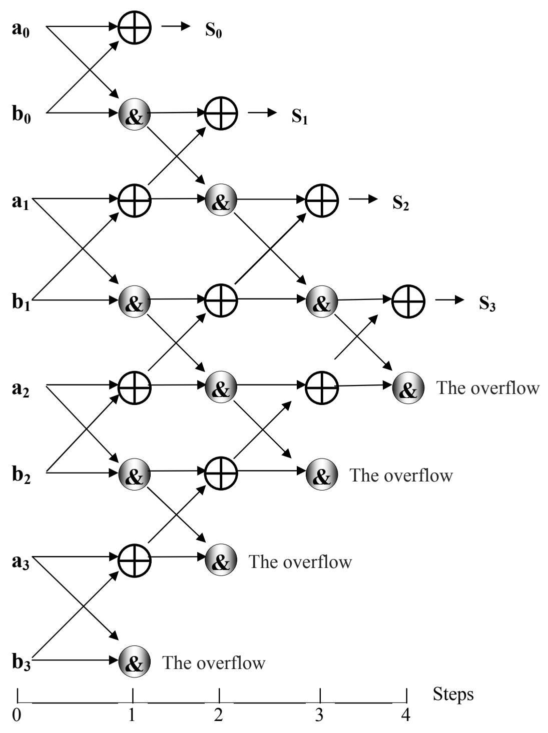


Fig. 3. The directed acyclic graph of the computing circuit of the parallel carry-save addition of 4-bit numbers of the Rademacher NTB

Thus, the computing circuit in Fig. 3 uses two logic operations – AND and XOR, the number of computing steps in it equals to the number of bits of binary numbers. For example, the parallel addition of 4-bit numbers requires four steps (Fig. 3).

The tabular arrangement of the computing circuit in Fig. 3 is presented in Table 5. The superscript of the parameter in Table 5 indicates the serial number of a computing step.

Example 3. Using the directed acyclic graph (Table 6), to compute the sum of 4-bit numbers 0001 and 0111.

In the 4th row of Table 6, the value of the MSB of the sum of binary numbers is recorded on the right, the value of the overflow is recorded on the left. The process of addition uses four computing steps.

The computing circuit of the parallel addition of 4-bit numbers with the logical OR in the last bit is shown in Fig. 4.

Table 6

The process of addition of binary numbers 0001 and 0111 according to the computing circuit, determined by the directed acyclic graph

№	0		0		0		1	
	0		1		1		1	
1	0	0	0	1	0	1	1	0
2		0	0	0	1	1	0	
3			0	0	1	0		
4				0	1			
					1	0	0	0

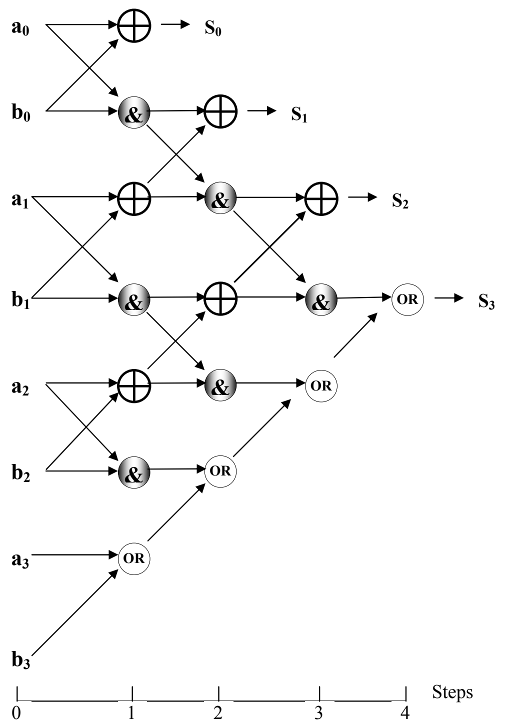


Fig. 4. The computing circuit of the parallel carry-save addition of 4-bit numbers with the logical OR in the last bit

The tabular arrangement of the computing circuit in Fig. 4 is presented in Table 7.

Table 5

The tabular arrangement of addition of 4-bit numbers according to the computing circuit, determined by the directed acyclic graph

№	a_3		a_2		a_1		a_0	
	b_3		b_2		b_1		b_0	
1	$a_3 \wedge b_3 = I_3^1$	$a_3 \oplus b_3 = S_3^1$	$a_2 \wedge b_2 = I_2^1$	$a_2 \oplus b_2 = S_2^1$	$a_1 \wedge b_1 = I_1^1$	$a_1 \oplus b_1 = S_1^1$	$a_0 \wedge b_0 = I_0^1$	$a_0 \oplus b_0 = S_0$
2		$S_3^1 \wedge I_2^1 = I_{23}^2$	$S_3^1 \oplus I_2^1 = S_{23}^2$	$S_2^1 \wedge I_1^1 = I_{12}^2$	$S_2^1 \oplus I_1^1 = S_{12}^2$	$S_1^1 \wedge I_0^1 = I_{01}^2$	$S_1^1 \oplus I_0^1 = S_1$	
3			$S_{23}^2 \wedge I_{12}^2 = I^3$	$S_{23}^2 \oplus I_{12}^2 = S^3$	$S_{12}^2 \wedge I_{01}^2 = I^3$	$S_{12}^2 \oplus I_{01}^2 = S_2$		
4				$S^3 \wedge I^3 = I^4$	$S^3 \oplus I^3 = S_3$			
					S_3	S_2	S_1	S_0

Table 7

The tabular arrangement of addition of 4-bit numbers with the computing circuit, determined by the directed acyclic graph with the logical OR in the last bit

№	a_3	a_2		a_1		a_0	
	b_3	b_2		b_1		b_0	
1	$a_3 \vee b_3 = S_3^1$	$a_2 \wedge b_2 = I_2^1$	$a_2 \oplus b_2 = S_2^1$	$a_1 \wedge b_1 = I_1^1$	$a_1 \oplus b_1 = S_1^1$	$a_0 \wedge b_0 = I_0^1$	$a_0 \oplus b_0 = S_0$
2	$S_3^1 \vee I_2^1 = S_{23}^2$		$S_2^1 \wedge I_1^1 = I_{12}^2$	$S_2^1 \oplus I_1^1 = S_{12}^2$	$S_1^1 \wedge I_0^1 = I_{01}^2$	$S_1^1 \oplus I_0^1 = S_1$	
3		$S_{23}^2 \vee I_{12}^2 = S^3$		$S_{12}^2 \wedge I_{01}^2 = I^3$	$S_{12}^2 \oplus I_{01}^2 = S_2$		
4			$S^3 \vee I^3 = S_3$				
			S_3		S_2	S_1	S_0

Example 4. Using the directed acyclic graph with the logical OR in the last bit (Table 8), to perform the addition of 4-bit numbers 0011 and 0011:

Table 8

The process of addition of binary numbers 0011 and 0011 according to the computing circuit, determined by the directed acyclic graph with the logical OR in the last bit

№	0	0		1		1	
	0	0		1		1	
1	0	0	0	1	0	1	0
2	0		0	1	0	1	
3		0		0	1		
4			0				
				0	1	1	0

In the 4th row of Table 8, the value of the MSB of the sum of binary numbers is recorded. The process of addition uses four computing steps.

Using Table 7, we write the logic equations of the 4-bit adder, the computing circuit of which is determined by the directed acyclic graph with the logical OR in the last bit.

$$S_0 = a_0 \oplus b_0,$$

$$S_1 = S_1^1 \oplus I_0^1 = (a_1 \oplus b_1) \oplus (a_0 \wedge b_0),$$

$$S_2 = S_{12}^2 \oplus I_{01}^2 = (S_2^1 \oplus I_1^1) \oplus (S_1^1 \wedge I_0^1) = \\ = ((a_2 \oplus b_2) \oplus (a_1 \wedge b_1)) \oplus ((a_1 \oplus b_1) \wedge (a_0 \wedge b_0)),$$

$$S_3 = S^3 \vee I^3 = (S_{23}^2 \vee I_{12}^2) \vee (S_{12}^2 \wedge I_{01}^2) = \\ = ((S_3^1 \vee I_2^1) \vee (S_2^1 \wedge I_1^1)) \vee ((S_2^1 \oplus I_1^1) \wedge (S_1^1 \wedge I_0^1)) = \\ = (((a_3 \vee b_3) \vee (a_2 \wedge b_2)) \vee ((a_2 \oplus b_2) \wedge (a_1 \wedge b_1))) \vee \\ \vee (((a_2 \oplus b_2) \oplus (a_1 \wedge b_1)) \wedge ((a_1 \oplus b_1) \wedge (a_0 \wedge b_0))).$$

or

$$S_0 = a_0 \oplus b_0,$$

$$S_1 = (a_1 \oplus b_1) \oplus (a_0 \wedge b_0),$$

$$S_2 = ((a_2 \oplus b_2) \oplus (a_1 \wedge b_1)) \oplus ((a_1 \oplus b_1) \wedge (a_0 \wedge b_0)),$$

$$S_3 = (((a_3 \vee b_3) \vee (a_2 \wedge b_2)) \vee ((a_2 \oplus b_2) \wedge (a_1 \wedge b_1))) \vee \\ \vee (((a_2 \oplus b_2) \oplus (a_1 \wedge b_1)) \wedge ((a_1 \oplus b_1) \wedge (a_0 \wedge b_0))). \quad (8)$$

Since the equations (6) and (8) coincide, the computing logic in Table 1 and Table 7 is similar. But the computing process in Table 7 uses computing steps instead of the carry process.

Statement. The number of computing steps of the directed acyclic graph, which models the computing process of the adder determines the optimum number of carries in the multi-bit parallel carry adder circuit in the Rademacher NTB.

In the case where the synthesized adder got a larger number of carries compared with the number of computing steps of the corresponding directed acyclic graph, this adder will be suboptimal regarding the computing time of the addition operation. In particular, according to the criterion of the directed acyclic graph, the 8-bit Brent-Kung PPA circuit [16] is suboptimal, yet it can be used in the super-adder system.

Thus, the number of computing steps defines a minimum sufficient number of carries in the adder circuit, which in turn provides the hyper-parameter for the adder structure optimization during its synthesis. The logic equations of the optimized 4-bit adder with the number of carries of four are, for example, the following:

$$S_0 = a_0 \oplus b_0,$$

$$S_1 = (a_1 \oplus b_1) \oplus (a_0 \wedge b_0),$$

$$S_2 = (a_2 \oplus b_2) \oplus ((a_1 \wedge b_1) \vee ((a_1 \vee b_1) \wedge (a_0 \wedge b_0))),$$

$$S_3 = (a_3 \vee b_3) \vee (a_2 \wedge b_2) \vee ((a_2 \vee b_2) \wedge (a_1 \wedge b_1)) \vee \\ \vee ((a_2 \vee b_2) \wedge ((a_1 \vee b_1) \wedge (a_0 \wedge b_0))). \quad (9)$$

The time T of addition of binary numbers of the adder is:

$$T = \tau \cdot k,$$

where τ is the time of one step (carry), k is the number of steps (carries). The optimum adder (9) will run faster because it contains fewer XOR operations, compared with the adder circuit in Fig. 1.

The adder in Fig. 1 performs 48 logical operations, among them XOR – 11, AND – 11, OR – 4. The optimum adder (9) performs 34 logical operations, among them XOR – 5, AND – 10, OR – 9.

Given that the XOR logic uses four logical operations (Fig. 1, *b*), we can estimate the acceleration rate S of the optimum adder:

$$S = T_1 / T_{opt.} = 59/39 = 1{,}5128 = 51{,}28\ \%,$$

where T_1, T_{opt} is the number of logical operations of the sub-optimal and optimal adders respectively.

6. Discussion of results of research of modeling of the process of addition of binary numbers in the parallel carry adder of the Rademacher NTB by a binary tree in the form of the acyclic graph with two logical operations – AND and XOR

The research shows that computing steps of the directed acyclic graph and carry of the unit from the previous adder bits represent a single object. Computing, arranged in the cascade circuit has logarithmic complexity, and since the acyclic graph provides the cascade circuit, the number of computing steps of the graph optimizes (indicates the minimum sufficient) the number of carries for the operation of addition of multi-bit numbers in the parallel carry adder circuit of the Rademacher NTB. The target function of the optimization process of the number of the adder carries is the equation (7).

Note that the method of empirical construction (including software) of the parallel carry adder does not guarantee the adder structure with a minimum number of carries and, consequently, the adder circuit thus synthesized may require more time for the addition operation.

The relationship between the number of computing steps of the directed acyclic graph and the number of carries in the parallel carry adder circuit indicates the feasibility of comparison of the adder structure with the corresponding directed acyclic graph, and feasibility is a necessity, so the usefulness of this research is that they cause the process of matching the adder structure with the corresponding directed acyclic graph to determine the optimum number of carries for the operation of addition of binary numbers. Thus, the research may be a part of the design technology of electronic circuits of adders because:

– make it clear what is the adder structure;

– teach to operate the adder circuit in the design stage;

– allow predicting the implications of the given adder structure.

Determination of conditions for reducing the computational complexity in terms of the adder computing time, for example to O (n–1) may be promising for further consideration of parallel carry adders.

7. Conclusions

1. It is found that computing of the sum and carry signals in the parallel carry adder of the Rademacher NTB can be justified by the mathematical model in the form of the directed acyclic graph, which is a binary tree.

2. It is revealed that the performance indicator of the directed acyclic graph in the form of a number of computing steps determines the optimum number of carries in the multi-bit parallel carry adder circuit in the Rademacher NTB.

3. It is found that the number of computing steps for the considered models of parallel carry adders (models of 4-bit adders) is equal to the number of bits of binary numbers n. Thus, the complexity of the algorithm for computing the sum and carry signals of the parallel carry adder in the Rademacher NTB is O (n) and is linear – the time of the algorithm increases linearly with n.

4. It is revealed that computing of the sum and carry signal in the parallel carry adder circuit is performed by the logarithmic addition algorithm.

5. It is found that the logic of the optimized adder using the considered mathematical model corresponds to the computing protocol of the parallel carry adder.

References

1. Borisenko, A. A. Remark about Fibonacci of microprocessors [Electronic resource] / A. A. Borisenko. – Academy Trinitarizm, 2011. – Available at: http://www.trinitas.ru/rus/doc/0232/009a/02321223.htm
2. Sajesh, K. Efficient Carry Select Adder Design for FPGA [Text] / K. Sajesh, S. Mohamed // Procedia Engineering. – 2012. – Vol. 30. – P. 449–456. doi: 10.1016/j.proeng.2012.01.884
3. Hiremath, Y. A Novel 8-bit Carry Select Adder using 180nm CMOS Process Technology [Text] / Y. Hiremath // International Journal of Emerging Engineering Research and Technology. – 2014. – Vol. 2, Issue 6. – P. 187–194. – Available at: http://www.ijeert.org/pdf/v2-i6/25.pdf
4. Balasubramanian, P. Mathematical Modeling of Timing Attributes of Self-Timed Carry Select Adders [Text] / P. Balasubramanian, C. Jacob Prathap Raj, S. Anandi, U. Bhavanidevi, N. E. Mastorakis // Recent Advances in Circuits, Systems, Telecommunications and Control. – 2013. – P. 228–243. – Available at: http://www.wseas.us/e-library/conferences/2013/Paris/CCTC/CCTC-34.pdf
5. Chithra, M. 128-bit Carry Select Adder Having Less Area And Delay [Text] / M. Chithra, G. Omkareswari // International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering. – 2013. – Vol. 2, Issue 7. – P. 3112–3118. – Available at: http://www.ijareeie.com/upload/2013/july/35E_128-BIT.pdf
6. Kunitskay, S. Y. Synthesis combiners in binary, ternary excess notation [Text] / S. Y. Kunitskay // Herald ChDTU. – 2015. – Vol. 1. – P. 86–90.
7. Tang, Y. KTV-Tree: Interactive Top-K Aggregation on Dynamic Large Dataset in the Cloud [Text] / Y. Tang, L. Liu, G. Tech, J. Tatemura, H. Hacigumus // IEEE 35th International Conference on Distributed Computing Systems Workshops, 2015. – P. 136–142. – Available at: https://pdfs.semanticscholar.org/cb3e/ae43d0e3465cd52acf73de974bcc374e6665.pdfdoi: 10.1109/icdcsw.2015.32
8. Martyniuk, T. B. Analiz operatsiynogo basis for neyromerezhevih intelektualnih systems [Text] / T. B. Martyniuk, A. B. Kozhem'yko, N. O. Denysiuk, T. Y. Pozdniakova // Informatsiyni tehnologii that Komp'yuterniy inzheneriya. – 2015. – Vol. 2. – P. 83–87.

9. Tsmots, I. The modified method and structure of the VLSI-device group for summing neyroelementa [Text] / I. Tsmots, O. Skorokhoda, B. Balych // Bulletin of the National University "Lviv Polytechnic": Computer Science and Information Technology. – 2012. – Vol. 732. – P. 51–57. –Available at: http://ena.lp.edu.ua:8080/bitstream/ntb/14865/1/9_Tsmots_51_57_732.pdf
10. Wu, C. A survey of advancements in nucleic acid-based logic gates and computing for applications in biotechnology and biomedicine [Text] / C. Wu, S. Wan, W. Hou, L. Zhang, J. Xu, C. Cui et. al. // Chem. Commun. – 2015. – Vol. 51, Issue 18. – P. 3723–3734. doi: 10.1039/c4cc10047f
11. Seelig, G. Time-Complexity of Multilayered DNA Strand Displacement Circuits. In: DNA computing and molecular programming [Text] / G. Seelig, D. Soloveichik. – Lecture Notes in Computer Science, 2009. – P. 144–153. – Available at: http://www.dna.caltech.edu/Papers/CRN_circuit_complexity.pdf doi: 10.1007/978-3-642-10604-0_15
12. Hamaiun, V. P. On the development of computational structures mnogooperandnyh [Text] / V. P. Hamaiun // Control systems and machines. – 1990. – Vol. 4. – P.31–33.
13. Gamajun, V. P. Teoretychni osnovy, algorytmy ta struktury bagatooperandnoi' obrobky [Text] / V. P. Gamajun. – NAN Ukrai'ny In-t kibernetyky im. V. M. Glushkova, 1999. – 33 p.
14. Martyniuk, T. B. Methods and means of parallel transformation vector data sets [Text] / T. B. Martyniuk, V. V. Homyuk. – Vinnitsa: «UNIVERCUM- Vinnitsa», 2005. – 202 p.
15. Martyniuk, T. B. Rekursyvni algorytmy bagatooperandnoi obrobky informacii [Text] / T. B. Martyniuk. – Vinnitsa: «UNIVERCUM- Vinnitsa», 2000. – 216 p.
16. Class ECE6332 Fall 12 Group-Fault-Tolerant Reconfigurable PPA [Electronic resource]. – Available at: http://venividiwiki.ee.virginia.edu/mediawiki/index.php/ClassECE6332Fall12Group-Fault-Tolerant_Reconfigurable_PPA