UDC 681.325
DOI: 10.15587/2312-8372.2018.133694

Solomko M.

OPTIMIZATION OF THE ACYCLIC ADDERS OF BINARY CODES

Об'єктом дослідження є префіксна модель обчислення сигналів суми і перенесення у схемі паралельного суматора з паралельним способом перенесення. Одним з найбільш проблемних місць префіксної моделі є процес вироблення сигналів суми і перенесення, у якому початок обчислення префікса передбачено з першого розряду схеми. Це приводить, у підсумку, до надлишкового нагромадження і ускладнення апаратної частини пристрою.

У ході дослідження використовувалась математична модель обчислення сигналів суми і перенесення у схемі паралельного суматора, що ґрунтується на властивостях направленого ациклічного графа з двома типовими операціями.

Отримано зменшення складності логічної структури суматора бінарних кодів, зменшення глибини схеми та зменшення загальної протяжності з'єднувальних проводів. Це пов'язано з тим, що запропонований метод обчислення сигналів суми і перенесення має ряд особливостей синтезу схеми пристрою, зокрема застосування математичної моделі, що ґрунтується на властивостях ациклічного графа, розраховано на:

– процес послідовного (для молодших розрядів схеми пристрою) і паралельного обчислення сигналів суми і перенесення, що, у підсумку, дає зменшення складності апаратної частини пристрою та не збільшує глибину схеми;

– співставлення числа обчислювальних кроків орієнтованого ациклічного графа з числом перенесень одиниці до старшого розряду у схемі суматора, що дозволяє встановлювати оптимальне число обчислювальних кроків для структури пристрою.

Завдяки цьому забезпечується можливість отримання оптимальних значень показників складності структури та глибини схеми суматора. Зв'язок між числом обчислювальних кроків орієнтованого ациклічного графа і числом перенесень у схемі паралельного суматора з паралельним способом перенесення вказує на доцільність співставлення структури суматора з відповідним орієнтованим ациклічним графом.

У порівнянні з аналогічними відомими структурами 8-bit префіксних суматорів це забезпечує збільшення показника якості 8-bit ациклічних суматорів, наприклад, за енергоспоживанням, площею чіпа, у залежності від обраної структури, на 10–40 %.

Ключові слова: *ациклічна модель, префіксна модель, направлений ациклічний граф, Ling Adder, Kogge-Stone Adder, Brent-Kung Adder.*

1. Introduction

The adder of binary codes is present in most digital electronic circuits, including digital signal processors (DSPs) and is one of the means of microprocessor data processing. The performance of the addition operation in the positional number system depends on the way the unit is transferred to the high-order bit. A variant of such transfer is, in particular, the technology of prefix adding of numbers [1–4].

In this paper, applications of acyclic models for calculating adding and transport signals [5] for the synthesis of parallel 8-bit adder of binary codes is presented. This gives a new apparatus for synthesizing parallel multi-bit adders with a parallel transfer method for their application in digital technologies.

Methods of arithmetic operations are realized by gate circuits of functional elements in bases consisting of functions of the algebra of logic. From the structure of the adder, the speed of the digital device depends on its reliability and energy saving. In this connection, minimization of complexity and depth of logic circuits is one of the central and practically important problems in this theory, which arises in the design of digital devices.

Processor evolution is the result of relentless optimization, so the research remains topical, in particular, on the improvement of such factors – manufacturing technologies, structural realization, power consumption, cost of digital devices.

2. The object of research and its technological audit

The object of solving the problem of synthesizing the circuit of the adder of binary codes is the acyclic model of one-stage calculation of adding and carry signals, based on the properties of a directed acyclic graph with two typical operations (Fig. 1, 2).

The acyclic model is designed for the logical structure of the adder with a series-parallel method of calculating the prefix, which, in the end, leads to a reduction in the complexity of the device hardware. The mathematical apparatus of the directed acyclic graph allows one to unambiguously obtain the values of the adding and carry signals in one calculation step, so the latter is able to effectively replace the three-stage prefix model for computing the adding and carry signals (Fig. 3). This expands the apparatus for synthesizing arithmetic devices for their application in digital technologies.

ISSN 2226-3780

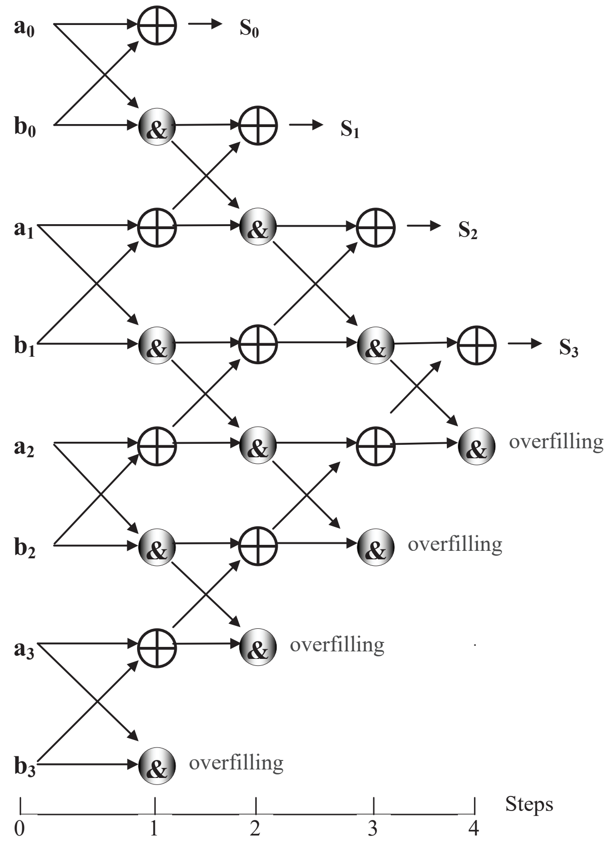


Fig. 1. Oriented acyclic graph – model of the computational circuit of parallel 4-bit acyclic adder with parallel transfer method

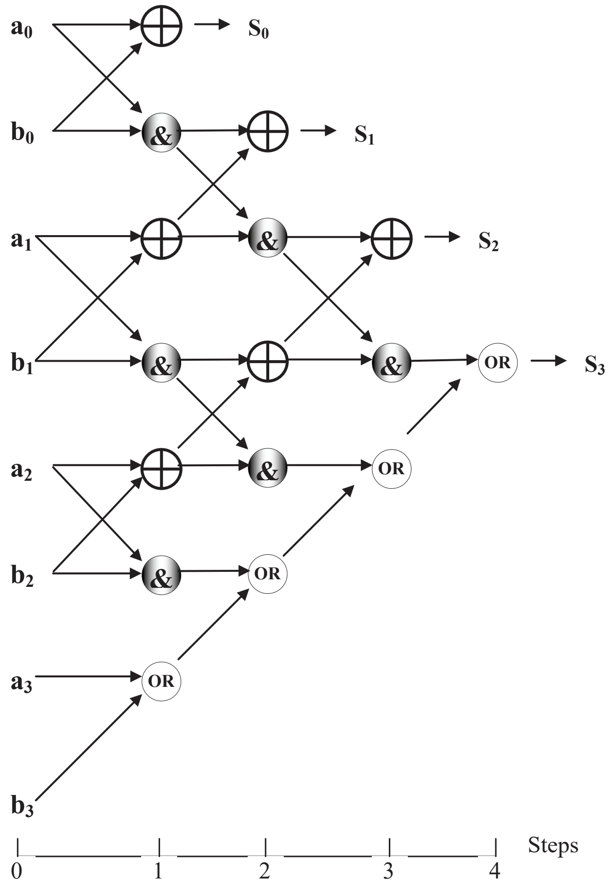


Fig. 2. Oriented acyclic graph – model of the computing circuit of parallel 4-bit acyclic adder with logical elements OR in the last bit

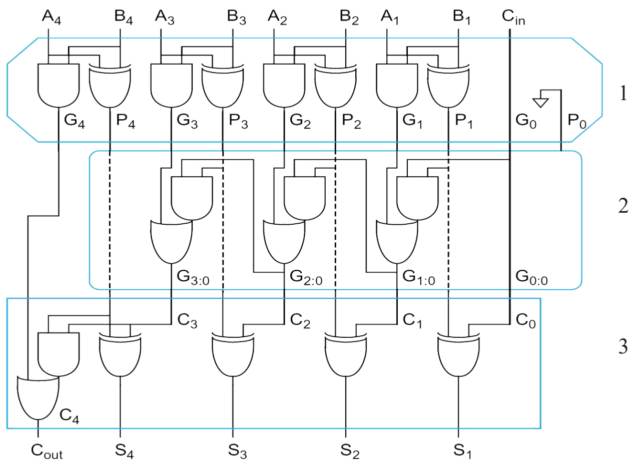


Fig. 3. Model of the parallel prefix adder:
1 – organizational logic; 2 – group logic; 3 – binary code addition logic

The acyclic model of the adder is represented by two typical operations – AND and XOR, admits ways of applying the transfer condition of the unit to the high order (1), which ultimately leads to the optimal complexity of the arithmetic device circuit.

$$p_i = a_i \vee b_i \;\text{ or }\; p_i = a_i + b_i. \tag{1}$$

The acyclic model of an arithmetic device is able to support aggregated structures for calculating adding and carry signals, by combining with other apparatus of calculation methods, in particular with the Ling transfer logic.

The relative lack of acyclic models of calculating adding and carry signals is currently associated with a small volume of theoretical developments in this direction, so the prospect of the method is based on practical chances of synthesizing the optimal structure of an arithmetic device.

3. The aim and objectives of research

The aim of research is synthesis of 8-bit optimal parallel acyclic adders of binary codes.

To achieve this aim, it is necessary to solve the following tasks:

1. To establish the adequacy of the mathematical model on the basis of an oriented acyclic graph with two typical operations.
2. To estimate the dynamics of increasing the depth of the circuit of a parallel acyclic adder with increasing the bit capacity of the circuit.
3. To carry out a comparable analysis of the complexity of the structure and speed of the adders obtained using acyclic and prefix models for calculating adding and carry signals.

4. Research of existing solutions of the problem

The cascade circuit, as a computational mechanism in the composition of the prefix adder model, which uses the logical structure of the three-step computation of adding and carry signals (Fig. 3), is presented in [6]. Let's note that the acyclic model for calculating adding and carry signals (Fig. 1, 2) is calculated for the logical structure of the adder with a sequentially parallel method of calculating the prefix and uses the structure of a one-step computation. Thus, the prefix and acyclic models are distinguished by objects-they have different principles (principles) for computation, and so they have different capabilities in terms of computing speed, chip area and energy saving.

To improve the computational efficiency in [7], prefix structures of Kogge-Stone and Ladner-Fischer are investigated. A comparison with other calculation circuits is presented, in particular with RCA, Carry Skip Adder (CSA). The project uses the Xilinx-ISE tool.

Modified Parallel Prefix Han-Carlson Adder are presented in [8], which uses various stages of synthesis of Brent-Kung and Kogge-Stone prefix structure, which makes it possible to reduce the complexity of the adder design.

Methods for reducing the delay in calculating adding and carry signals in the adder circuit using the parallel prefix structure are discussed in [9], since such structure precompensates the transfers. The structures of Kogge-Stone and Brent-Kung prefix are considered. The simulation and synthesis process is performed using the model sim6.4b, Xilinx ISE9.2i.

A hybrid prefix architecture for the synthesis of 8-, 16- and 32-bit parallel adders is presented in [10]. Comparisons of delay, energy saving, the number of computational nodes with classical prefix structures are conducted. The comparison results show a smaller delay and energy consumption in the proposed computing structures. To simulate the adder on the technology of 180 nm and 130 nm, the Tanner EDA tool is used.

Hybrid prefixed architecture of Han Carlson Adder to reduce power consumption and delay in parallel prefix adding (PPA) is considered in [11]. Comparison with other prefix structures is made.

In [12], the development and comparison of high-speed supplementary elements of the prefix, such as Kogge-Stone, Brent-Kung, Sklansky and Ling, are presented. It is revealed that the structure of Kogge-Stone-Ling is more effective than other prefix structures. Design uses CMOS logic. Design and simulation is performed using 65-nm technology.

It is noted in [13] that each type of parallel prefix adder has its advantages and disadvantages and is selected in accordance with the requirements of the claimed design. In this paper, we mainly study two types of structures that contain combined trees and the Kogge-Stone adder and compare them. The projects are implemented on the Xilinx Virtex 5 FPGA. It is found that combined trees occupy a smaller area than Kogge-Stone structure.

A method for minimizing power consumption by obtaining the optimal structure of the parallel Kogge-Stone and Ladner-Fischer adder for 32-, 64-, 128, and 256-bit for 45 nm CMOS technology is investigated in [14]. The results of the study show a decrease in energy consumption by 22–50 % for the optimal structure of the adder with unchanged computational performance.

It is noted in [15] that prefix structures are efficient for the implementation of ASIC, but these advantages are not enough for the FPGA development. There are various types of parallel prefixes for comparison and selection. Verilog HDL, Xilinx ISE13.2 software and Cadence RTL compiler are used to develop applications. Among all Kogge-Stone applications, the adder provides better performance in the ASIC implementation, but it is not suitable for FPGA development. In order to make it suitable for the FPGA implementation of Kogge-Stone, the adder is modified with the help of fast logic, it ensures optimal performance.

Patent [16] represents the pyramidal structure of a combination adder with vertical and horizontal information links between single-bit binary half-adders. The technical result of the patent is expansion of the device functionality, reducing hardware complexity due to the introduction of high-speed single-bit half-adder, which contain three logic elements, and improve the device performance.

In contrast to the publications reviewed in this paper, the object for synthesizing the structure of adders of binary codes is the acyclic model, the description of which is given in Section 2.

5. Methods of research

5.1. Prefix model of binary codes adder. The prefix sum or simply the prefix of the sequence of numbers x_0, x_1, x_2, ..., x_n is another sequence of bits y_0, y_1, y_2, ..., y_n, which is calculated from the original one according to this principle:

$$y_0 = x_0,$$

$$y_1 = x_0 + x_1,$$

$$y_2 = x_0 + x_1 + x_2,$$

$$\ldots$$

$$y_n = x_0 + \ldots + x_{n-1} + x_n.$$

In the cascade adder, the carry bit c_i is calculated at time i. The values of a_i and b_i are known from the beginning. In some cases, they specify the carr bit c_i:

if $a_i = b_i = 0$, then $c_i = 0$ (carry is «kill»),
if $a_i = b_i = 1$, then $c_i = 1$ (carry is «generated»).

However, if one of the bits a_i or b_i is 1 and the other is 0, then c_{i-1} has an essential content for the carry, that is:

if $a_i \neq b_i$, then $c_i = c_{i-1}$ (carry is propagated).

Each category, therefore, corresponds to one of the three carry statuses: k (kill), g (generate), or p (propagate). This type is known in advance, which allows to reduce the time of the addition operation.

Since the type of carry for neighboring bits ((i–1)-th and i-th) is known, it is possible to determine the type of carry to combine them, considering c_{i-1} as the input bit, and c_{i+1} as the output one. Thus, getting information about how the carry bit changes at each step, it is possible to calculate what happens in two steps, that is, how depends c_{i+1} on c_{i-1}. If the i-th bit has a carry type p, then the carry type for the union is the same as the type of the (i–1)-th bit (Table 1).

Table 1

Table of operation Θ

Θ		FA_i		
		k	p	g
FA_{i-1}	k	k	k	g
	p	k	p	g
	g	k	k	g

Table 1 can be considered as the definition of an operation (a composition of carry types) on the set $\{k, p, g\}$; it is denoted by the symbol Θ and is associative. The operation Θ determines the carry type of some part of the adder if the carry types of its individual bits are known.

Let's denote by x_i the carry type of transfer in the i-th bit:

$$x_i = \begin{cases} k, \text{ if } a_i = b_i = 0; \\ g, \text{ if } a_i = b_i = 1; \\ p, \text{ if } a_i \neq b_i. \end{cases}$$

Then the dependence, for example, bit c_7 on c_4 is determined by the composition:

$$x_5 \Theta x_6 \Theta x_7.$$

Since the carry of a unit to a zero bit from the least significant bits is not carried out, conditionally $x_0 = k$ is assumed. Then the transfer at the output of the i-th bit is determined by the composition $x_0 \Theta x_1 ... \Theta x_i : c_i = 0$ if the composition is k, and $c_i = 1$ if the composition is g. The value of p for the composition is impossible, because for this all members must be equal to p, and this is not true for x_0.

More formally, it is written like this. Let's take $y_0 = k$ and define $y_1, y_2, ..., y_n$ in the form:

$$y_i = x_i \Theta y_{i-1} = x_0 \Theta x_1 \Theta ... \Theta x_i.$$

Then the elements $y_1, y_2, ..., y_n$ are prefixes of the expression $x_0 \Theta x_1 ... \Theta x_n$.

Thus, the calculation of the sum of the transmission bits of c_i unit in a cascade adder can be reduced to the calculation of prefixes.

The parallel prefix method emerged as the fastest process of adding unit-shifting bits in binary code addition operations for high-performance data processing systems, which original ideas can be found in early works [1–4]. Further publications [17–20] confirmed this assessment of such technologies.

When determining the transfer in a parallel multi-bit adder, the principle of obtaining a prefix sum on the sequence of numbers $x_0, x_1, x_2, ..., x_n$ extends to obtaining the prefix sum on the sequence of pairs of transfer functions of the unit $(g_0, p_0), (g_1, p_1), ..., (g_{k-1}, p_{k-1})$ (Table 2):

– carry generation function:

$$g_i = a_i b_i;$$

– carry propagation function:

$$p_i = a_i \oplus b_i. \qquad (2)$$

The transfer of a unit at the output of the i-th bit is determined by the composition:

$$(g_0, p_0)\Theta(g_1, p_1)\Theta ... \Theta(g_{k-2}, p_{k-2})\Theta(g_{k-1}, p_{k-1}).$$

The operator defining the transfer Θ is associative, but not commutative:

$$[(g_1, p_1)\Theta(g_2, p_2)]\Theta(g_3, p_3) = (g_1, p_1)\Theta[(g_2, p_2)\Theta(g_3, p_3)].$$

The carry propagation function (1) can more accurately be called a function of the condition of carry of the unit to the high-order bit. Often the carry propagation function (2) (carry condition) is defined as a function (1).

If $p_i = 1$, then the propagation of the unit to the subsequent bits will be possible, in the case when $p_i = 0$ the propagation of the unit to subsequent bits is impossible (Fig. 4).

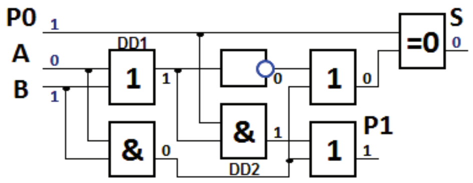


Fig. 4. Demonstration of the function of the carry condition of the unit to the high-order bit

Taking into account Fig. 4 it is possible to see that in the case when the DD1 element, which realizes the carry condition function $p = A \vee B$ on the output, will get the value of the logical unit $(p = 1)$, it will be possible to transfer the unit P0 to the element DD2 (output DD2 = output DD1(1)P0(1) = 1). In the case where the output of the DD1 element is a logical zero, the transfer of the unit P0 to the DD2 element will be impossible (output DD2 = output DD1(0)P0(1) = 0).

The condition of carry of a unit to the high-order bit can be demonstrated by the operation of adding binary numbers to a column (Table 3).

From the variants of adding single-bit binary numbers it is possible to see that if A∨B = 1, the unit from the least significant bit P_0 is carried to the high-order (second) bit of the sum ($P_1 = 1$). If A∨B = 0, the sum remains single-bit, the unit from the least significant bit P_0 to the high-order (second) bit of the sum is not carried ($P_1 = 0$). A similar condition for the carry of a unit to the high-order bit is preserved when adding multi-bit binary numbers.

Parallel prefix adders have been used as the most efficient circuits in the operations of adding binary codes of digital systems. Their regular structure and high performance make them especially attractive for the creation of SLIC (super-large integrated circuits).

Table 2

Calculations of the prefix sum on the sequence of pairs of carry functions of the unit $(g_0, p_0), (g_1, p_1), \dots, (g_{k-1}, p_{k-1})$

Given	(g_0, p_0)	(g_1, p_1)	…	(g_{k-2}, p_{k-2})	(g_{k-1}, p_{k-1})
Find	$(g[0,0], p[0,0])$	$(g[0,1], p[0,1])$	…	$(g[0,k-2], p[0,k-2])$	$(g[0,k-1], p[0,k-1])$
Carry of a unit at the output of the i-th level	c_1	c_2	…	c_{k-1}	c_k

Table 3

Adding binary numbers to a column

Possible options for adding				
The unit from the lower bit (P_0)	1	1	1	1
Number A	0	0	1	1
Number B	0	1	0	1
Sum	1	10	10	11

These adders provide a theoretical basis for compromises in terms of delay, area and power, in order to provide a wide range of services in the design process.

The parallel prefix adder (PPA) uses the logical structure shown in Fig. 3, which provides for the calculation in three stages:

– pre-processing (pre-processing or initialization phase):

$$g_i = a_i b_i,$$

$$p_i = a_i \oplus b_i;$$

– prefix calculation (signal generation network):

$$G_{[i:k]} = G_{[i:j]} + P_{[i:j]} G_{[j-1:k]},$$

$$P_{[i:k]} = P_{[i:j]} P_{[j-1:k]};$$

– post-processing (after prefix processing or summation):

$$C_{i+1} = G_{[i:0]} + P_{[i:0]} \cdot C_0,$$

$$S_i = p_i \oplus C_i.$$

Prefix architectures for the calculation of the carry signal are known, for example:

– 1966: Ling adder;
– 1973: Kogge-Stone adder;
– 1980: Ladner-Fisher adder;
– 1982: Brent-Kung adder;
– 1987: Han Carlson adder;
– 1999: S. Knowles. (3)

Among the known prefix structures, the main one is the parallel prefix adder with the Ling and Kogge-Stone prefix carry structure, which is the final case of a large list of summing circuits, each of which is unique with its minimal logical capacity property.

Ling adder (Fig. 5) [21–23] has the least delay compared to other methods of prefix carry, but requires relatively larger chip area and power consumption.

The drawbacks of the prefix model for calculating adding and carry signals include:

– the process of parallel computation of the prefix by architectures (3) provides for the beginning of the calculation from the first bit of the circuit, leads, in the end, to excessive accumulation and complications of the hardware part of the device;

– the principle of three-stage generation of the adding and carry signal (Fig. 3), which specifies a certain complexity of such calculation, in particular complicates the didactics of the method;

– the parallel one-to-many structure of a prefix adder generally has fewer links and may occupy several ranks of the circuit. This, at least from the technological side, is not an effective indicator in comparison with the acyclic computation model. And from this, there may be a contradiction between the requirements for the speed of calculating adding and carry signals and energy consumption, as well as the area of the device, in particular in the SLIC design system.

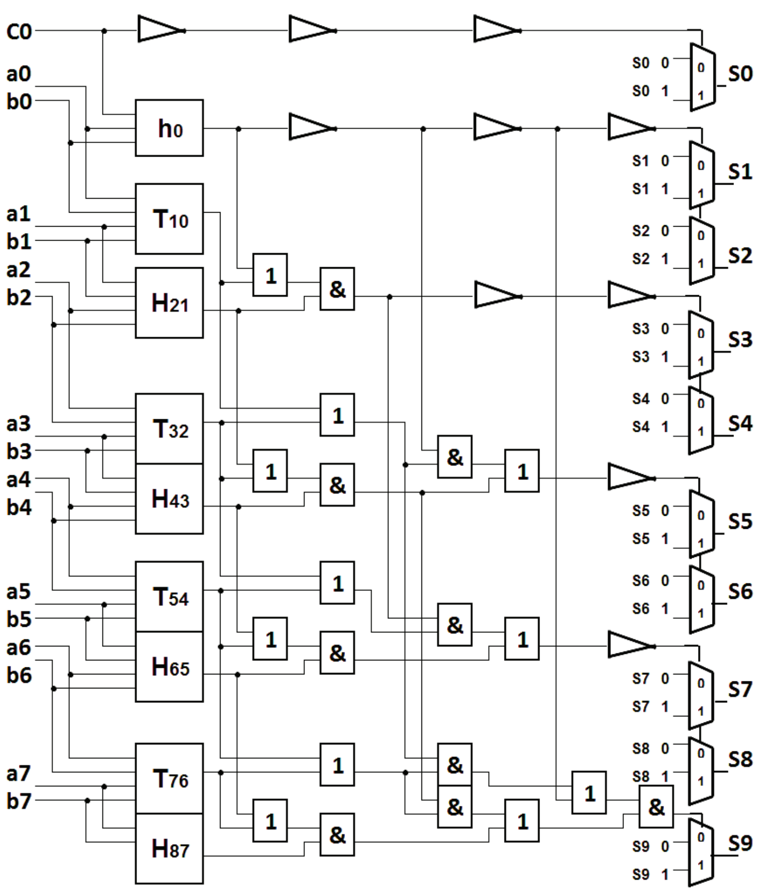


Fig. 5. 8-bit Ling Adder [21–23]

5.2. The acyclic model of binary codes adder. The calculation principle for the model of an acyclic adder is determined by an acyclic graph, when adjacent pairs of terms are added simultaneously, and then their sums are added (Table 4). This is a doubling algorithm (or a logarithmic addition algorithm), which ultimately gives a one-step method for generating adding and carry signals.

Table 4

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

Steps	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
1	x_1+x_2		x_3+x_4		x_5+x_6		x_7+x_8	
2	$x_1+x_2+x_3+x_4$				$x_5+x_6+x_7+x_8$			
3	$x_1+x_2+x_3+x_4+x_5+x_6+x_7+x_8$							

Bitwise addition of binary codes is possible with the help of the doubling algorithm, similar to the process of multi-operative summation. If $n=2^k$, where n – the number of addends, then the doubling algorithm consists of *k* steps: in the first stage, *n*/2 additions are executed, on the second stage – *n*/4, ..., on the last stage – one addition. The number of steps k is determined by the formula:

$$k = \log_2 n. \tag{4}$$

This version of the multi-operand addition is realized using an acyclic graph or a cascade circuit [24].

Using the procedure of multi- operand addition using a cascade circuit, it is easy to see that for the process of parallel addition of binary codes, data pairs here will be bits of the same name bits, for each of which the adding and carry signal is calculated. Further, similarly to the procedure of multi-operand addition, all the obtained sums of the same-named bit pairs of binary codes, with their specificity, are also divided into pairs and addition of pair values, etc. is performed again.

As a result, the value of the high-order bit of the sum of binary codes can be compared with the value of the total amount for multi-operand additions. In addition to the sum of the high-order bit, in the process of parallel addition of binary codes, intermediate results automatically appear as sums of the previous bits of binary codes.

The computational circuit for the parallel addition of 4-bit binary codes can be defined by an oriented acyclic graph (Fig. 1), which is a binary tree, where in particular the following parameters are accepted:

k – the number of steps in time;

ω – the total number of operations of the algorithm;

τ – the execution time of one step;

$T=\tau\cdot k$ – the execution time of the algorithm;

L – number of transaction types, etc.

The computer circuit in Fig. 1 is also a model of a 4-bit acyclic parallel adder of binary codes with a parallel carry method.

The model of the computational circuit of the acyclic adder in Fig. 1 uses two logical operations – AND and XOR, the number of computational steps in it is equal to the bit depth of the binary codes. For example, to parallel add 4-bit binary codes, it is necessary four steps (Fig. 1).

The model of a 4-bit acyclic adder of binary codes with logical elements OR in the last bit is shown in Fig. 2.

Application of acyclic models is calculated on:

– the process of sequential (for lower-order device circuits) and parallel calculation of adding and carry signals, which, in the end, leads to a reduction in the complexity of the hardware of the device and does not increase the depth of the circuit;

– the establishment of the optimal number of computational steps.

In [24] it is shown that the number of computational steps determines the minimally sufficient number of carries in the circuit of an acyclic adder. In the case where the synthesized adder has received a larger number of carries than the number of computational steps of the corresponding oriented acyclic graph, then such adder will not be optimal relative to the number of computational operations.

The logical equations of an optimized 4-bit adder with a carry number (4), for example:

$$S_0 = a_0 \oplus b_0;$$

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

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

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

The variant of the circuit of a 4-bit acyclic adder, which is determined by the computational model in Fig. 2 is shown in Fig. 6.

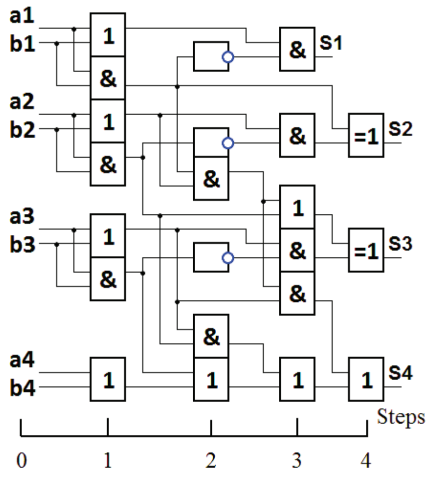


Fig. 6. 4-bit acyclic adder of binary codes

The model of the computational circuit of a parallel 8-bit acyclic adder of binary codes will be similar to the computational circuits shown in Fig. 1, 2, with the difference that here the number of computational steps will be equal to eight. The first four logical equations of an 8-bit acyclic adder can, for example, be:

$$S_0 = a_0 \oplus b_0;$$

$$S_1 = a_0 b_0 \overline{a_1}\,\overline{b_1} + \overline{b_0} a_1 \overline{b_1} + \overline{a_0} a_1 \overline{b_1} + \overline{b_0}\,\overline{a_1} b_1 + \overline{a_0}\,\overline{a_1} b_1 + a_0 b_0 a_1 b_1;$$

$$S_2 = a_0 b_0 a_1 \overline{a_2}\,\overline{b_2} + a_0 b_0 b_1 \overline{a_2}\,\overline{b_2} + a_1 b_1 \overline{a_2}\,\overline{b_2} + \overline{a_1}\,\overline{b_1} a_2 \overline{b_2} +$$
$$+ \overline{b_0}\,\overline{b_1} a_2 \overline{b_2} + \overline{a_0}\,\overline{b_1} a_2 \overline{b_2} + \overline{b_0}\,\overline{a_1} a_2 \overline{b_2} + \overline{a_0}\,\overline{a_1} a_2 \overline{b_2} + \overline{a_1}\,\overline{b_1}\,\overline{a_2} b_2 +$$
$$+ \overline{b_0}\,\overline{b_1}\,\overline{a_2} b_2 + \overline{a_0}\,\overline{b_1}\,\overline{a_2} b_2 + \overline{b_0}\,\overline{a_1}\,\overline{a_2} b_2 + \overline{a_0}\,\overline{a_1}\,\overline{a_2} b_2 +$$
$$+ a_0 b_0 a_1 a_2 b_2 + a_0 b_0 b_1 a_2 b_2 + a_1 b_1 a_2 b_2;$$

$$S_3 = a_0 b_0 a_1 a_2 \overline{a_3}\,\overline{b_3} + a_0 b_0 b_1 a_2 \overline{a_3}\,\overline{b_3} + a_1 b_1 a_2 \overline{a_3}\,\overline{b_3} +$$
$$+ a_0 b_0 a_1 b_2 \overline{a_3}\,\overline{b_3} + a_0 b_0 b_1 b_2 \overline{a_3}\,\overline{b_3} + a_1 b_1 b_2 \overline{a_3}\,\overline{b_3} + a_2 b_2 \overline{a_3}\,\overline{b_3} +$$
$$+ \overline{a_2}\,\overline{b_2} a_3 \overline{b_3} + \overline{a_1}\,\overline{b_1}\,\overline{b_2} a_3 \overline{b_3} + \overline{b_0}\,\overline{b_1}\,\overline{b_2} a_3 \overline{b_3} + \overline{a_0}\,\overline{b_1}\,\overline{b_2} a_3 \overline{b_3} +$$
$$+ \overline{b_0}\,\overline{a_1}\,\overline{b_2} a_3 \overline{b_3} + \overline{a_0}\,\overline{a_1}\,\overline{b_2} a_3 \overline{b_3} + \overline{a_1}\,\overline{b_1}\,\overline{a_2} a_3 \overline{b_3} + \overline{b_0}\,\overline{b_1}\,\overline{a_2} a_3 \overline{b_3} +$$
$$+ \overline{a_0}\,\overline{b_1}\,\overline{a_2} a_3 \overline{b_3} + \overline{b_0}\,\overline{a_1}\,\overline{a_2} a_3 \overline{b_3} + \overline{a_0}\,\overline{a_1}\,\overline{a_2} a_3 \overline{b_3} +$$
$$+ \overline{a_2}\,\overline{b_2}\,\overline{a_3} b_3 + \overline{a_1}\,\overline{b_1}\,\overline{b_2}\,\overline{a_3} b_3 + \overline{b_0}\,\overline{b_1}\,\overline{a_2}\,\overline{a_3} b_3 + \overline{a_0}\,\overline{b_1}\,\overline{b_2}\,\overline{a_3} b_3 +$$
$$+ \overline{b_0}\,\overline{a_1}\,\overline{b_2}\,\overline{a_3} b_3 + \overline{a_0}\,\overline{a_1}\,\overline{b_2}\,\overline{a_3} b_3 + \overline{a_1}\,\overline{b_1}\,\overline{a_2}\,\overline{a_3} b_3 + \overline{b_0}\,\overline{b_1}\,\overline{a_2}\,\overline{a_3} b_3 +$$
$$+ \overline{a_0}\,\overline{b_1}\,\overline{a_2}\,\overline{a_3} b_3 + \overline{b_0}\,\overline{a_1}\,\overline{a_2}\,\overline{a_3} b_3 + \overline{a_0}\,\overline{a_1}\,\overline{a_2}\,\overline{a_3} b_3 + a_0 b_0 a_1 a_2 a_3 b_3 +$$
$$+ a_0 b_0 b_1 a_2 a_3 b_3 + a_1 b_1 a_2 a_3 b_3 + a_0 b_0 a_1 b_2 a_3 b_3 + a_0 b_0 b_1 b_2 a_3 b_3 +$$
$$+ a_1 b_1 b_2 a_3 b_3 + a_2 b_2 a_3 b_3.$$

Circuits of 8-bit acyclic adders are shown in Fig. 8, 10, 12. With the increase in the bit capacity of the acyclic adder (16-, 32-, 64-bit ...), the number of computational steps will be determined by the logarithmic law (Fig. 7).

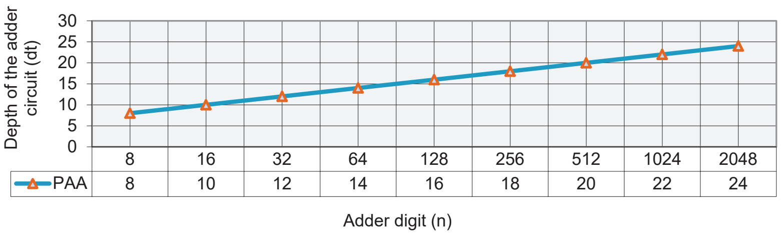


Fig. 7. Dynamics of increasing the depth of the circuit of the acyclic adder (PAA)

6. Research results

6.1. 8-bit acyclic adder with a depth of 8 elements. To ensure the same comparison conditions, let's represent the circuits of prefix (PPA) and acyclic (PAA) adders with logical elements OR in the last bit.

Fig. 8 shows an acyclic 8-bit PAA with logical elements OR in the last bit and depth of the circuit 8 typical 2-input elements. The complexity of the circuit in Fig. 8 is 77 discrete elements.

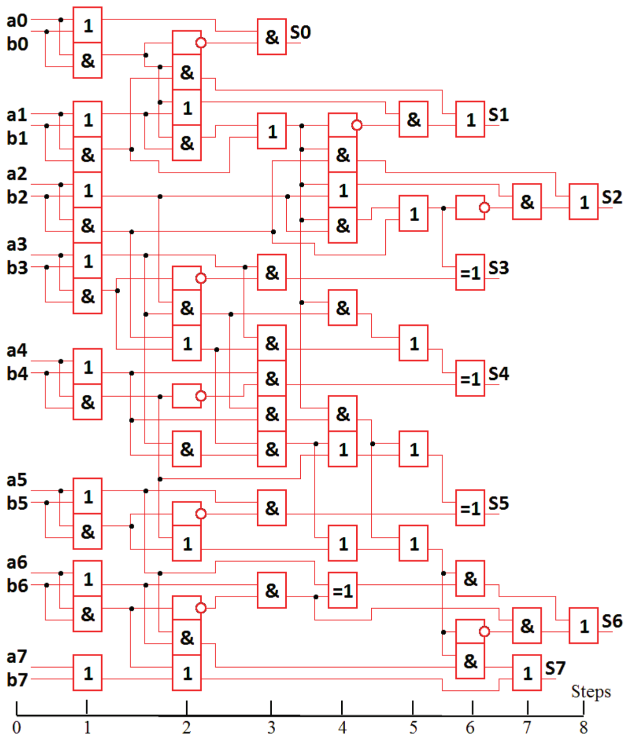


Fig. 8. Acyclic 8-bit PAA with logical elements OR in the last bit and depth of circuit of 8 typical 2-input elements

Prefix 8-bit Ling Adder [21–23] with logical elements OR in the last bit is shown in Fig. 9. The circuit determines the depth of the adder circuit in Fig. 9 separated by a thick line and is accompanied by the numbering of logical elements along this chain. Thus, the depth of the 8-bit Ling Adder circuit [21–23] PPA (Fig. 9) is 8 typical logic elements, the complexity of the circuit is 109 elements. In accordance with the structure of the prefix model (Fig. 3), the third stage of calculating the sum signal in the adder in Fig. 9 is realized by a multiplexer in each bit of the circuit.

The computing process of the 8-bit Ling Adder PPA adder (Fig. 9) uses such logical operations: XOR – 13, AND – 27, OR – 24, Inventor – 6. 8-bit PAA adder (Fig. 8) uses: XOR – 9, AND – 19, OR – 19, Inventor – 3. Given that the logic of the XOR element uses four logic elements, including Inventor, it is possible to estimate the quality score S (for example, on energy saving) of the 8-bit PAA adder (Fig. 8):

$$S=\frac{T_1}{T_2}=\frac{109}{77}=1.4156=41.56\ \%,$$

where T_1, T_2 – the number of discrete logic elements 8-bit Ling Adder PPA and 8-bit PAA, respectively.

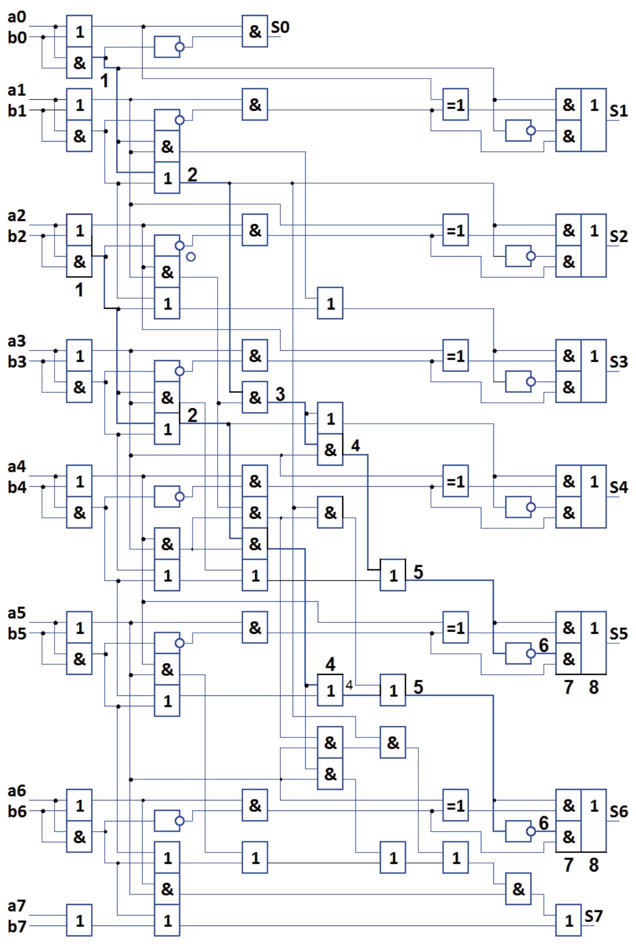


Fig. 9. Prefix 8-bit Ling Adder PPA [21–23] with logical elements OR in the last bit

6.2. 8-bit acyclic adder with a depth of 9 elements. Fig. 10 shows an acyclic 8-bit PAA with logical elements OR in the last bit and the depth of circuit of 9 of the typical 2-input elements. The complexity of the circuit in Fig. 10 is 72 discrete elements.

The prefix 8-bit Kogge-Stone PPA [24] with the logical elements OR in the last bit is shown in Fig. 11. The circuit determines the depth of the adder circuit in Fig. 11 separated by a thick line and is accompanied by the numbering of logical elements along this chain. Thus, the depth of the 8-bit Kogge-Stone PPA circuit (Fig. 11) is 9 typical logic elements, the complexity of the circuit is 90 elements.

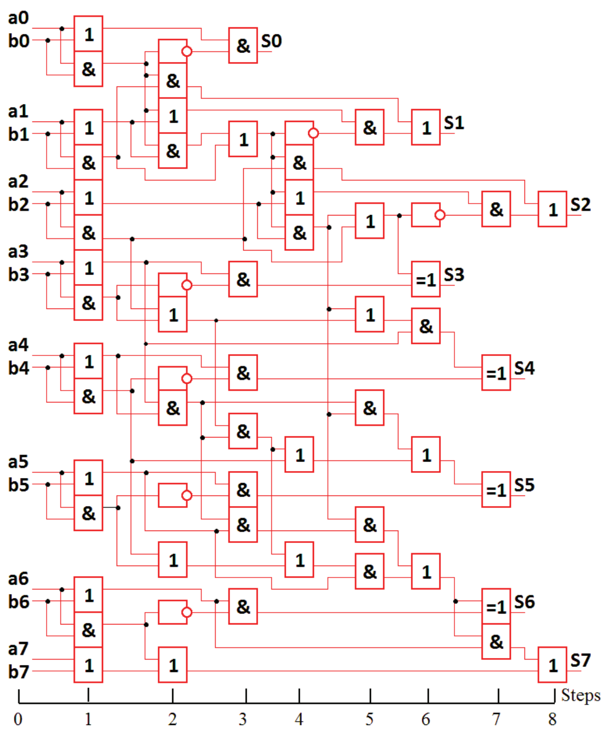


Fig. 10. Acyclic 8-bit PAA with logical elements OR in the last bit and depth of circuit of 9 of the typical 2-input elements

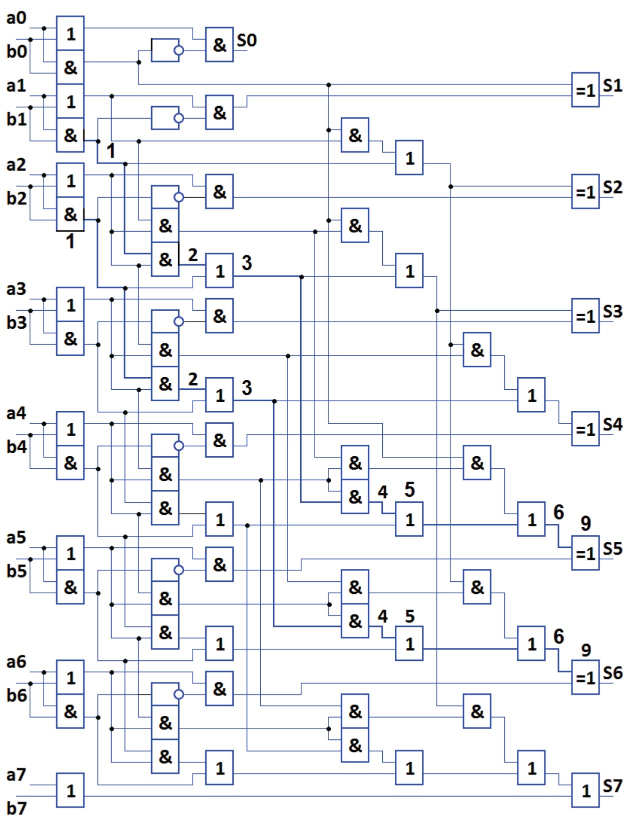


Fig. 11. Prefix 8-bit Kogge-Stone PPA with logical OR elements in the last bit [24]

The computing process of the 8-bit Kogge-Stone PPA adder (Fig. 11) uses such logical operations: XOR – 13, AND – 22, OR – 26. The 8-bit PAA adder (Fig. 10) uses: XOR – 9, AND – 16, OR – 18, Inventor – 2. The quality indicator S (for example, on energy saving) of the 8-bit PAA adder (Fig. 10) is as follows:

$$S=\frac{T_1}{T_2}=\frac{90}{72}=1.25=25\ \%,$$

where T_1, T_2 – the number of discrete logic elements of 8-bit Kogge-Stone PPA and 8-bit PAA, respectively.

6.3. 8-bit acyclic adder with a depth of 10 elements. Fig. 12 shows the acyclic 8-bit PAA with the logical elements OR in the last bit and the depth of the circuit of 10 of the typical 2-input elements. The complexity of the circuit in Fig. 12 is 66 discrete elements.

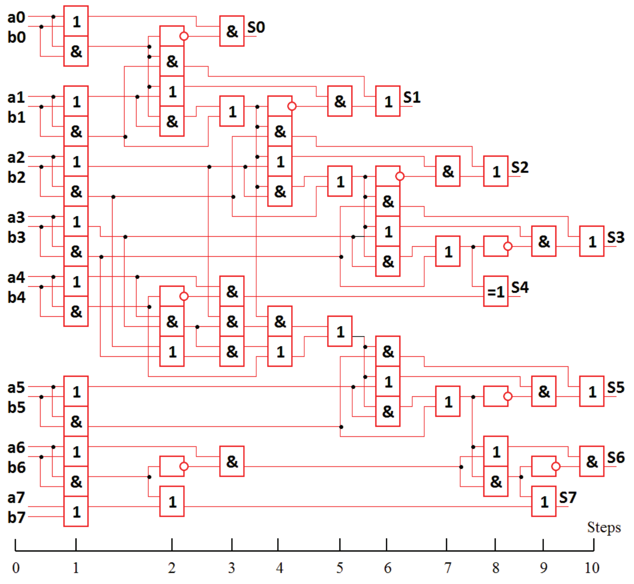


Fig. 12. Acyclic 8-bit PAA with logical elements OR in the last bit and the depth of circuit of 10 of typical 2-input elements

The prefix 8-bit Brent-Kung PPA [24] with logical elements OR in the last bit is shown in Fig. 13.

The circuit determines the depth of the adder circuit in Fig. 13 separated by a thick line and is accompanied by the numbering of logical elements along this chain. Thus, the depth of the 8-bit Brent-Kung PPA circuit (Fig. 13) is 10 typical logic elements, the complexity of the circuit is 72 discrete elements. Let's note that the XOR element has a three-discrete circuit depth and consists of four discrete logic elements, including Inventor.

The computing process of the 8-bit Brent-Kung PPA adder (Fig. 13) uses such logical operations: XOR – 13, AND – 10, OR – 10. 8-bit PAA (Fig. 12) uses: XOR – 5, AND – 20, OR – 22, Inventor – 4. The quality indicator S (for example, on energy saving) of the 8-bit PAA adder (Fig. 12) is as follows:

$$S=\frac{T_1}{T_2}=\frac{72}{66}=1.0909=9.09\ \%,$$

where T_1, T_2 – the number of discrete logic elements of 8-bit Brent-Kung PPA and 8-bit PAA, respectively.

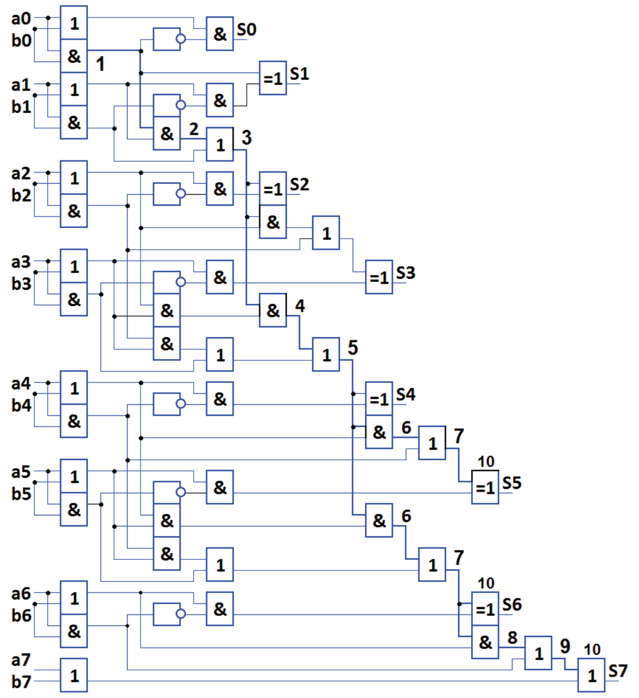


Fig. 13. Prefix 8-bit Brent-Kung PPA with logical elements OR in the last bit [24]

6.4. Comparative analysis of acyclic and prefix models for calculating adding and carry signals. The parameters of the synthesized circuits of acyclic and prefix adders are summarized in a comparative Table 5.

Considering Table 5 it is possible to see that for the selected depth of the circuit, the complexity of the circuits of acyclic adders is smaller.

Indicators of the quality of acyclic adders, for example, energy consumption are presented in Table 6.

Fig. 14 shows the dynamics of increasing the depth of the circuit for three acyclic adders (PAA) (Fig. 8, 10, 12) with an increase in the bit capacity of the circuit.

Table 7 presents a comparison of the prefix [1–4] and acyclic models for calculating adding and carry signals in the adder circuit.

Considering Table 7 it follows that the acyclic model for calculating adding and carry signals for adders' circuits of binary codes negates the prefix computational model.

Table 5

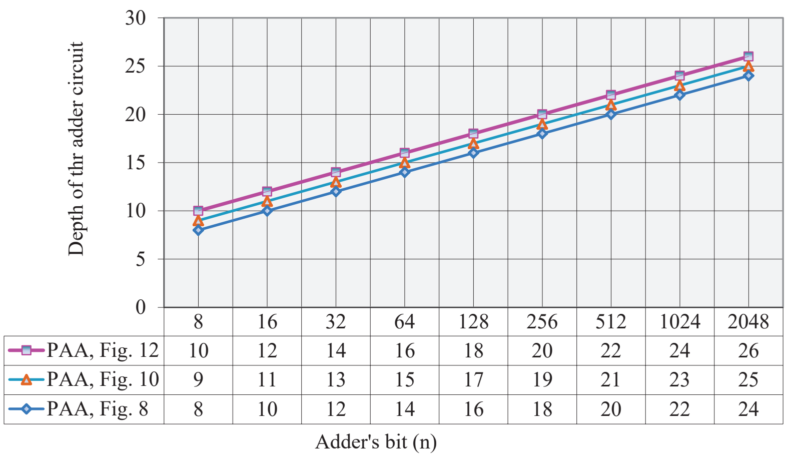
Comparison table of parameters of prefix and acyclic adders

Parallel adder of binary codes with parallel carry		Circuit depth	Circuit complexity	Adder's bit
Acyclic adder	Fig. 8	8	77	8-bit
Prefix adder	Ling Adder (Fig. 9)	8	109	8-bit
Acyclic adder	Fig. 10	9	72	8-bit
Prefix adder	Kogge-Stone (Fig. 11)	9	90	8-bit
Acyclic adder	Fig. 12	10	66	8-bit
Prefix adder	Brent-Kung (Fig. 13)	10	72	8-bit

Table 6

Comparative table of quality indicators for energy consumption of prefix and acyclic adders

Parallel adder of binary codes with parallel carry		Quality indicator of the acyclic adder
Acyclic adder	Fig. 8	41.56 %
Prefix adder	Ling Adder (Fig. 9)	
Acyclic adder	Fig. 10	25 %
Prefix adder	Kogge-Stone (Fig. 11)	
Acyclic adder	Fig. 12	9.09 %
Prefix adder	Brent-Kung (Fig. 13)	



	8	16	32	64	128	256	512	1024	2048
PAA, Fig. 12	10	12	14	16	18	20	22	24	26
PAA, Fig. 10	9	11	13	15	17	19	21	23	25
PAA, Fig. 8	8	10	12	14	16	18	20	22	24

Fig. 14. Dynamics of increasing the depth of the circuit of acyclic adders (PAA)

Table 7

Comparison table of two models for calculating adding and carry signals

Prefix model	Acyclic model
Method of prefix calculation	
Prefix model involves the process of prefix calculation, starting with the first digit of the circuit, leads, in the end, to excessive accumulation and complications of the hardware of the device	Application of acyclic models is calculated on: – logical structure of the adder circuit with a sequentially parallel method of calculating the prefix, which, in the end, reduces the complexity of the hardware of the device and does not increase the depth of the circuit; – determination of the optimal number of computational steps
Number of calculation stages	
The prefix model uses three stages to generate adding and carry signals (Fig. 3)	The acyclic model uses one stage of generating adding and carry signals (Fig. 1, 2)
The parallelism of the adder structure	
The parallel one-to-many structure of the prefix adder (Fig. 9, 11, 13) generally has fewer links than the acyclic adder	The parallel «one to many» structure of the acyclic adder (Fig. 8, 10, 12) generally has a larger number of links than the prefix adder, which certifies a high degree of parallelism of the circuit of the acyclic adder

7. SWOT analysis of research results

Strengths. To the strong side, acyclic models for calculating adding and carry signals include didactic simplifications and hardware compactness of the method, which allows to replace the three-stage prefix model by a one-stage acyclic model for calculating adding and carry signals. This will give an extension of the apparatus for synthesizing arithmetic devices for their use in digital technologies.

The connection between the number of computational steps of an oriented acyclic graph and the number of unit transfers to the highest degree causes the process of comparing the structure of the adder with the corresponding oriented acyclic graph. The purpose of this comparison is establishing the minimum sufficient number of carries for the operation of adding binary numbers in the circuit of a parallel adder with a parallel method of transfer. In the case where the synthesized adder has received a larger number of transfers than the number of computational steps of the corresponding oriented acyclic graph, then such adder will not be optimal relative to the number of computational operations.

The acyclic model is able to support aggregated structures for calculating adding and carry signals, by combining with other apparatus computational methods, in particular with the Ling transfer logic.

This is more advantageous in comparison with analogues for the following factors:

– lower cost of development and implementation, because the acyclic model defines a relatively simple structure of the adder;

– the presence of an optimization criterion – the number of computational steps of an acyclic graph indicates a minimally sufficient number of transfers of a unit to the high-order bit.

Weaknesses. The weak side of the acyclic models of calculating adding and carry signals is associated with an increase in the complexity of the synthesis of the computational structure and insufficient study of this synthesis with an increase in the device circuit capacity.

Negative internal factors inherent in acyclic models consist in increasing the time of obtaining the optimal calculation structure with the increase in the bit capacity of the adder circuit.

Opportunities. The prospect of further studies of acyclic models can be the development of a protocol for optimal alternation of the Ling transfer logic and transfer logic acyclic models with the aim of reducing the complexity of the adder circuit.

Additional possibilities that the introduction of acyclic models can bring are the study of variants of applying the transfer condition of the unit to the highest order (1). This will make it possible to obtain the optimal complexity of the computational structure of an arithmetic device.

Threats. The protocol for calculating the adding and carry signals of acyclic models does not depend on the protocols of other calculation methods, therefore there is no threat of negative impact on the object of research of external factors.

To a certain extent, the acyclic synthesis model of the adder circuit is a prefix model. At the moment, the prefix model is better because it has already created and implemented arithmetic devices with prefix structure of calculation.

8. Conclusions

1. It is revealed that the calculation of the adding and transport signal in the circuit of a parallel acyclic adder is carried out by the algorithm of logarithmic addition. The number of computational steps of an acyclic graph determines the optimal number of transfers in the parallel adder circuit with a parallel carry method.

2. The estimation of the dynamics of increasing the depth of the circuit of an acyclic adder is $O(n)$ and is linear for $n \leq 8$. With an increase in the circuit capacity from $n > 8$, the estimation of the dynamics of increasing the depth of the circuit of an acyclic adder is O (log$n$) and is logarithmic.

3. The effectiveness of acyclic models is demonstrated by examples of the synthesis of 8-bit parallel adders,

borrowed from the works of other authors for the purpose of comparison:

– Ling adder circuit (Fig. 9) [21–23] and the circuit of an acyclic 8-bit parallel adder with a depth of 8 elements circuit (Fig. 8);
– circuit of Kogge-Stone prefix adder (Fig. 11) [24] and circuit of acyclic 8-bit parallel adder with depth of 9 elements circuit (Fig. 10);
– circuit of the Brent-Kung prefix adder (Fig. 13) [24] and the circuit of an acyclic 8-bit parallel adder with a depth of 10 elements circuit (Fig. 12).

Given these examples of parallel adders, the acyclic model gives grounds for the expediency of its application in the processes of synthesis of arithmetic devices for processing digital data, since these circuits are capable of:

– increase the speed;
– reduce power consumption and heat dissipation of a digital device, integrated circuit.

References

1. Brent R. P., Kung H. T. A regular layout for parallel adders // IEEE Transactions on Computers. 1982. Vol. 31, No. 3. P. 260–264. doi: http://doi.org/10.1109/tc.1982.1675982
2. Han T., Carlson D. A. Fast area-efficient VLSI adders // IEEE 8th Symposium on Computer Arithmetic (ARITH). 1987. doi: http://doi.org/10.1109/arith.1987.6158699
3. Kogge P. M., Stone H. S. A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations // IEEE Transactions on Computers. 1973. Vol. 22, No. 8. P. 786–793. doi: http://doi.org/10.1109/tc.1973.5009159
4. Ladner R. E., Fischer M. J. Parallel Prefix Computation // Journal of the ACM. 1980. Vol. 27, No. 4. P. 831–838. doi: http://doi.org/10.1145/322217.322232
5. Solomko M., Olshansky P. The Parallel Acyclic Adder // 2017 14th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM). Lviv, 2017. P. 125–129.
6. Mathematical Modeling of Timing Attributes of Self-Timed Carry Select Adders / Balasubramanian P. et al. // Recent Advances in Circuits, Systems, Telecommunications and Control. 2013. P. 228–243. URL: http://www.wseas.us/e-library/conferences/2013/Paris/CCTC/CCTC-34.pdf
7. Venkatanaga Kumar G., Pushpalatha C. H. Implementation of Carry Tree Adders and Compare with RCA and CSLA // International Journal of Emerging Engineering Research and Technology. 2016. Vol. 4, No. 1. P. 1–11. URL: http://www.ijeert.org/pdf/v4-i1/1.pdf
8. Gedam K. S., Zode P. P. Parallel prefix han-carlson adder // International Journal of Research in Engineering and Applied Sciences. 2014. Vol. 2, No. 2. P. 81–84. URL: http://mgijournal.com/pdf_new/electronics/swapna%20gedam-1.pdf
9. Krishna Kumari V., Sri Chakrapani Y., Kamaraju M. Design and Characterization of Kogge-Stone, Sparse Kogge-Stone, Spanning tree and Brent-Kung Adders // International Journal of Scientific & Engineering Research. 2013. Vol. 4, No. 10. P. 1502–1506. URL: https://www.ijser.org/researchpaper/Design-and-Characterization-of-Koggestone-Sparse-Koggestone-Spanning-tree-and-Brentkung-Adders.pdf
10. Ramanathan P., Vanathi P. T. Hybrid Prefix Adder Architecture for Minimizing the Power Delay Product // World Academy of Science, Engineering and Technology International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering. 2009. Vol. 3, No. 4. P. 869–873. URL: https://waset.org/publications/5272/hybrid-prefix-adder-architecture-for-minimizing-the-power-delay-product
11. Kaarthik K., Vivek C. Hybrid Han Carlson Adder Architecture for Reducing Power and Delay Middle-East // Journal of Scientific Research. 2016. Vol. 24. P. 308–313. URL: https://www.idosi.org/mejsr/mejsr24(IIECS)16/48.pdf
12. Yagain D., Vijaya K. A., Baliga A. Design of High-Speed Adders for Efficient Digital Design Blocks. ISRN Electronics. 2012. Vol. 2012. P. 1–9. doi: http://doi.org/10.5402/2012/253742
13. Krishna B., Siva Durga Rao P., Prasad N. V. G. High Speed and Low Power Design of Parallel Prefix Adder // International Journal of Electronics & Communication Technology. 2012. Vol. 3, No. 4. P. 472–475. URL: http://www.iject.org/vol34/3/a572
14. Aktan M., Baran D., Oklobdzija V. G. Minimizing Energy by Achieving Optimal Sparseness in Parallel Adders // 2015 IEEE 22nd Symposium on Computer Arithmetic. 2015. P. 10–17. doi: http://doi.org/10.1109/arith.2015.13
15. Anitha R., Bagyaveereswaran V. High performance parallel prefix adders with fast carry chain logic // International Journal of Advanced Research in Engineering and Technology (IJARET). 2012. Vol. 3, No. 2. URL: https://www.slideshare.net/iaemedu/high-performance-parallel-prefix-adders-with-fast-carry-chain-logic
16. Kombinatsiinyi sumator: Patent 115751 UA, MPK G 06 F 7/501 (2006.01) / Vozna N. Ya. et al. Appl. No. a201701347; Filed: 13.02.2017; Published: 11.12.2017; Bul. No. 23. URL: http://uapatents.com/6-115751-kombinacijjnijj-sumator.html
17. Gurkayna F. K. et al. Higher radix Kogge-Stone parallel prefix adder architectures // 2000 IEEE International Symposium on Circuits and Systems. Emerging Technologies for the 21st Century. Proceedings (IEEE Cat No. 00CH36353). Presses Polytech. Univ. Romandes, 2000. doi: http://doi.org/10.1109/iscas.2000.857516
18. Knowles S. A family of adders // Proceedings 14th IEEE Symposium on Computer Arithmetic (Cat. No. 99CB36336). IEEE Comput. Soc., 1999. doi: http://doi.org/10.1109/arith.1999.762825
19. Beaumont-Smith A., Lim C.-C. Parallel prefix adder design // Proceedings 15th IEEE Symposium on Computer Arithmetic. ARITH-15 2001. IEEE Comput. Soc., 2001. doi: http://doi.org/10.1109/arith.2001.930122
20. Zimmermann R. Efficient VLSI implementation of modulo (2/sup *n*/±1) addition and multiplication // Proceedings 14th IEEE Symposium on Computer Arithmetic (Cat. No. 99CB36336). IEEE Comput. Soc., 1999. doi: http://doi.org/10.1109/arith.1999.762841
21. Zeydel B. R., Baran D., Oklobdzija V. G. Energy-Efficient Design Methodologies: High-Performance VLSI Adders // IEEE Journal of Solid-State Circuits. 2010. Vol. 45, No. 6. P. 1220–1233. doi: http://doi.org/10.1109/jssc.2010.2048730
22. Govindarajulu S., Vijaya Durga Royal T. Design of Energy-Efficient and High-Performance VLSI Adders // International Journal of Engineering Research. 2014. Vol. 3, No. 2, P. 55–59. URL: http://ijer.irponline.in/ijer/publication/v3si2/IJER_2014_NCSC%2013.pdf
23. Pinto R., Shama K. Efficient shift-add multiplier design using parallel prefix adder // International Journal of Control Theory and Applications. 2016. Vol. 9, No. 39. P. 45–53. URL: http://serialsjournals.com/serialjournalmanager/pdf/1500377875.pdf
24. Solomko M., Krulikovskyi B. Study of carry optimization while adding binary numbers in the rademacher number-theoretic basis // Eastern-European Journal of Enterprise Technologies. 2016. Vol. 3, No. 4 (81). P. 56–63. doi: http://doi.org/10.15587/1729-4061.2016.70355

Solomko Mykhailo, *PhD, Associate Professor, Department of Computer Engineering, National University of Water and Environmental Engineering, Rivne, Ukraine, e-mail: doctrinas@ukr.net, ORCID: http://orcid.org/0000-0003-0168-5657*