Проведеними дослідженнями встановлена перспектива збільшення продуктивності обчислювальних компонентів, зокрема комбінаційних 16-bit суматорів, на основі використання принципів обчислення цифрових сигналів ациклічної моделі.

Застосування ациклічної моделі для синтезу 16-bit паралельних суматорів розраховано на:

– процес послідовного (для молодших розрядів схеми суматора) та паралельного (для решти розрядів) обчислення сигналів суми і перенесення. Завдяки зазначеному підходу стає можливим, у підсумку, зменшити складність апаратної частини пристрою та не збільшити глибину схеми;

– фіксацію (планування) глибини схеми суматора перед його синтезом. Це дозволяє використовувати логічну структуру транзитивного перенесення, що забезпечує оптимальну глибину схеми суматора та не збільшує її складність.

Використання ациклічної моделі для побудови 16-bit паралельних суматорів вигідніше у порівнянні з аналогами за такими чинниками:

– меншою вартістю розробки, оскільки ациклічна модель визначає простішу структуру 16-bit суматора;

– застосуванням останніх розроблених логічних структур транзитивного перенесення, що дозволяє зменшити затримку сигналів суми та перенесення, площу, потужність та підвищити загальну продуктивність 16-bit суматорів бінарних кодів.

Завдяки цьому забезпечується можливість отримання оптимальних значень показників складності структури та логічної глибини схеми цифрової компоненти. У порівнянні з аналогами це забезпечує збільшення показника якості 16-bit ациклічних суматорів, наприклад, за енергоспоживанням, площею чипа, у залежності від обраної структури, на 15–27 %, а за швидкодією на 10–60 %.

Є підстави стверджувати про можливість збільшення продуктивності обчислювальних компонентів, зокрема 16-bit суматорів бінарних кодів, шляхом використання принципів обчислення цифрових сигналів ациклічної моделі

Ключові слова: оптимальна швидкодія ациклічних суматорів, Ling Adder, Kogge-Stone Adder, Knowles Adder

UDC 681.325
DOI: 10.15587/1729-4061.2019.168485

OPTIMAL PERFORMANCE OF 16-BIT ACYCLIC ADDERS OF BINARY CODES

M. Solomko
PhD, Associate Professor*
E-mail: doctrinas@ukr.net
P. Tadeyev
PhD, Doctor of Pedagogical Sciences, Professor
Department of Higher Mathematics**
E-mail: ptadeyev@gmail.com
V. Nazaruk
PhD*
E-mail: vitalij.nazaruk@gmail.com
N. Khariv
Senior Lecturer
Department of Applied Mathematics**
E-mail: hnata@ukr.net
*Department of Computer Engineering**
**National University of Water and Environmental Engineering
Soborna str., 11, Rivne, Ukraine, 33028

1. Introduction

Computer industry creates more and more productive computing components using integrated circuits (IC). Better production of chips is achieved through the development of new computing architecture with the efficient use of technological improvements. However, the improvement of IC parameters, including the performance of its operation, power consumption and temperature mode continue to be a relevant task for designing and technology of manufacturing integrated circuits. The performance and accuracy of a processor or an information system depends on the efficiency of the adder. Binary addition is a major arithmetic operation in the systems of super-large integrated circuits (SLIC). Binary adders are among the most important elements in processor chips, ALU, counters, methods of memory addressing, as a part of the filter, for example, the filter of DSP-grid, etc. For this reason, the addition operation is the most commonly used operation in digital circuits. As the adder takes a critical position inside the ALU of microprocessors, it remains relevant to ensure that its performance should be adequate to meet assigned specifications of performance, area and power consumption of different topologies of adders.

This paper deals with the architecture of the 16-bit parallel acyclic adder (PAA) [1, 2]. In addition, it presents the latest designed logical structures of transitive carry, which make it possible to reduce the delay of sum and carry signals, area, power and improve the overall efficiency of digital components.

The processor evolution is a result of continuous optimization, so it remains relevant to study 16-bit adders of binary codes, which are aimed, specifically, at the improvement of such factors, as:

– manufacturing technology;
– structural implementation;
– performance and power consumption.

2. Literature review and problem statement

The use of parallel-prefix adders in the development of SLIC was considered in [3]. The optimization of the logical structure of a 16-bit prefix adder Ladner-Fischer was presented. The proposed system consists of three operation stages – the pre-processing stage, the generation stage and the post-processing stage. The pre-processing stage focuses

© M. Solomko, P. Tadeyev, V. Nazaruk, N. Khariv, 2019
Received date: 22.05.2019 Accepted date: 10.06.2019 Published date: 29.06.2019

on expansion and generation, the generation stage focuses on the performance of generation, and the post-processing stage focuses on the end result. Computation performance of logical structures of Ripple Carry Adder and Ladner-Fischer Adder was compared.

The logical structure, which reduces power consumption of SLIC, was proposed in [4]. The presented model of delays of the computing track of the built-in SLIC system was presented. The prefix 16-bit adder with reversible logic elements was developed using the PERES logic. The structure of the adder has the minimal logical depth and complexity of the circuit. The results of modeling revealed that the net delay of the computing track for the 16 X16-bit prefix using reversible logic is 20.828 ns, for Kogge Adder Stone, reading is up to 17.247 ns.

A high-speed fault-tolerant parallel prefix adder was proposed in paper [5]. Because the logical structure of Kogge-Stone has inherent redundancy in the logical structure of carry, a fault-tolerant parallel prefix adder can be implemented. The Kogge-Stone structure can perform only correction of faults, but does not detect them. Therefore, to achieve this goal, it is proposed to use Sparse Kogge-Stone. The method uses the Sparse Kogge-Stone adder, which is able both to detect and correct problems. The synthesis and simulation of fault-resistant structures for the FPGA platform were performed.

Development and implementation of a hybrid parallel prefix adder 16-bit Ling Adder were presented in paper [6]. The topology of a hybrid adder uses the Ladner-Fischer approach for even indices and the Kogge-Stone for odd indices. An independent computation of carries for odd and even bits directly enables reducing the branching of the logical structure of a prefix and thus reduces the signal delay. The area effectiveness is achieved by calculating the real carry using the modified Ling equations. The proposed adders are implemented with the 16-bit and 32-bit size of a word based on the modified Ling equations using the technology of CMOS of 0.18 microns. The synthesis results demonstrate that the proposed adders can reach up to 24 % and 35 % of power saving and the time of digital device delay, respectively, compared with the adders synthesized based on the conventional Ling equations.

Optimization of the parameters of 16-bit prefix Kogge-Stone Adder and Ladner Fischer Adder during designing with the help of Verilog is considered in article [7]. The code was implemented in Xilinx Spartan 3E100CP132. It was noted that the changed structure of the parallel prefix demonstrates the best efficiency indicators as compared with the traditional prefix adders and can be widely used in the industries to achieve the desired computation efficiency.

Designing and simulation of the prefix Brent Kung Adder using CMOS logic and 45 nm technology are explored in [8]. The results of designing with the known structures of Ripple Carry Adder and Carry Look Adder were compared. The obtained results show that energy consumption and delay in propagation of the sum and carry signals for Brent Kung Adder are reduced compared with the Ripple Carry Adder or the Carry Look Adder.

Comparison of the parameters of delay, power consumption and area for logical structures Ripple Carry Adder (RCA), Carry Look Adder (CLA), the Manchester Carry Chain (MCC) and the Kogge-Stone Adder (KSA) was performed in [9]. Modeling of these structures was carried out using the 180 nm technology. It was noted that the KSA architecture is the best when it comes to computation efficiency. It was established that the RCA occupies the smallest area of chips – 1,118 nm^2.

The structure of the parallel prefix is a typical structure of the adder of binary codes, which emphasizes concurrency in transmission of carry signals [10]. This structure ensures a compromise between complexity and the logical depth of the adder circuit. In paper [10], the structural-decomposition and procedure design of possible structures of the parallel prefix was proposed. «Join», «paste» and «alternate» are introduced as the main operations for the construction of a possible parallel prefix diagram. In this work it is shown that all of the well-known structures, specifically, the Sklansky prefix-diagram, Kogge-Stone prefix-diagram, Han-Carlson prefix-diagram, Brent-Kung prefix-diagram can be successfully represented using this method. The proposed approach extends the apparatus for the synthesis of parallel prefix structures that can be used to optimize the design of digital components.

The adder with accelerated carry was presented with a patent [11]. The adder includes: an input $2n$-bit bus; n/m m-bit adders; m-bit incremental adder; m+1-bit multiplexer with paraphrase control inputs; an output n+1-bit bus. Hardware complexity of the proposed adder with accelerated carry in relation to the nearest analogue is reduced twice, while performance increases by 1.5 times.

The explored literary sources [3–10] prove that the source objects to increase the efficiency of signals' processing in digital components are the models for computation of parallel prefix, specifically, the architecture of Ling Adder, Kogge-Stone, Ladner-Fischer, Brent Kung, Sklansky and Han-Carlson.

Among the well-known prefix structures, the major ones include the Ling and Kogge-Stone parallel prefix adder with the structure of prefix carry. They are the end case of a large list of circuits of adding binary codes, each of which is unique for its property of minimum logical capacity.

These adders provide a theoretical base, forming the taxonomy of the prefix adders (Fig. 1) [12]. However, the use of such architectures is justified only by reaching compromises in terms of delay, area and capacity in order to display a wide range of services in the design. When trying to go beyond these limits in order to increase the performance of processing digital signals, there arise objective difficulties associated with high complexity of the circuits and a huge number of connecting wires (tracks) (for example, the Kogge-Stone architecture).

Taxonomy always forms an encyclopedic list of some objects. In terms of practical application, not all the objects of an encyclopedia will be used. As the adder takes a critical position inside the ALU of microprocessors, it remains relevant to ensure that its efficiency should be adequate to meet the specifications of performance, area and power consumption of the digital component, and preferably without compromises.

A tool to ensure efficient operation of an adder and a digital component without compromises, to some extent, is the protocol of dynamics of increasing the depth of the circuit of an acyclic adder of binary codes with an increase in bit size of its circuit (Fig. 2).

The dynamics of increasing the depth of the PAA circuit is determined by the logarithmic dependence – doubling the number of bits of n of an adder increases the circuit depth by the constant magnitude – by two logical elements.

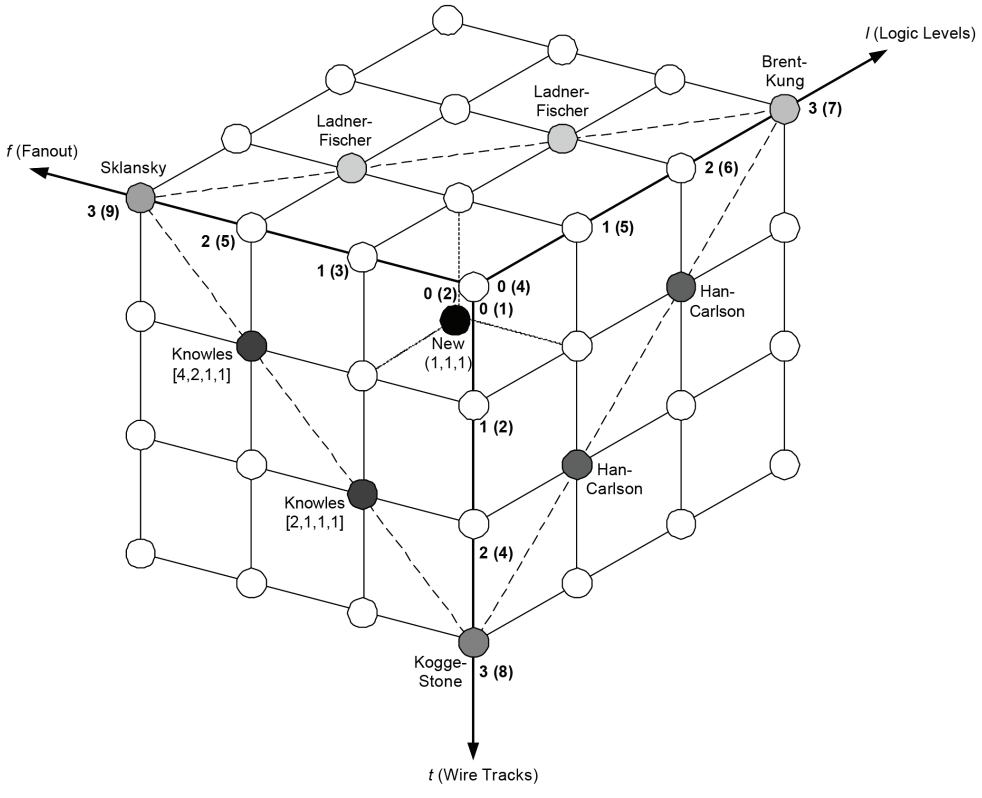

Fig. 1. Taxonomy of prefix adders: Logicallevels – $L=\log 2n+1$, Max fanouts – $F=2^f+1$, Wire tracks – $T=2^t$ [12]

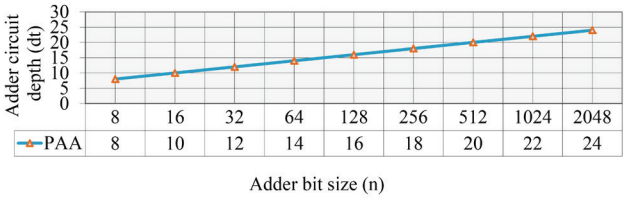

Fig. 2. Dynamics of increasing the depth of the circuit of acyclic adder (PAA), based on 2-input logical elements

An acyclic model of the adder [1, 2] was designed for a logical structure with the sequential-parallel way of computation of a digital signal. The sequential method of carry is fundamental in relation to minimum consumption of the hardware of digital components. Thus, the prefix and acyclic models are different objects – they have different beginnings (principles) of computation, and therefore have different capabilities with respect to performance, chip area and power saving.

That is why there are certain reasons to believe that the theoretical base, which is represented by the prefix architectures, including Ling Adder, Kogge-Stone Adder, Knowles Adder and generalized by the taxonomy of prefix adders (Fig. 1) is insufficient to carry out the optimization of delay, area and power without compromises. This causes the need for research into an acyclic model of digital signals' processing, including the protocol of dynamics of increasing the depth of the circuit of an acyclic adder with an increase in the magnitude of its bit size (Fig. 2).

3. The aim and objectives of the study

The aim of the research is the synthesis of the optimal structure of 16-bit parallel adders of binary codes with logical XOR elements in the last bit by using an acyclic model of signal processing. This will make it possible to increase performance, reduce energy consumption of 16-bit adders, in comparison with the analogues, and to spread the principle of synthesis on larger bit size of acyclic adders with the sequential-parallel way of carry.

To achieve the aim, the following tasks were set:

– to synthesize the optimum logical structure of the sequential-parallel transitive carry of unity to higher bits in the circuit of the acyclic 16-bit adder of binary codes;

– to establish the dynamics of increasing the depth of the circuit of the acyclic 16-bit Adder (PAA) based on 2-input logical elements, compared to the 8-bit acyclic adder;

– to perform a comparable analysis of performance and complexity of the structures of the 16-bit acyclic adder with logical XOR elements in the last bit, and 16-bit adders of the prefix model of calculation of the sum and carry signals. In particular, to carry out the analysis of the dependence of the circuit simplicity according to the logical depth on adder's circuit.

4. The logic of transitive carry

Operation of binary addition in the position system uses such types of carry of unity to the higher bits: «kill», «generate», «propagate» or transitive carry:

if $a_i=b_i=0$, then $c_i=0$ (the «kill» carry),

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

However, if one of bits a_i or b_i is equal to 1, and another is 0, then c_{i-1} has significant content for carry, that is,

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

Each bit, therefore, corresponds to one of the three types of carry statuses: k (kill), g (generate) or p (propagate). This status is known, first of all, as it allows decreasing the time to perform the addition operation.

Carry statuses for the 1-bit (full) adder (Fig. 3) are given in Table 1.

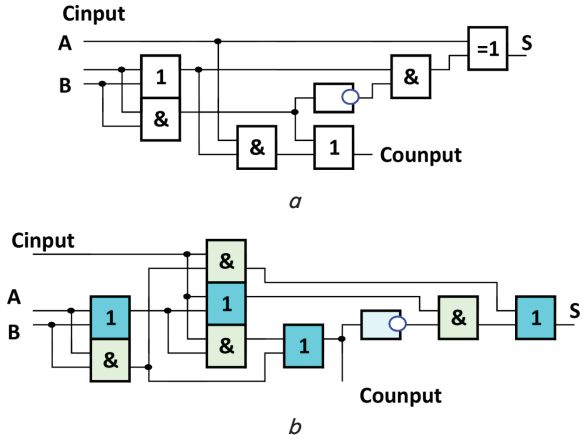

Fig. 3. Full adder of binary codes:
a – for logical element 2-In XOR (complexity of 10 elements);
b – on complicated logical element MUL «2 in 1»
(complexity of 9 elements)

Table 1

Carry status of the 1-bite adder of binary codes

A	B	C_{input}	C_{ounput}	S	Carry status
0	0	0	0	0	kill
0	0	1	0	1	kill
0	1	0	0	1	propagation
0	1	1	1	0	propagation
1	0	0	0	1	propagation
1	0	1	1	0	propagation
1	1	0	1	0	generation
1	1	1	1	1	generation

Logical equations of the 1-bit adder in Fig. 2 are the following:

$$G = AB;$$

$$P = A \oplus B;$$

$$K = \overline{A}\,\overline{B};$$

$$S = A \oplus B \oplus C_{input} = P \oplus C_{input};$$

$$C_{ounput} = AB + AC_{input} + BC_{input} = G + PC_{input}.$$

The main carry is the «propagation» status, on which the performance and complexity of the device circuit depends. The logical equation that determines the carry of «propagation» status (transitive carry) in most cases is as follows:

$$p_i = (a_i + b_i)c_{in},$$

where the sign «+» means the logical OR operation. It is possible to determine the «propagation» type using the logical XOR operation:

$$p_i = (a_i \oplus b_i)c_{in}.$$

The logical structure of the adder that repeats the arithmetic result should take into consideration not only the carry of «propagation» status, but also to ensure and implement the condition of propagation of the signals of carrying the unity to higher bits. The logical equations of the carry condition are the following:

$$p_i = a_i \vee b_i \;\text{ or }\; p_i = a_i + b_i. \qquad (1)$$

If p_i=1, transitive carry to the next bits will be possible, in the case p_i=0 transitive carry to the next bits is impossible.

We will demonstrate the condition of carrying unity to the higher bit using the example of operation of addition of 1-bit numbers to the column (Table 2).

Table 2

Addition of 1-bit numbers to the column

Possible variants of addition				
Unity from lower bit (C_{input})	1	1	1	1
Number A	0	0	1	1
Number B	0	1	0	1
Sum	01	10	10	11

Considering the variants of the addition of 1-bit binary numbers shown in Table 2, we see that if $A \vee B$=1, the unity from the lower bit C_{input} is carried to higher (second) bit of the sum (C_{ounput}=1). If $A \vee B$=0, the sum remains 1-bit, the unity from the lower bit C_{input} is not carried to the higher (second) bit of the sum (C_{ounput}=0). A similar logic of carrying the unity to the higher bit is also retained when adding n-bit binary numbers.

Example. Conduct arithmetic addition of binary codes: A=0110101100 and B=0010010100 (Fig. 4).

Carry	1	1		1	1	1	1			
Code A	0	1	1	0	1	0	1	1	0	0
	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕	⊕
Code B	0	0	1	0	0	1	0	1	0	0
Sum S	1	0	0	1	0	0	0	0	0	0

Fig. 4. Addition of binary codes

When adding binary codes, the carry that appeared in the bit with index i=3g_0=a_0b_0=1∧1=1 carries to the bit with index i=6, the carry that appeared in the bit with index i=7: g_5=a_5b_5=1∧1=1 carries to the bit with index i=9.

The procedure of arithmetic addition is, in fact, a description of the operation of binary addition. In turn, the circuit of the adder, which implements binary addition is the method, therefore, the condition of carry of the «propagation» status (1) must be represented with the corresponding logical structure (Fig. 5–9).

Logical equations of the 3-bit adder in Fig. 6 are the following:

$$S_0 = a_0\overline{b_0} + \overline{a_0}b_0;$$

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

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

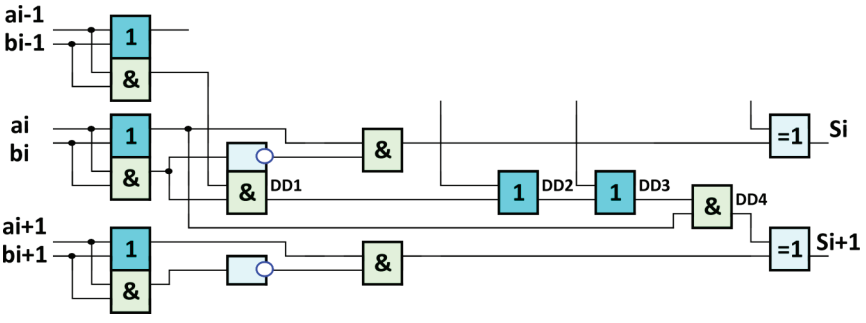

Fig. 7. Sequential structure of transitive carry on elements DD1 DD2 DD3 DD4

Logical structure «OR-AND» for a number of cases makes it possible to organize the carry using fewer logical elements in comparison with «AND-OR», which in this way reduces the complexity of the digital device.

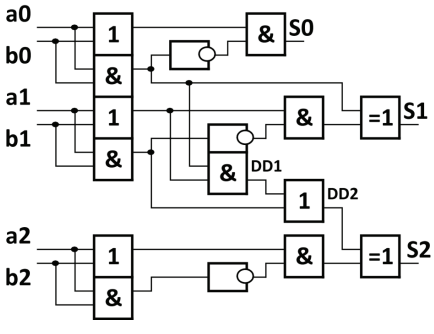

Fig. 5. The structure of transitive carry «AND-OR» on elements DD1, DD2

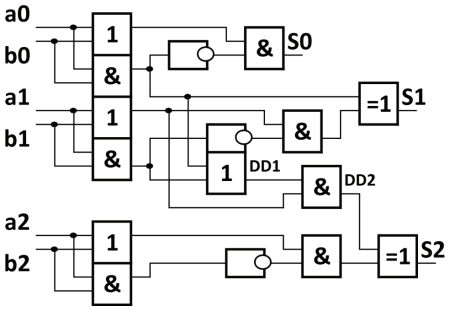

Fig. 6. The structure of transitive carry «OR-AND» on elements DD1, DD2

Logical elements, DD1 DD2 DD3 DD4 in Fig. 7 demonstrate a sequential structure of the carry of unity to the higher bit. A sequential structure, compared with the parallel one, requires a smaller number of logical elements, which in the end reduces the device circuit complexity.

The structure of carry with logical XOR and MUL elements, which make up the Ling logic (Fig. 8), ensures the optimal logical depth of the adder circuit for neighboring bits, beginning with the 8-bit circuit of the device.

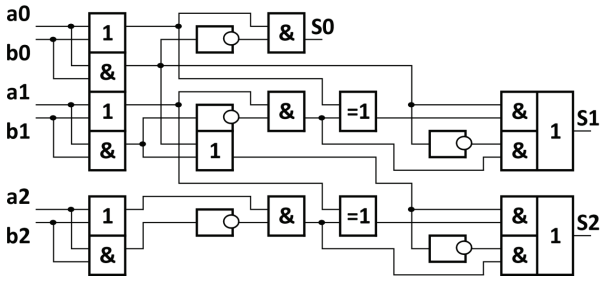

Fig. 8. Optimal transitive carry on the logical Ling structure for neighboring bits of adder

Logical equation of the 3-bit adder in Fig. 8 are the following:

$$S_0 = a_0 \overline{b_0} + \overline{a_0} b_0;$$

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

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

Optimization of the depth of the adder circuit not only within neighboring bits of a digital device, but also for some interval of bits, is provided by the structure with two XOR elements and one MUL elements (Fig. 9).

The application of the logical structure with two XOR elements and one MUL element to ensure optimal transitive carry for the interval of bits of the adder circuit is demonstrated by the circuit of an acyclic 16-bit PAA (p. 5).

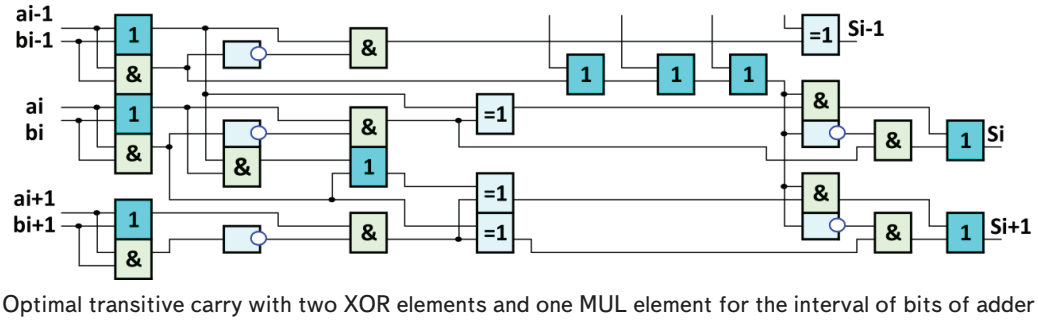

Fig. 9. Optimal transitive carry with two XOR elements and one MUL element for the interval of bits of adder circuit

5. Results of application of acyclic model to reduce complexity and increase performance of 16-bit adders of binary codes

To ensure identical conditions for comparing 16-bit adders of binary codes, we will represent the circuits of acyclic (PAA) and prefix (PPA) with logical XOR elements in the last bit.

Fig. 10 shows the acyclic 16-bit PAA with logical XOR elements in the last bit and the depth of the circuit of 10 typical 2-input elements. Given the fact that XOR is composed of four elements [2], the complexity of the circuit in Fig. 10 is 221 2-input elements.

The first five logical equations of the 16-bit acyclic adder in Fig. 10 are the following:

$$S_0 = a_0\overline{b_0} + \overline{a_0}b_0;$$

$$S_1 = a_0b_0\overline{a_1}\,\overline{b_1} + \overline{a_0}a_1\overline{b_1} + \overline{a_0}\,\overline{a_1}b_1 + a_0b_0a_1b_1 + \overline{b_0}a_1\overline{b_1} + \overline{b_0}\,\overline{a_1}b_1;$$

$$S_2 = \overline{a_0}\,\overline{b_1}a_2\overline{b_2} + \overline{a_0}\,\overline{a_1}a_2\overline{b_2} + a_0\overline{b_1}\,\overline{a_2}b_2 + a_0b_0b_1\overline{a_2}\,\overline{b_2} + a_0b_0a_1\overline{a_2}\,\overline{b_2} + \overline{b_0}\,\overline{b_1}a_2b_2 + \overline{b_0}\,\overline{a_1}a_2\overline{b_2} + \overline{b_0}\,\overline{b_1}\,\overline{a_2}b_2 + \overline{b_0}\,\overline{a_1}\,\overline{a_2}b + a_0b_0b_1a_2b_2 + a_0b_0a_1a_2b_2 + \overline{a_1}\,\overline{b_1}a_2\overline{b_2} + \overline{a_1}\,\overline{b_1}\,\overline{a_2}b_2 + a_1b_1\overline{a_2}\,\overline{b_2} + a_1b_1a_2b_2;$$

$$S_3 = \overline{a_0}\,\overline{b_1}\,\overline{b_2}a_3\overline{b_3} + \overline{a_0}\,\overline{a_1}\,\overline{b_2}a_3\overline{b_3} + \overline{a_0}\,\overline{b_1}\,\overline{a_2}a_3\overline{b_3} + \overline{a_0}\,\overline{a_1}\,\overline{a_2}a_3\overline{b_3} + \overline{a_0}\,\overline{b_1}\,\overline{b_2}\,\overline{a_3}b_3 + \overline{a_0}\,\overline{a_1}\,\overline{b_2}\,\overline{a_3}b_3 + a_0\overline{b_1}\,\overline{a_2}\,\overline{a_3}b_3 + a_0\overline{a_1}\,\overline{a_2}\,\overline{a_3}b_3 + \overline{b_0}\,\overline{b_1}\,\overline{b_2}a_3\overline{b_3} + \overline{b_0}\,\overline{a_1}\,\overline{b_2}a_3\overline{b_3} + \overline{b_0}\,\overline{b_1}\,\overline{a_2}a_3\overline{b_3} + \overline{b_0}\,\overline{a_1}\,\overline{a_2}a_3\overline{b_3} + \overline{b_0}\,\overline{b_1}\,\overline{b_2}\,\overline{a_3}b_3 + \overline{b_0}\,\overline{a_1}\,\overline{b_2}\,\overline{a_3}b_3 + \overline{b_0}\,\overline{b_1}\,\overline{a_2}\,\overline{a_3}b_3 + \overline{b_0}\,\overline{a_1}\,\overline{a_2}\,\overline{a_3}b_3 + \overline{a_1}\,\overline{b_1}\,\overline{b_2}a_3\overline{b_3} + \overline{a_1}\,\overline{b_1}\,\overline{a_2}a_3\overline{b_3} + \overline{a_1}\,\overline{b_1}\,\overline{b_2}\,\overline{a_3}b_3 + \overline{a_1}\,\overline{b_1}\,\overline{a_2}\,\overline{a_3}b_3 + a_0b_0b_1b_2\overline{a_3}\,\overline{b_3} + a_0b_0a_1b_2\overline{a_3}\,\overline{b_3} + a_0b_0b_1a_2\overline{a_3}\,\overline{b_3} + a_0b_0a_1a_2\overline{a_3}\,\overline{b_3} + a_0b_0b_1b_2a_3b_3 + a_0b_0a_1b_2a_3b_3 + a_0b_0b_1a_2a_3b_3 + a_0b_0a_1a_2a_3b_3 + a_1b_1b_2\overline{a_3}\,\overline{b_3} + a_1b_1a_2\overline{a_3}\,\overline{b_3} + a_1b_1b_2a_3b_3 + a_1b_1a_2a_3b_3 + \overline{a_2}\,\overline{b_2}a_3\overline{b_3} + \overline{a_2}\,\overline{b_2}\,\overline{a_3}b_3 + a_2b_2\overline{a_3}\,\overline{b_3} + a_2b_2a_3b_3.$$

The dynamics of an increase in the depth of the circuit of acyclic 16-bit adder based on 2-input logical elements, compared to the 8-bit acyclic adder [2], is two logical elements. This corresponds to the protocol of dynamics of an increase in the depth of the acyclic adder circuit based on 2-input logical elements and shown in Fig. 2.

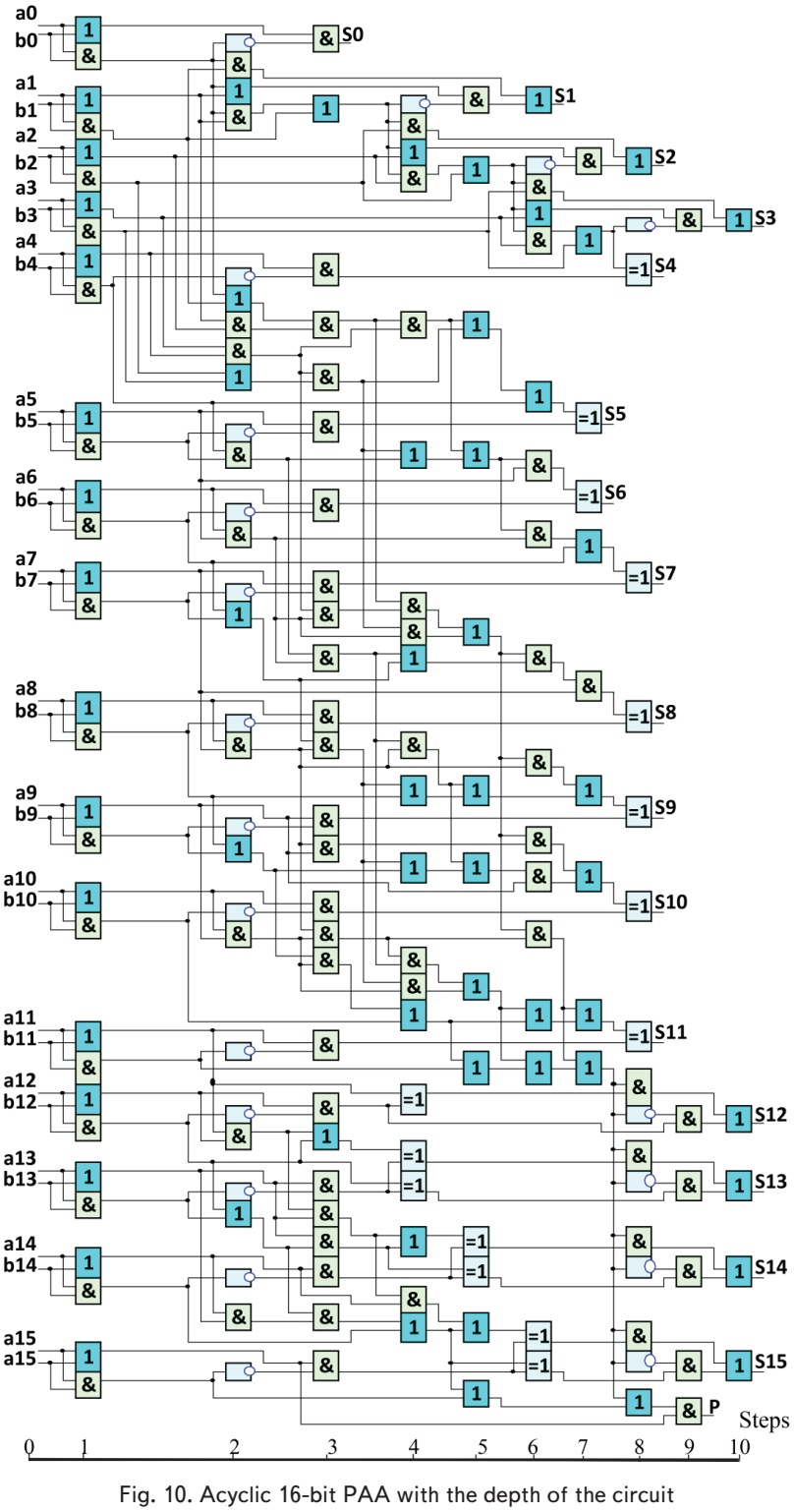

Fig. 10. Acyclic 16-bit PAA with the depth of the circuit of 10 typical 2-input elements

6. Comparative analysis of 16-bit acyclic and the prefix adders of binary codes

The prefix of 16-bit Ling Adder [13–15] with logical XOR elements in the last bit and the depth of the circuit of 11 typical 2-input logic elements with improvement of the logical structure of the adder, which reduces the circuit complexity, is shown in Fig. 11. Given that XOR is composed of

four elements [2], the complexity of the circuit in Fig. 11 is 281 of 2-input elements.

Computation process of the 16-bit Ling Adder PPA (Fig. 11) uses the following logical operations: XOR – 15, AND – 115, OR – 75, Inventor – 31. The 16-bit adder PAA (Fig. 10) uses: XOR – 15, AND – 80, OR – 61, Inventor – 20. Given the fact that the logic of the XOR element uses four logical elements, including Inventor, it is possible to estimate the quality indicator S (for example, in terms of energy saving) of operation of the 16-bit adder PAA (Fig. 10), compared with the adder in Fig. 11:

$$S = \frac{T_1}{T_2} = \frac{281}{221} = 1.2715 = 27.15\ \%,$$

where T_1, T_2 are the number of 2-input logical elements of the 16-bit Ling Adder PPA (Fig. 11) and 16-bit PAA (Fig. 10), respectively.

Quality indicator V in terms of computation performance of PAA (Fig. 10), compared to the 16-bit Ling Adder PPA (Fig. 11) makes up:

$$V = \frac{N_1}{N_2} = \frac{11}{10} = 1.1 = 10\ \%,$$

where N_1, N_2 are the depth of the circuit of 16-bit Ling Adder PPA (Fig. 11) and 16-bit PAA (Fig. 10), respectively.

The prefix 16-bit Kogge-Stone PPA [16, 17] with logical XOR elements in the last bit is shown in Fig. 12. Taking into consideration the depth of three XOR elements, complexity of four elements [2], the depth of the 16-bit Kogge-Stone PPA (Fig. 12) will be 11 typical 2-input logic elements, the circuit complexity is 256 elements. One of the variants of the depth of the circuit of the 16-bit Kogge-Stone PPA in Fig. 12 is highlighted by bold line, along which the numbering of logical elements is accompanied by the figures, highlighted in red.

The computation process of the 16-bit Kogge-Stone PPA (Fig. 12) uses such logical operations: XOR – 15, AND – 115, OR – 65, Inventor – 16. The 16-bit adder PAA (Fig. 10) uses: XOR – 15, AND – 80, OR – 61, Inventor – 20. Considering that the logic of the XOR element uses four logical elements, the quality indicator S (for example, in terms of power saving) of operation of the 16-bit adder PAA (Fig. 10), compared with the adder in Fig. 12, is as follows:

$$S = \frac{T_1}{T_2} = \frac{256}{221} = 1.1584 = 15.84\ \%,$$

where T_1, T_2 are the number of 2-input logical elements of the 16-bit Kogge-Stone PPA (Fig. 12) and 16-bit PAA (Fig. 10), respectively.

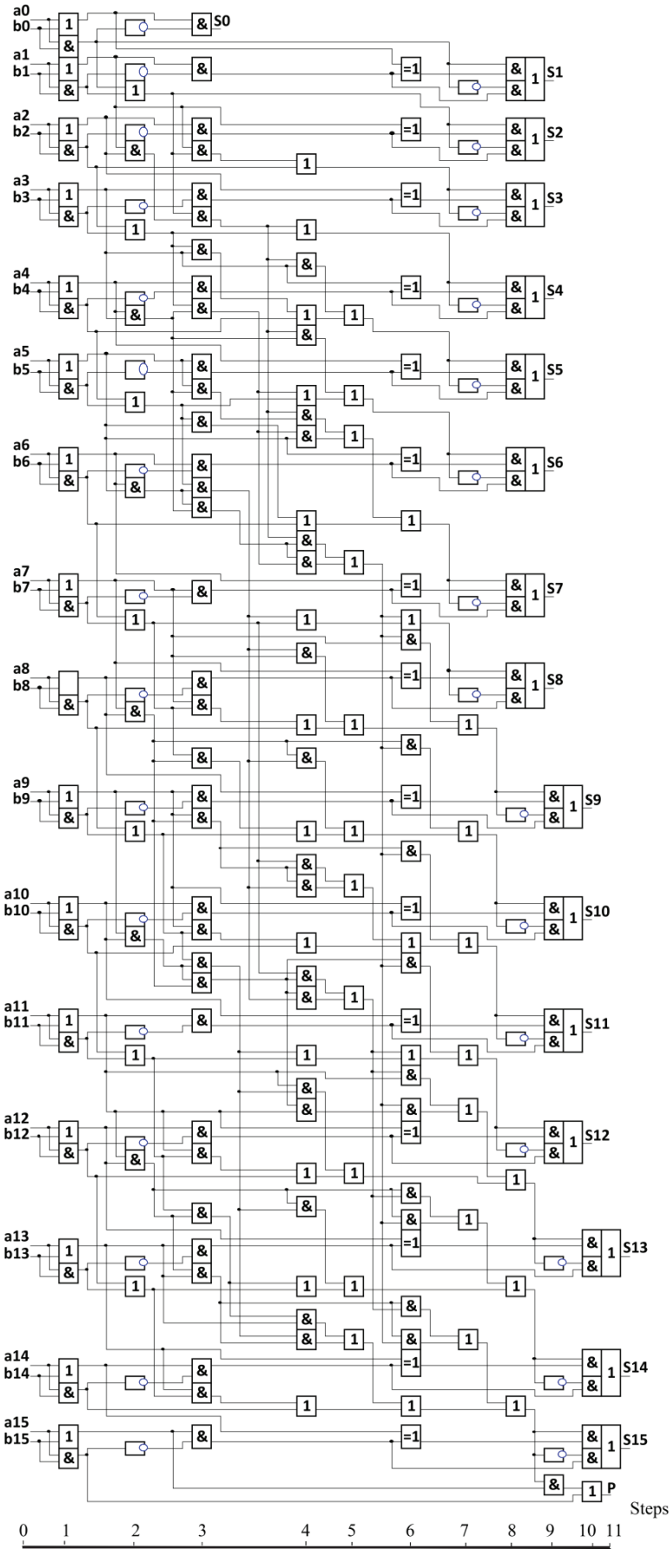

Fig. 11. Prefix 16-bit Ling Adder PPA with depth of circuit of 11 typical 2-input elements [13–15]

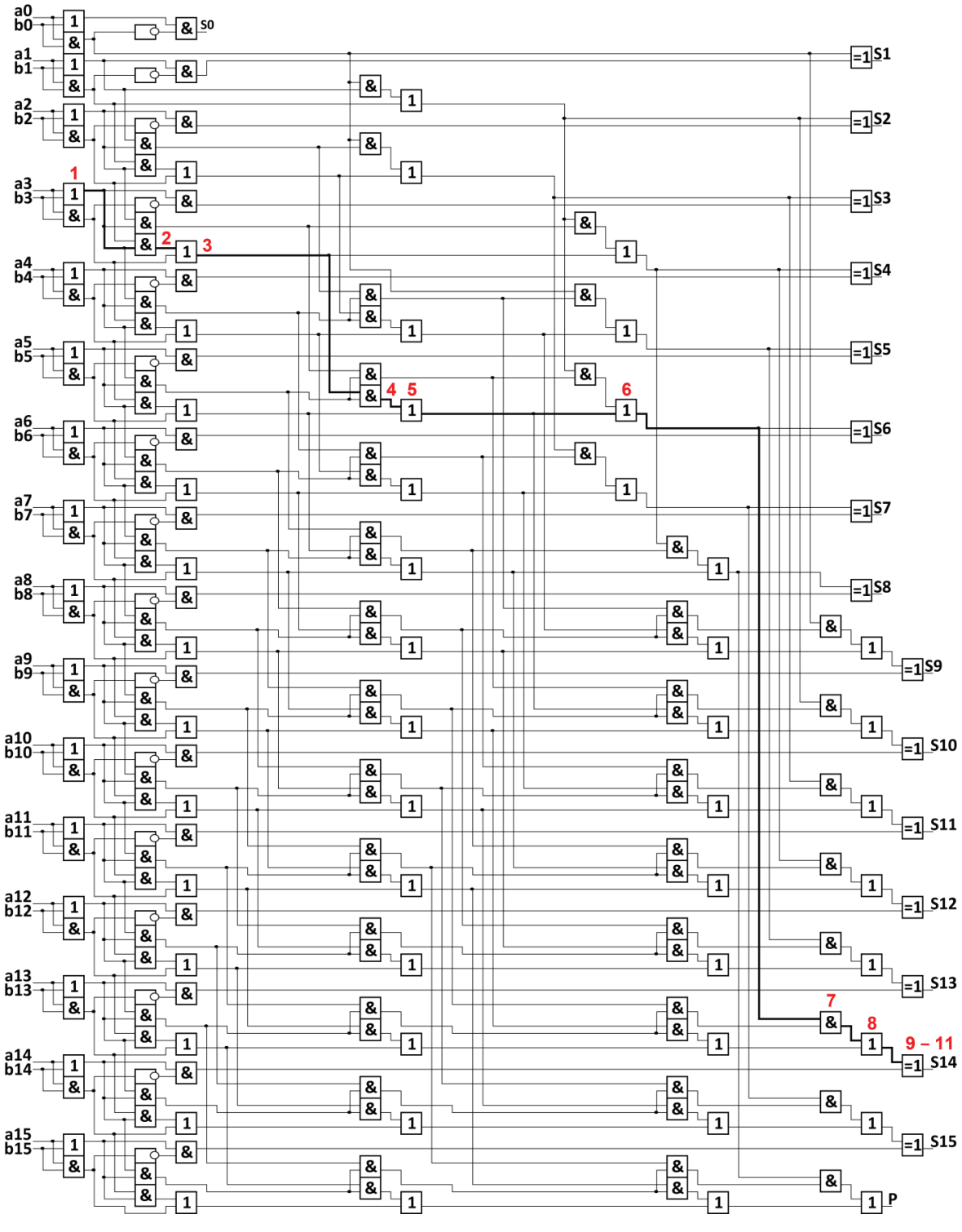

Fig. 12. Prefix 16-bit Kogge-Stone PPA with the depth of circuit of 11 typical 2-input elements [16, 17]

Quality indicator V in terms of computation performance of PAA (Fig. 10), compared to the 16-bit Kogge-Stone Adder PPA (Fig. 12) is:

$$V = \frac{N_1}{N_2} = \frac{11}{10} = 1.1 = 10\,\%,$$

where N_1, N_2 are the depth of the circuit of the 16-bit Kogge-Stone Adder PPA (Fig. 12) and of 16-bit PAA (Fig. 10), respectively.

The prefix 16-bit Knowles PPA [18, 19] with logical XOR elements in the last bit is presented in Fig. 13. Taking into consideration that the depth of the XOR is three elements, complexity is four elements, the depth of the 16-bit Knowles PPA (Fig. 13) will make up 11 typical 2-input logic elements, the circuit complexity is 256 elements. One of the variants of the depths of the circuit of the 16-bit Knowles PPA in Fig. 13 is highlighted in bold line, along which the numbering of the logical elements is accompanied by the figures, highlighted in red.

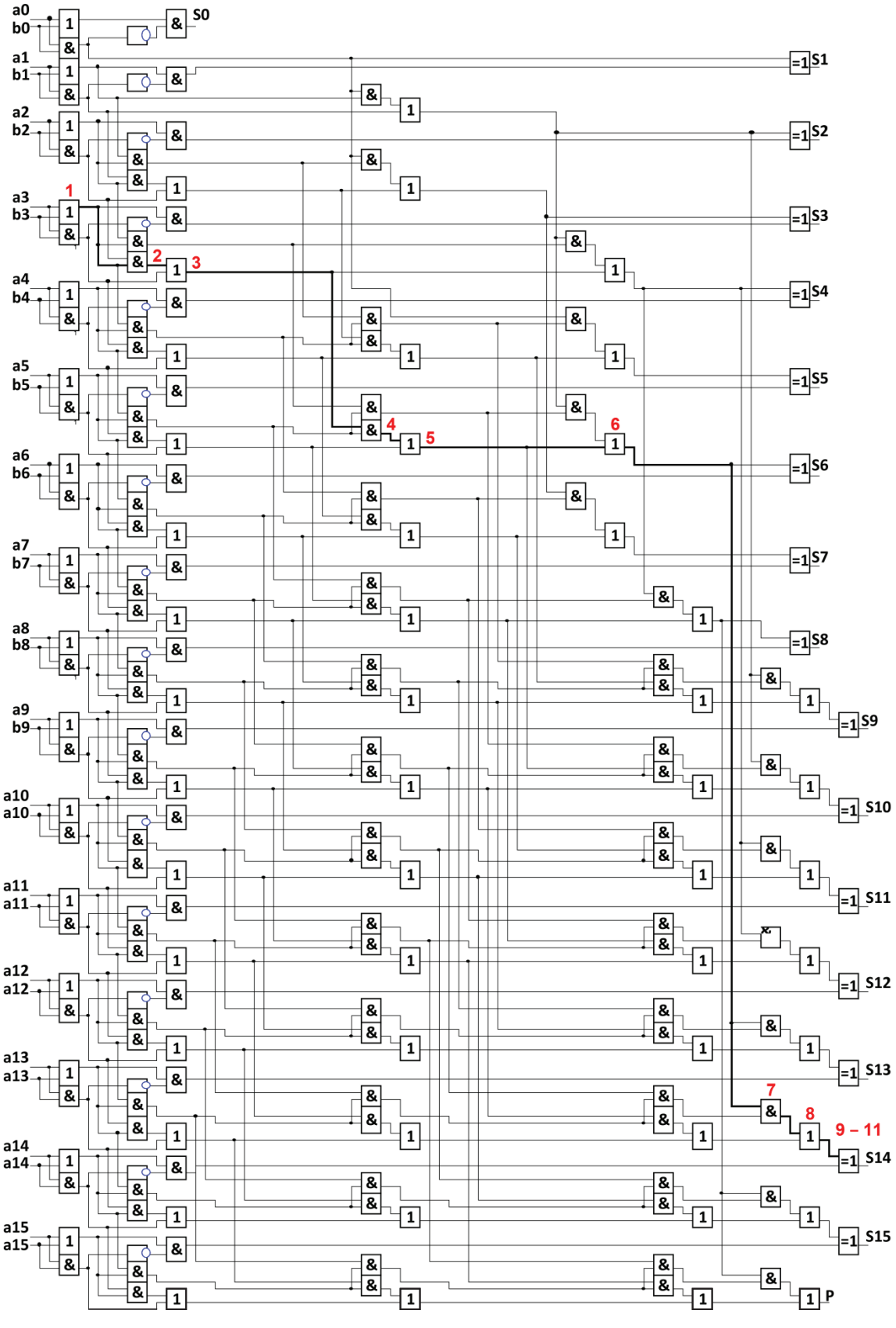

Fig. 13. Prefix 16-bit Knowles PPA with the depth of circuit of 11 typical 2-input elements [18, 19]

The computation process of the 16-bit Knowles PPA (Fig. 13) uses the following logical operations: XOR – 15, AND – 115, OR – 65, Inventor – 16. The 16-bit adder PAA (Fig. 10) uses: XOR – 15, AND – 80, OR – 61, Inventor – 20. Taking into consideration that the logic of the XOR element uses four logical elements, quality indicator *S* (for example,

in terms of power saving) of operation of the 16-bit adder PAA (Fig. 10), compared to the adder in Fig. 13, is as follows:

$$S=\frac{T_1}{T_2}=\frac{256}{221}=1.1584=15.84\ \%,$$

where T_1, T_2 are the number of 2-input logical elements of the 16-bit Knowles PPA (Fig. 13) and of 16-bit PAA (Fig. 10), respectively.

Quality indicator V in terms of computation performance of the PAA (Fig. 10), compared with 16-bit Knowles Adder PPA (Fig. 13) is:

$$V=\frac{N_1}{N_2}=\frac{11}{10}=1.1=10\,\%,$$

where N_1, N_2 are the depth of the circuit of the 16-bit Knowles Adder PPA (Fig. 13) and 16-bit PAA (Fig. 10), respectively.

The 16-bit Knowles Adder PPA (Fig. 13), compared to 16-bit Kogge-Stone Adder PPA (Fig. 12) has a smaller length of connecting tracks. The values of other parameters of these adders are the same.

The prefix 16-bit Sklansky Adder [18, 20] with the logical XOR elements in the last bit and the depth of the circuit of 12 typical 2-input elements is shown in Fig. 14.

Circuit complexity in Fig. 14 makes up 204 2-input elements.

Quality indicator V in terms of calculation performance of the PAA (Fig. 10), compared to the 16-bit Sklansky Adder PPA (Fig. 14) makes up:

$$V=\frac{N_1}{N_2}=\frac{12}{10}=1.2=20\,\%,$$

where N_1, N_2 are the depth of the circuit of the 16-bit Sklansky Adder PPA (Fig. 14) and of the 16-bit PAA (Fig. 10), respectively.

The prefix 16-bit Han-Carlson Adder [18, 21] with the logical XOR elements in the last bit and the circuit depth of 13 typical 2-input logical elements is shown in Fig. 15.

The complexity of the circuit in Fig. 15 is 205 2-input elements.

Quality indicator V in terms of calculation performance of the PAA (Fig. 10), compared to the 16-bit Han-Carlson Adder PPA (Fig. 15) makes up:

$$V=\frac{N_1}{N_2}=\frac{13}{10}=1.3=30\,\%,$$

where N_1, N_2 are the depth of the circuit of the 16-bit Han-Carlson Adder PPA (Fig. 15) and 16-bit PAA (Fig. 10), respectively.

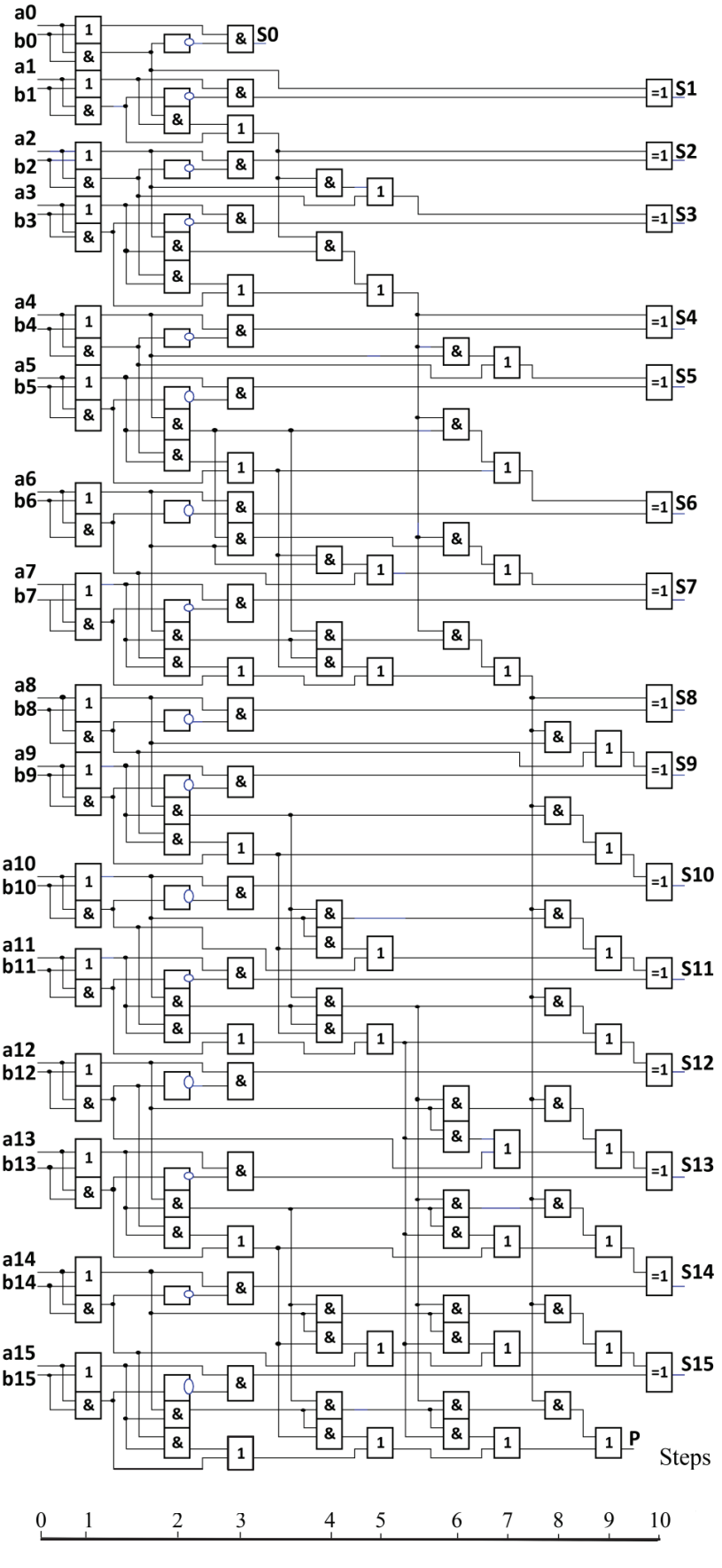

Fig. 14. The prefix 16-bit Sklansky PPA with the circuit depth of 12 typical 2-input elements [18, 20]

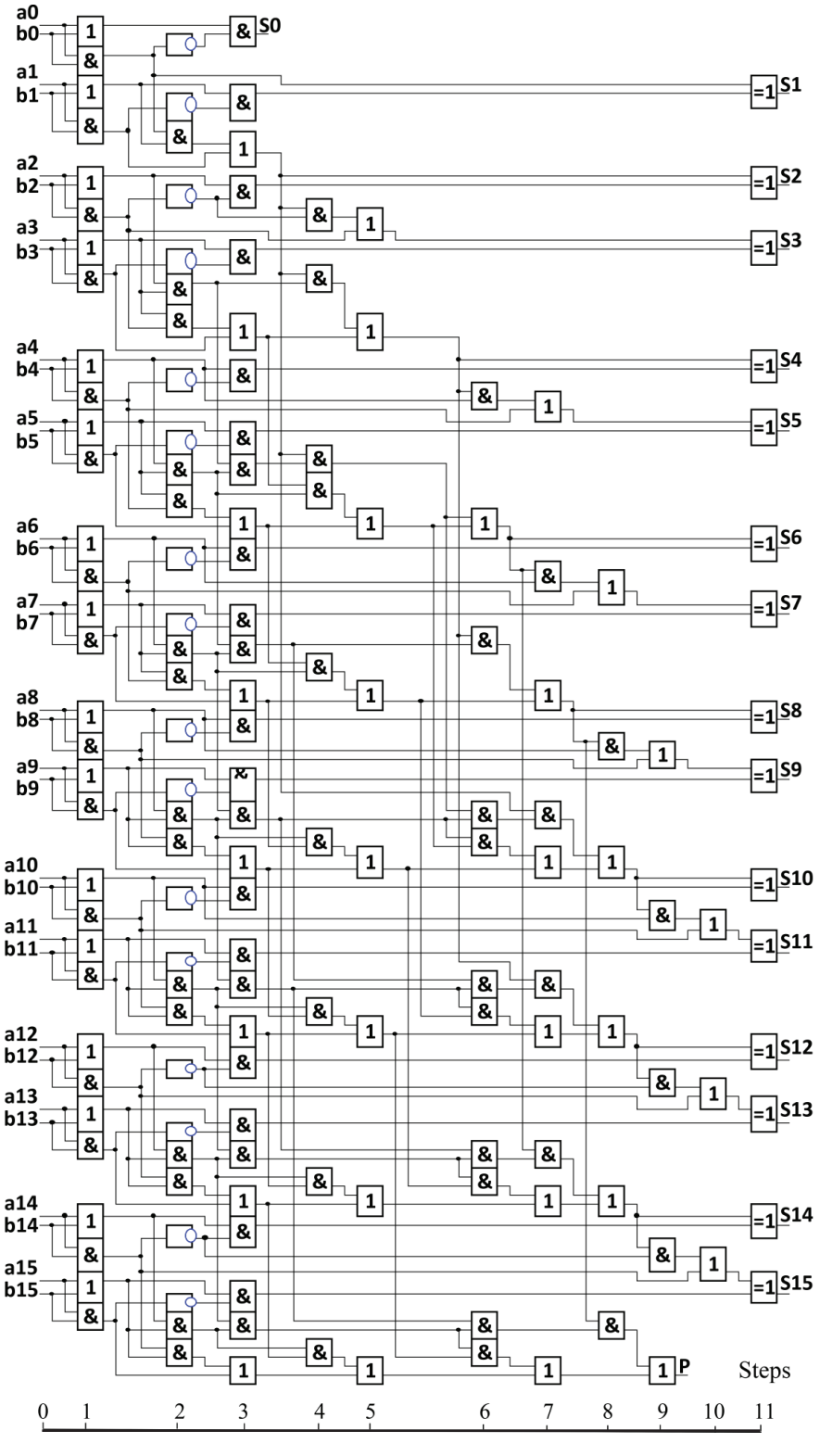

Fig. 15. Prefix 16-bit Han-Carlson PPA with the depth of circuit of 13 typical 2-input elements [18, 21]

The prefix of the 16-bit Ladner-Fischer Adder [18, 22] with the logical XOR elements in the last bit and the depth of the circuit of 13 typical 2-input logical elements is shown in Fig. 16. Complexity of the circuit in Fig. 16 is 190 2-input elements. Quality indicator V in terms of computation performance of the PAA (Fig. 10), compared to the 16-bit Ladner-Fischer Adder PPA (Fig. 16) is:

$$V = \frac{N_1}{N_2} = \frac{14}{10} = 1.4 = 40\,\%,$$

where N_1, N_2 are the depth of the circuit of the 16-bit Ladner-Fischer Adder PPA (Fig. 16) and 16-bit PAA (Fig. 10), respectively.

The prefix 16-bit Brent-Kung Adder [17, 23] with the logical XOR elements in the last bit and the depth of the circuit of 16 typical 2-input logical elements is shown in Fig. 17. The complexity of the circuit in Fig. 17 is 187 2-input elements.

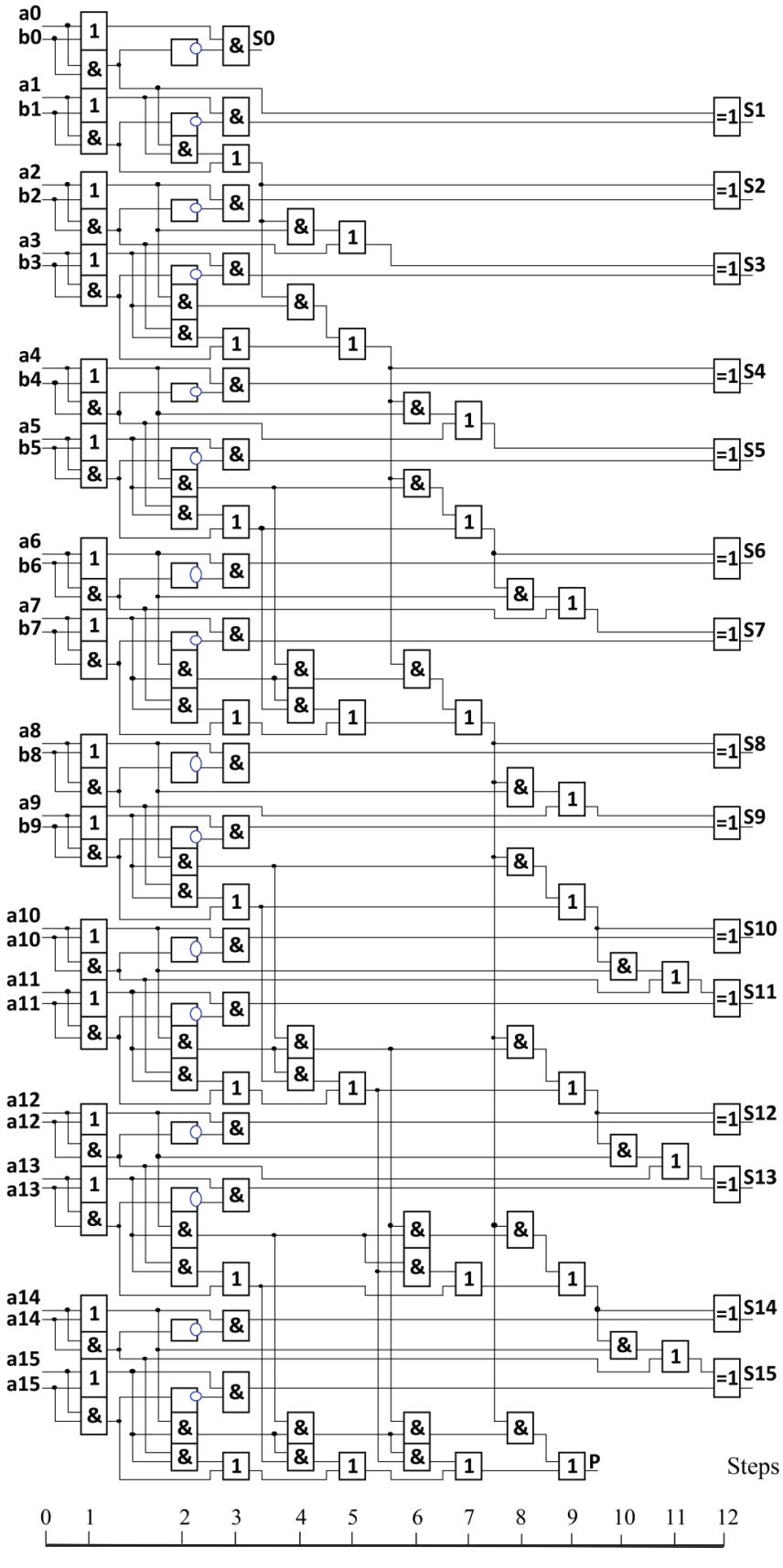

Fig. 16. Prefix 16-bit Ladner-Fischer PPA with the circuit depth of 14 typical 2-input elements [18, 22]

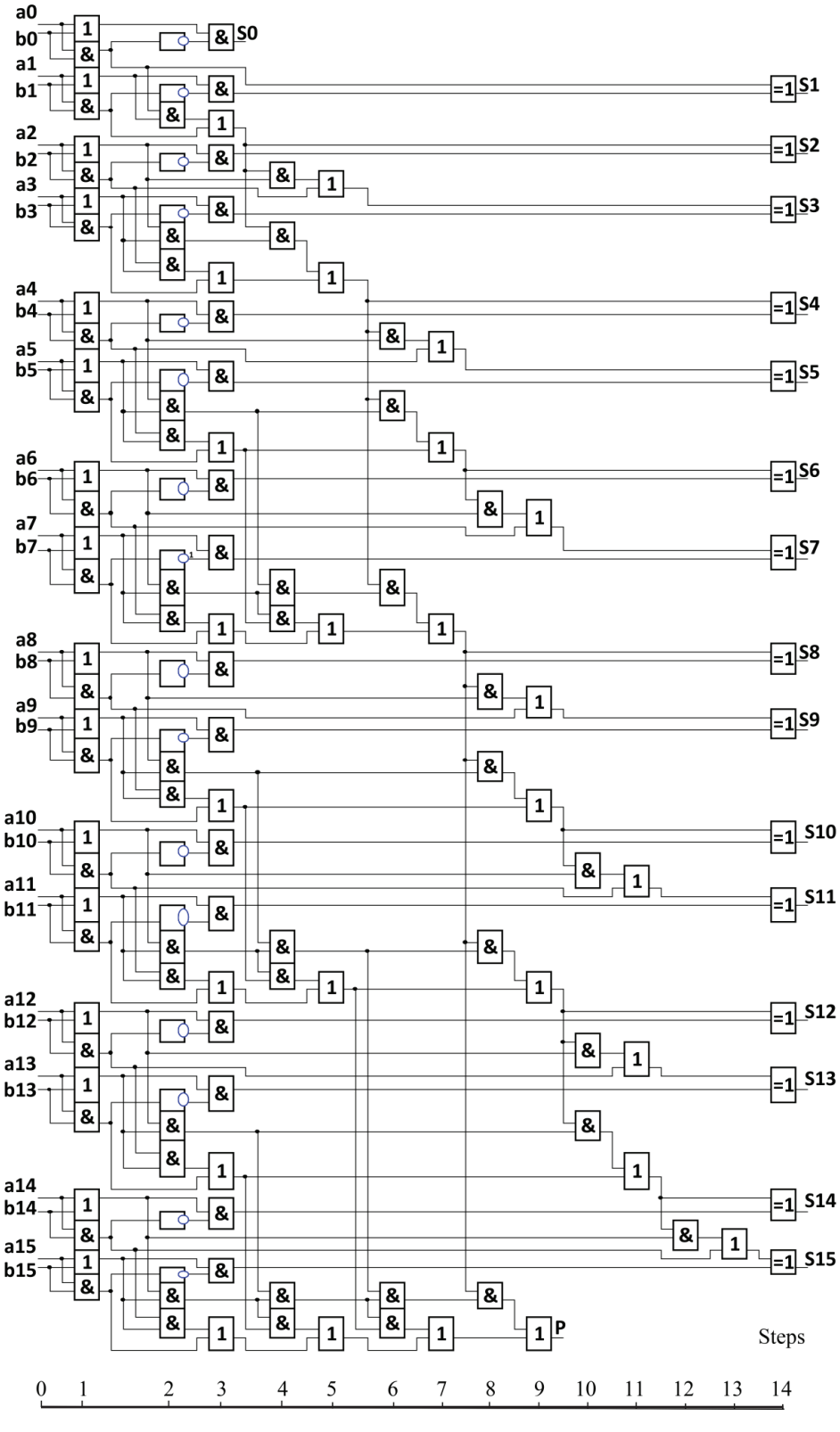

Fig. 17. Prefix 16-bit Brent-Kung PPA with the circuit depth of 16 typical 2-input elements [17, 23]

Quality indicator V in terms of computation performance of PAA (Fig. 10), compared with the 16-bit Brent-Kung Adder PPA (Fig. 17) is:

$$V = \frac{N_1}{N_2} = \frac{16}{10} = 1.6 = 60\,\%,$$

where N_1, N_2 are the depth of the circuit of the 16-bit Brent-Kung Adder PPA (Fig. 17) and of 16-bit PAA (Fig. 10), respectively.

Quality indicators of the 16-bit acyclic adder (PAA) (Fig. 10) and of the 16-bit prefix adders (PPA) (Fig. 11–17) of performance are shown in Table 3.

Table 3

Quality indicators of performance of 16-bit acyclic adder

Parallel adder of binary codes with parallel carry		Circuit depth	Quality indicator of performance of acyclic adder
PAA	Fig. 10	10	–
Ling Adder PPA	Fig. 11	11	10 %
Kogge-Stone Adder PPA	Fig. 12	11	10 %
Knowles Adder PPA	Fig. 13	11	10 %
Sklansky Adder PPA	Fig. 14	12	20 %
Han-Carlson Adder PPA	Fig. 15	13	30 %
Ladner-Fisher Adder PPA	Fig. 16	14	40 %
Brent-Kung Adder PPA	Fig. 17	16	60 %

Considering Table 3, we see that the smallest depth of the circuit of the 16-bit parallel adder with a parallel way of carry belongs to the 16-bit acyclic adder.

Quality indicators of the acyclic adder compared with prefix adders in terms of power consumption are shown in Table 4. Here is the comparison of the 16-bit acyclic adder (PAA) with the circuit depth of 10 logic elements and the 16-bit prefix adders (PPA) with the circuit depth of 11 logical elements.

Table 4

Quality indicators of power consumption of the 16-bit acyclic adder

Parallel adder of binary codes with parallel carry		Circuit complexity	Quality indicator of power consumption of acyclic adder
PAA	Fig. 10	221	–
Ling Adder PPA	Fig. 11	281	27.15 %
Kogge-Stone Adder PPA	Fig. 12	256	15.84 %
Knowles Adder PPA	Fig. 13	256	15.84 %

Considering Table 4, we see that despite a smaller depth (10 logic elements) of the parallel 16-bit acyclic adder, compared with the depth of the circuit of the parallel 16-bit prefix adders (11 logical elements), the lowest complexity of the circuit belongs to the 16-bit parallel acyclic adder (PAA) with the circuit depth of 10 logic elements.

Thus, the architecture of the 16-bit acyclic adder (Fig. 10) shows the optimization of the circuit parameters in terms of delays, area and power without compromises. That is, the structure of the specified adder has both the better computation performance and less power consumption, compared with the architectures of Ling Adder PPA, Kogge-Stone Adder PPA and Knowles Adder PPA which are the end case of a large list of circuits of addition of binary codes, each of which is unique for its property of minimal logical capacity.

7. Discussion of results of application of acyclic model of signal processing for synthesis of 16-bit adders of binary codes

Hardware complexity of an adder depends on the organization of the computation process in its logical model. The prefix model of calculation of the sum and carry signals is shown in Fig. 18 [18].

Considering Fig. 18, we see that:

– the process of parallel computation of prefix starts with lower bits of the adder circuit (this is actually the way (method) of the prefix), which will give in the end the excessive accumulation and complication of the device hardware part;

– the adder circuit depth increases by the extent of an increase in the magnitude of the adder bit, which eventually will also give excessive accumulation and complication of the device hardware part.

In turn, the application of the acyclic model is designed for:

– the process of sequential (for lower bits of the adder circuit) and parallel (for the rest of the bits) computation of the sum and carry signals, which allows decreasing the complexity of the device hardware part and does not increase the logical depth of the circuit;

– fixation (planning) of the adder circuit depth before its synthesis. This makes it possible to use the logical structure of transitive carry, which provides the optimal depth of the adder circuit and does not increase its complexity. An example of such a structure is shown in Fig. 7. The specified logical structure of transitive carry uses the sequential style of relations of logical elements, so the circuit depth increases rapidly. The maximal depth of the adder circuit can be reached at lower bits. However, such process of synthesis decreases the overall complexity of the hardware of the 16-bit digital component, because the structure of transitive carry, like in Fig. 7, is optimal by a number of their elements.

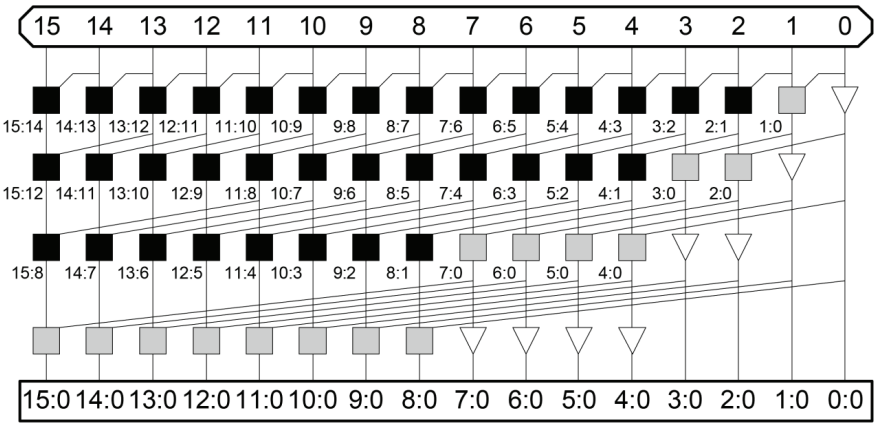

Fig. 18. Prefix 16-bit Kogge Stone Adder [18]

Thus, the use of the acyclic model, compared with the prefix model for the synthesis of the 16-bit circuits of adders of binary codes makes it possible to increase the computation performance by digital components. In particular, the sequential parallel principle of calculation of the acyclic model and fixation (planning) of the depth of the adder circuit before its synthesis ensures the construction of the combining parallel 16-bit adder with the circuit depth of 10 typical 2-input logical elements (Fig. 10). The analog of the specified adder does not exist in the case of the circuit synthesis using the prefix model.

Because the acyclic model demonstrates the 16-bit PAA with the circuit depth of 10 typical 2-input logical elements (Fig. 10), the analogue to which was not found for the structure of the PPA, the principle of enhancing the efficiency of the computations of digital components moves from the prefix to the acyclic model. And therefore, the prospect of further research of digital circuits can be the reassessment of the method of parallel expansion of the calculation process in modern digital devices, re-assessment of the algorithms of adding in the nanometer range, re-assessment of the structure of adders, implemented with memristors, etc.

The weak point of the considered technology of the synthesis of the adder of binary codes is associated with small practice of application of the acyclic model. Negative internal factors that are inherent to the process of designing an adder using the acyclic model are related to the need for additional time costs of making the technological map and equipment of the digital component.

8. Conclusions

1. The optimal logical structure that implements the condition of transitive carry of unity to higher bits in the circuit of the acyclic 16-bit adder of binary codes ensures the least depth of the adder circuit. The specified logical structures are shown in Fig. 6, 7, 9. Such structures make it possible to perform fixation (planning) of the adder circuit depth before its synthesis, which eventually enables a decrease in the general complexity of hardware of the digital component.

That is why the presented examples of logical structures of transitive carry give grounds for the expediency of their application in the processes of synthesis of arithmetic devices for digital data processing, as these structures are able:

– to increase the computation performance in comparison with the analogues;

– to decrease power consumption and heat release of a digital device and the integrated circuit.

2. It was found that the logical depth of the circuit of the acyclic 16-bit adder, synthesized on 2-input logical elements, compared to 8-bit acyclic adder [2] increases by two logical elements. This is proved by the assumption that the protocol of dynamics of increasing the depth of the circuit of the acyclic adder, synthesized on 2-input logical elements is determined by a logarithmic dependence – doubling of bit size of the adder circuit increases the logical depth of the circuit by a constant magnitude – by two logical elements.

3. The effectiveness of the 16-bit acyclic adder with logical XOR elements in the last bit is demonstrated by the examples of the synthesized 16-bit parallel adders, borrowed from the papers of other authors for the purpose of comparison:

– for the circuit of the acyclic 16-bit parallel adder on 2-input elements with the circuit depth of 10 elements (Fig. 10), the analogue of the PPA was not found;

– of the circuit of the prefix Ling Adder (Fig. 11) [13–15], Kogge-Stone PPA (Fig. 12) [16, 17], Knowles PPA [18, 19] (Fig. 13) and the circuit of the acyclic 16-bit parallel adder with the circuit depth of 10 elements (Fig. 10). Power consumption of the 16-bit adder PAA (Fig. 10), compared to Ling Adder (Fig. 11) decreases by 27, 15 %; compared to the adders Kogge-Stone PPA and Knowles PPA (Fig. 12, 13), it decreases by 15.84 %. Performance of the 16-bit adder PAA (Fig. 10), compared with Ling Adder (Fig. 11), Kogge-Stone Adder (Fig. 12), Knowles Adder (Fig. 13) increases by 10 %; compared with Sklansky Adder (Fig. 14) it increases by 20 %; compared with Han-Carlson Adder (Fig. 15) it increases by 30 %; compared with Ladner-Fisher Adder (Fig. 16) it increases by 40 %; compared with Brent-Kung Adder (Fig. 17) it increases by 60 %.

Despite the lower depth of the circuit of the 16-bit acyclic adder (10 logic elements, Fig. 10), compared with the circuit depth (11 logical elements) of parallel 16-bit prefix Ling Adder PPA (Fig. 11), the Kogge-Stone Adder PPA (Fig. 12) and Knowles Adder PPA (Fig. 13), the lowest circuit complexity belongs to the 16-bit parallel acyclic adder with the circuit depth of 10 logic elements. Thus, the 16-bit acyclic adder (Fig. 10) in terms of delay, area and power demonstrates the optimization without compromises. That is, the specified adder has both better computation performance and less power consumption compared to the architectures of Ling Adder PPA, Kogge-Stone Adder PPA and Knowles Adder PPA. The dynamics of increasing the circuit depth of the acyclic adder (PAA), synthesized on 2-input logical elements, shown in Fig. 2, is a tool to control the synthesis of parallel acyclic adders of binary codes.

Taking into account the specified examples of the parallel 16-bit adders, the acyclic model gives grounds for feasibility of its application in the processes of synthesis of 16-bit arithmetic devices of digital data processing, as these circuits can:

– enhance performance;

– decrease power consumption and heat release by a digital device and an integrated circuit.

References

1. Solomko M. Optimization of the acyclic adders of binary codes // Technology audit and production reserves. 2018. Vol. 3, Issue 2 (41). P. 55–65. doi: https://doi.org/10.15587/2312-8372.2018.133694
2. Reduction and optimal performance of acyclic adders of binary codes / Solomko M., Tadeyev P., Zubyk Y., Hladka O. // Eastern-European Journal of Enterprise Technologies. 2019. Vol. 1, Issue 4 (97). P. 40–53. doi: https://doi.org/10.15587/1729-4061.2019.157150
3. Baba Fariddin S., Vargil Vijay E. Design of Efficient 16-Bit Parallel Prefix Ladner-Fischer Adder // International Journal of Computer Applications. 2013. Vol. 79, Issue 16. P. 11–14. doi: https://doi.org/10.5120/13943-1784

4. Design of Prefix Adder Amalgamation Reversible Logic Gates using 16 Bit Kogge Stone Adder / Michael Preetam Raj P., Sandeep B., Sai Mallik Reddy D., Ramanjaneyulu P., Sai Pravallika S. // Indian Journal of Science and Technology. 2016. Vol. 9, Issue 13. doi: https://doi.org/10.17485/ijst/2016/v9i13/87911
5. Shanil Mohamed N., Siby T. Y. 16-bit velocious fault lenient parallel prefix adder // 2014 International Conference on Electronics, Communication and Computational Engineering (ICECCE). 2014. doi: https://doi.org/10.1109/icecce.2014.7086612
6. Poornima N., Bhaaskaran V. S. K. Area Efficient Hybrid Parallel Prefix Adders // Procedia Materials Science. 2015. Vol. 10. P. 371–380. doi: https://doi.org/10.1016/j.mspro.2015.06.069
7. Payal R., Goel M., Manglik P. Design and Implementation of Parallel Prefix Adder for Improving the Performance of Carry Lookahead Adder // International Journal of Engineering Research & Technology. 2015. Vol. 4, Issue 12. P. 566–571. doi: https://doi.org/10.17577/ijertv4is120608
8. Simulation study of brent kung adder using cadence tool / Vamshi Krishna T., Niveditha S., Mamatha G. N., Sunil M. P. // International Journal of Advance Research, Ideas and Innovations in Technology. 2018. Vol. 4, Issue 3. P. 564–573. URL: https://www.ijariit.com/manuscripts/v4i3/V4I3-1383.pdf
9. Design of 16-Bit Adder Structures-Performance Comparison / Padma Balaji R. D., Tarun P., Yeswanth Kumar E., Anita Angeline A. // International Journal of Pure and Applied Mathematics. 2018. Vol. 118, Issue 24. URL: https://acadpubl.eu/hub/2018-118-24/3/492.pdf
10. Kaneko M. A Novel Framework for Procedural Construction of Parallel Prefix Adders // 2019 IEEE International Symposium on Circuits and Systems (ISCAS). 2019. doi: https://doi.org/10.1109/iscas.2019.8702117
11. Sumator z pryskorenym perenosom: Pat. No. 117572U UA. MPK G 06 F 7/38 (2006.01) / Krulikovskyi B. B., Vozna N. Ya., Hryha V. M., Nykolaichuk Ya. M., Davletova A. Ya. No. u201701336; declareted: 13.02.2017; published: 26.06.2017, Bul. No. 12.
12. Gedam S. K., Zode P. P. Parallel prefix Han-Carlson adder // International Journal of Research in Engineering and Applied Sciences. 2014. Vol. 02, Issue 02. P. 81–84. URL: http://www.mgijournal.com/pdf_new/Electronics/Swapna%20Gedam-1.pdf
13. 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, Issue 6. P. 1220–1233. doi: https://doi.org/10.1109/jssc.2010.2048730
14. Govindarajulu S., Vijaya Durga Royal T. Design of Energy-Efficient and High-Performance VLSI Adders // International Journal of Engineering Research. 2014. Vol. 3. P. 55–59. URL: https://pdfs.semanticscholar.org/a54c/5727cdc2be7830ea734f15eb1ba9ecfc2110.pdf
15. Pinto R., Shama K. Efficient shift-add multiplier design using parallel prefix adder // International Journal of Control Theory and Applications. 2016. Vol. 9, Issue 39. P. 45–53.
16. 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. C-22, Issue 8. P. 786–793. doi: https://doi.org/10.1109/tc.1973.5009159
17. Class ECE6332 Fall 12 Group-Fault-Tolerant Reconfigurable PPA. URL: http://venividiwiki.ee.virginia.edu/mediawiki/index.php/ClassECE6332Fall12Group-Fault-Tolerant_Reconfigurable_PPA
18. Two-Operand Addition. URL: https://pubweb.eng.utah.edu/~cs5830/Slides/addersx6.pdf
19. Knowles S. A family of adders // Proceedings 14th IEEE Symposium on Computer Arithmetic (Cat. No.99CB36336). 1999. doi: https://doi.org/10.1109/arith.1999.762825
20. Sklansky J. Conditional-Sum Addition Logic // IEEE Transactions on Electronic Computers. 1960. Vol. EC-9, Issue 2. P. 226–231. doi: https://doi.org/10.1109/tec.1960.5219822
21. Han T., Carlson D. A. Fast area-efficient VLSI adders // 1987 IEEE 8th Symposium on Computer Arithmetic (ARITH). 1987. doi: https://doi.org/10.1109/arith.1987.6158699
22. Ladner R. E., Fischer M. J. Parallel Prefix Computation // Journal of the ACM. 1980. Vol. 27, Issue 4. P. 831–838. doi: https://doi.org/10.1145/322217.322232
23. Brent R., Kung H. T. A Regular Layout for Parallel Adders // IEEE Transactions on Computers. 1982. Vol. C-31, Issue 3. P. 260–264. doi: https://doi.org/10.1109/tc.1982.1675982