ISSN 1063-7397, Russian Microelectronics, 2021, Vol. 50, No. 7, pp. 491–498. © Pleiades Publishing, Ltd., 2021.
Russian Text © The Author(s), 2020, published in Izvestiya Vysshikh Uchebnykh Zavedenii, Elektronika, 2020, Vol. 25, No. 2, pp. 123–135.

Improving the Operating Efficiency of a Multibit Binary Parallel-Prefix Adder

A. N. Yakunin[a, *], Aung Myo San[a], and Han Myo Htun[a]

[a] *National Research University "MIET," Moscow, 124498 Russia*

**e-mail: yakunin.alexey@gmail.com*

Received October 28, 2019; revised October 28, 2019; accepted January 28, 2020

Abstract—For the fast addition of two multibit binary numbers, parallel-prefix adders (PPAs) are currently considered effective. Several PPAs are known with different time and hardware characteristics, and in particular, the Kogge–Stone adder is faster than other PPAs. However, this adder has a large number of logical elements and, therefore, occupies a large area, which leads to an increase in its price. This paper analyzes the Kogge–Stone adder. To reduce its hardware and time costs, a modified PPA is developed. Adders are compared in terms of the occupied area and the maximum delay of an operation. A results' verification scheme is implemented to confirm the reliability of the modified adder's operation. This circuit is simulated in the CAD Altera Quartus-II environment. As a result, it is found that when performing operations with 32- and 64-bit operands, the developed adder reduces the occupied area by 11 and 16.5%, respectively, and the maximum delay by 7%, compared to a Kogge–Stone adder.

Keywords: parallel-prefix adder (PPA), Kogge–Stone adder, footprint, maximum delay

DOI: 10.1134/S106373972107012X

INTRODUCTION

The hardware implementation of binary addition is a fundamental architectural component in microprocessor systems such as microprocessors, digital signal processors, mathematical coprocessors, microcontrollers, mobile devices, etc. In these systems, in the design of operating devices, combiners play an important role in performing many computer arithmetic operations based on addition. In this case, the increase in the performance of operating devices depends on the efficiency of the adder. Developing an efficient adder that delivers high performance is a pressing challenge.

Currently, almost all modern computers in critical paths use parallel-prefix adders (PPAs) [1]. They are highly efficient in terms of speed and are used to quickly add two multidigit binary numbers. The literature describes several PPAs with a different occupied area and maximum delay time [2]. PPAs include Sklansky, Kogge–Stone, Brent–Kung, Han–Carlson, and Ladner–Fischer adders. The Kogge-Stone adder is considered to be the fastest, since it has the smallest propagation delay time in the circuit implementation among these PPAs [3, 4].

The aim of this study is to develop a multibit modified PPA to reduce the hardware and time costs in comparison with the well-known Kogge–Stone adder.

KOGGE–STONE ADDER

The Kogge–Stone adder is considered the fastest standard PPA, which adds two n-bit numbers $A = a_{n-1}a_{n-2}....a_0$, $B = b_{n-1}b_{n-2}....b_0$, and forms an n-bit result, $S = s_{n-1}s_{n-2}....s_0$ and output transfer c_{out}. The Kogge–Stone adder circuits for 8 and 32 bits are shown in Figs. 1 and 2. First the circuits compute the carry generation signals g_i and half-sum bits h_i for bit pairs by logical equations, and $g_i = a_i \cdot b_i$; $h_i = a_i \otimes b_i$, then for nodes using the expression $(g_{i:k}, p_{i:k}) = (g_{i:j} + p_{i:j} \cdot G_{j-1:k}, p_{i:j} \cdot p_{j-1:k})$ until the carry generation signal c_i is not be known for each digit. The sum s_i is determined by signals h_i and c_{i-1} according to $s_i = h_i \oplus c_{i-1}$ through the Exclusive OR element. The Kogge–Stone adder is described in more detail in [5, 6].

The Kogge–Stone adder is faster than the Sklansky, Brent–Kung, Han–Carlson, and Ladner–Fischer adders [7, 8]. However, for the hardware implementation, the Kogge–Stone adder requires more logic elements, hence, an increased area, consumes more power, and is more expensive. Nevertheless, the Kogge–Stone adder is used in high-performance applications, since performance is the most important determining factor.

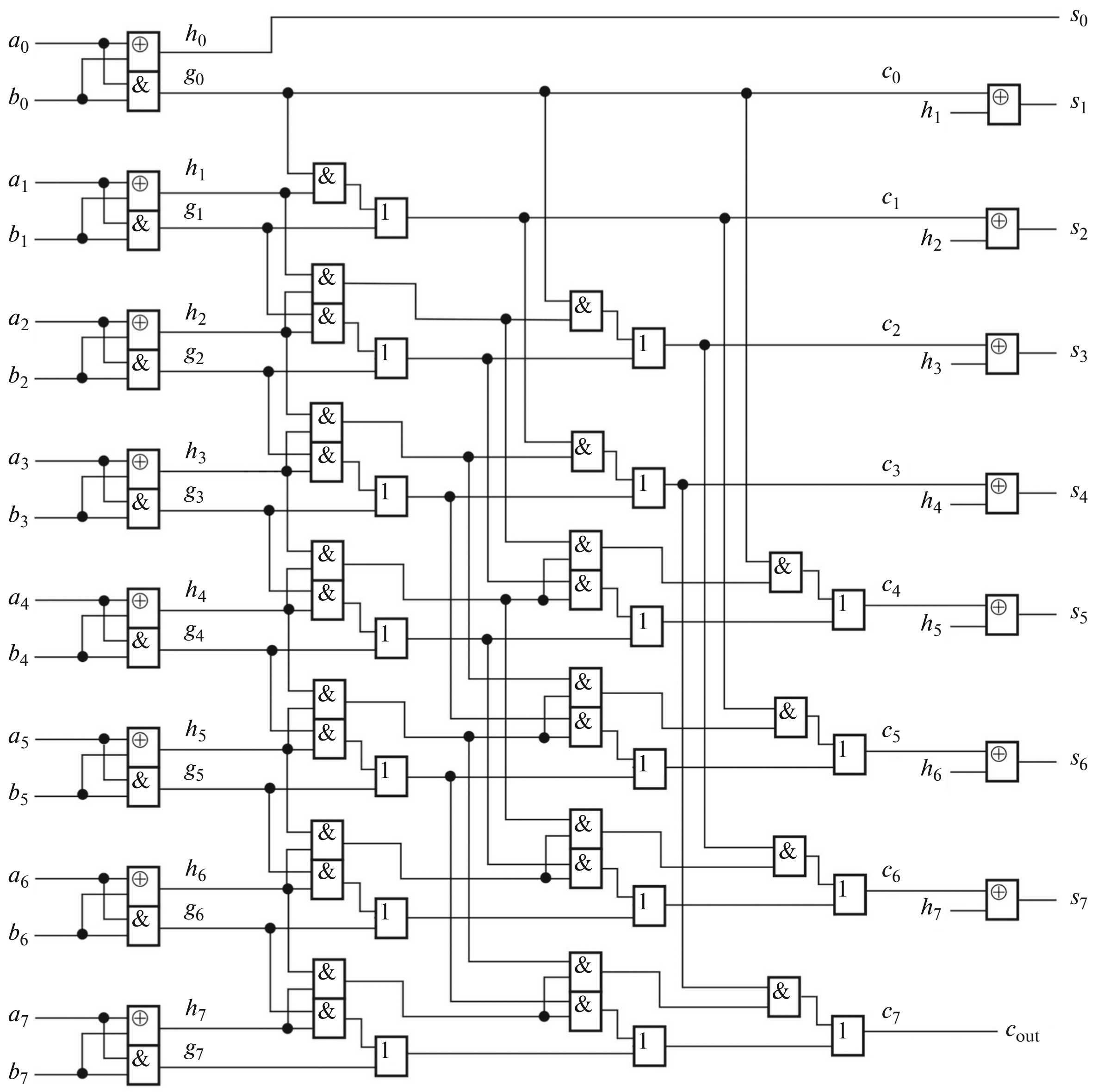

Fig. 1. Logic diagram of the 8-bit Kogge–Stone adder.

DEVELOPMENT OF A MODIFIED PPA

In order to reduce the hardware and time costs, a modified version of the PPA has been developed, which uses its own prefix tree structure. Assume that $A = a_{n-1}a_{n-2}...a_0$ and $B = b_{n-1}b_{n-2}...b_0$ are two binary numbers that will be added up, and $S = s_{n-1}s_{n-2}...s_0$ and c_{out} represent their sum and output carry. The idea of designing a modified adder begins with the logic of generating a propagation carry with a parallel-prefix carry. To calculate the generation g and propagation p of the transfer, we assume that

$$g_i = a_i \cdot b_i, \quad p_i = a_i + b_i, \tag{1}$$

where $i = 0 \le i \le n - 1$; and the symbols "·" and "+" denote the logical operations AND and OR, respectively.

The calculation of the normal carries of the adder is performed as

$$c_i = g_i + p_i \cdot g_{i-1} + p_i \cdot p_{i-1} \cdot g_{i-2} + \ldots + p_i \cdot p_{i-1} \cdot p_{i-2} \cdot \ldots \cdot p_1 \cdot g_0. \tag{2}$$

According to definition (1), it is assumed that $p_i \cdot G_i = g_i$. Then we can perform identical transformations similarly to (2):

$$c_i = (g_i + g_{i-1} + p_{i-1} \cdot g_{i-2} + \ldots + p_{i-1} \cdot p_{i-2} \cdot \ldots \cdot p_1 \cdot g_0)\, p_i. \tag{3}$$

After defining the term of variable k_i, Eq. (3) can be written as

$$c_i = k_i \cdot p_i, \tag{4}$$

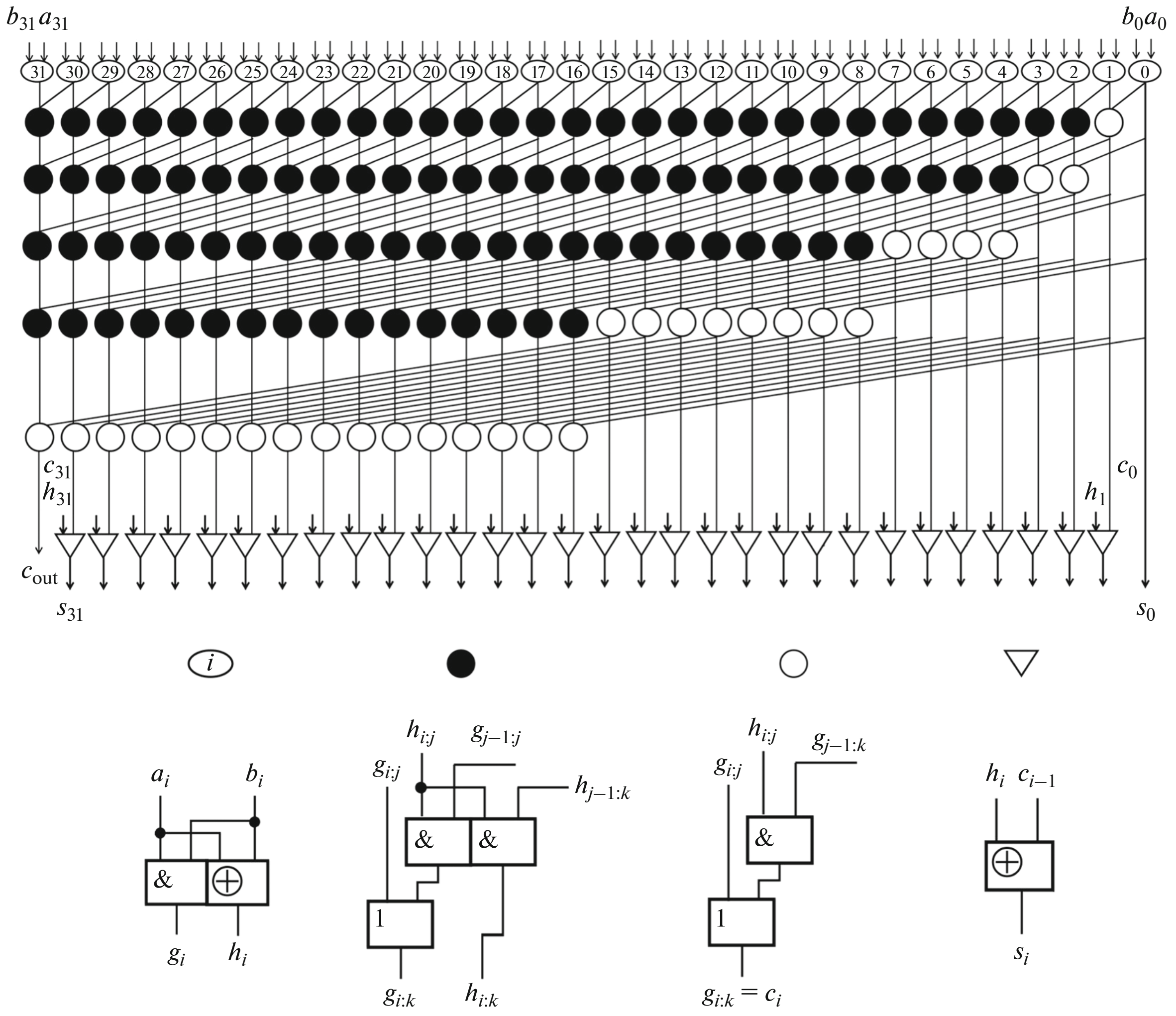

Fig. 2. 32-bit Kogge–Stone adder circuit.

$$\text{where } k_i = g_i + g_{i-1} + p_{i-1} \cdot g_{i-2} + \dots + p_{i-1} \cdot p_{i-2} \cdot \dots \cdot p_1 \cdot g_0. \tag{5}$$

From (4) it follows that the bit switching activity c_i equally depends on the values received by bits p_i of the propagation carry, and values k_i. In particular, for k_5 according to (5) we have

$$k_5 = g_5 + g_4 + p_4 \cdot g_3 + p_4 \cdot p_3 \cdot g_2 + p_4 \cdot p_3 \cdot p_2 \cdot g_1 + p_4 \cdot p_3 \cdot p_2 \cdot p_1 \cdot g_0. \tag{6}$$

Using the definition $p_i \cdot g_i = g_i$, Eq. (6) can be transformed:

$$k_5 = g_5 + g_4 + p_4 \cdot p_3(g_3 + g_2) + p_4 \cdot p_3 \cdot p_2 \cdot p_1(g_1 + g_0). \tag{7}$$

We assume that

$$G_i = g_i + g_{i-1}, \quad P_i = p_i \cdot p_{i-1}, \tag{8}$$

then Eq. (7) is equivalent to the equation

$$k_5 = G_5 + P_4 \cdot G_3 + P_4 \cdot P_2 \cdot G_1.$$

The value of signal k_5 can be formed using the prefix associative operator ° according to the following expression:

$$k_5 = (G_5, P_4) \circ (G_3, P_2) \circ (G_1).$$

Thus, an equation is obtained that realizes the value k_5. In the case of an 8-bit adder, the remaining values $k_7 \dots k_0$ are calculated similarly:

$$\left.\begin{aligned}
k_7 &= G_7 + P_6 \cdot G_5 + P_6 \cdot P_4 \cdot G_3 + P_6 \cdot P_4 \cdot P_2 \cdot G_1 \\
k_6 &= G_6 + P_5 \cdot G_4 + P_5 \cdot P_3 \cdot G_2 + P_4 \cdot P_3 \cdot P_1 \cdot G_0 \\
k_5 &= G_5 + P_4 \cdot G_3 + P_4 \cdot P_2 \cdot G_1 \\
k_4 &= G_4 + P_3 \cdot G_2 + P_3 \cdot P_1 \cdot G_0 \\
k_3 &= G_3 + P_2 \cdot G_1 \\
k_2 &= G_2 + P_1 \cdot G_0 \\
k_1 &= G_1 \\
k_0 &= G_0
\end{aligned}\right\},$$

where $P_1 = p_1$ and $G_0 = g_0$.

The received bits k_i can be represented in the prefix associative operator form:

$$\left.\begin{aligned} k_7 &= (G_7, P_6) \circ (G_5, P_4) \circ (G_3, P_2) \circ (G_1) \\ k_6 &= (G_6, P_5) \circ (G_4, P_3) \circ (G_2, P_1) \circ (G_0) \\ k_5 &= (G_5, P_4) \circ (G_3, P_2) \circ (G_1) \\ k_4 &= (G_4, P_3) \circ (G_2, P_1) \circ (G_0) \\ k_3 &= (G_3, P_2) \circ (G_1) \\ k_2 &= (G_2, P_1) \circ (G_0) \\ k_1 &= (G_1) \\ k_0 &= (G_0) \end{aligned}\right\}. \quad (9)$$

After analyzing (9), by induction we can obtain that the bits k_i are even, and odd bit positions can be expressed as

$$k_{2m} = (G_{2m}, P_{2m-1}) \circ (G_{2m-2}, P_{2m-3}) \circ \ldots \circ (G_0), \quad (10)$$

$$k_{2m+1} = (G_{2m+1}, P_{2m}) \circ (G_{2m-1}, P_{2m-2}) \circ \ldots \circ (G_1). \quad (11)$$

Each associative operator links pairs of signals G and P and is defined as follows:

$$(G, P) \circ (G', P') = (G + P \cdot G'), \ (P \cdot P'). \quad (12)$$

Taking into account the operator $^\circ$, the calculations of signal k_i with a parallel prefix can be represented as a prefix tree of the modified adder. Computing bits k_i and p_i instead of the normal translations c_i makes it harder to get the bits of the final sum s_i since, in this case,

$$s_i = a_i \oplus b_i \oplus c_{i-1} = a_i \oplus b_i \oplus k_{i-1} p_{i-1}.$$

After the identical transformations ($a_i \oplus b_i$) we get

$$\begin{aligned} s_i &= (\overline{a_i} + \overline{b_i})(a_i + b_i) \oplus c_{i-1} \\ &= \overline{a_i b_i}\,(a_i + b_i) \oplus k_{i-1} p_{i-1}, \end{aligned} \quad (13)$$

where the symbols – and $\oplus$ denote the logical operations NOT and Exclusive OR. According to (1) at $\overline{a_i b_i} = \overline{g_i}$ and $a_i + b_i = p_i$, in order to realize the final sum s_i expression (13) can be represented as

$$s_i = \overline{g_i} \cdot p_i \oplus k_{i-1} \cdot p_{i-1}.$$

However, computing bits s_i can be transformed in the following identical way:

$$s_i = \overline{k_{i-1}}\left(\overline{g_i} \cdot p_i\right) + k_{i-1}\left(\left(\overline{g_i} \cdot p_i\right) \oplus p_{i-1}\right). \quad (14)$$

Equation (14) can be implemented using a multiplexer that chooses either $\left(\overline{g_i} \cdot p_i\right)$ or $\left(\left(\overline{g_i} \cdot p_i\right) \oplus p_{i-1}\right)$ according to value k_{i-1}. As a rule, the Exclusive OR gate has an almost equal multiplexer delay and propagation delay time from the input of the input operands to the establishment of the function's result $\left(\overline{g_i} \cdot p_i\right) \oplus p_{i-1}$ is less than before the establishment of the signal k_{i-1}. Therefore, the excess delay will not appear in the adder circuit due to the multiplexer for the output of the final sum s_i. The output carryover c_{out} is formed almost simultaneously with the bits of the sum using the relation $c_{n-1} = k_{n-1} \cdot p_{n-1}$. Thus, according to expressions (1), (8), (10), (11), and (14), the modified adder is implemented by a three-stage circuit consisting of a precomputation stage, a prefix tree formation stage, and a result calculation stage. The logic diagram of the 8-bit modified adder is shown in Fig. 3.

This scheme works as follows. The precomputing stage generates signals g_i and p_i for all categories a_i and b_i using the logical elements AND and OR in accordance with (1). After combining bits g_i, p_i, g_{i-1}, and p_{i-1}, signals G_i and P_{i-1} are calculated based on (8) using the logical elements AND and OR. The second stage adder, or prefix tree, calculates signals k_i using bits G_i and P_{i-1} in accordance with (10) and (11). At the stage of generating the result, the adder calculates sum s_i based on (14) using the elements NOT, AND, Exclusive OR, and the multiplexer. The output carryover c_{out} is formed using the ratio $c_{n-1} = k_{n-1} \cdot p_{n-1}$.

To design a 64-bit modified adder circuit, schematic blocks (black and white) are implemented using (12), which are used for greater clarity of the prefix tree. These blocks take inputs from both the given bit position (${}^{l-1}G_i$, ${}^{l-1}P_{i-1}$), and from the lower bit positions (${}^{l-1}G_i'$, ${}^{l-1}P_{i-1}'$) at the previous level. They are connected to form signals k_i. The network of these blocks is called the prefix tree. Figure 4 shows a schematic of a 64-bit modified adder.

This circuit calculates signals g_i and p_i for pairs of input bits, then calculates signals G_i and P_{i-1} to create a prefix tree. Then $\log_2 n - 1 = 5$ levels of schematic blocks is used to form a prefix tree that calculates signals k_i for each digit. After that, at the last stage, the result of addition s_i and the output transfer c_{out}, together with signals k_i are calculated.

COMPARISON OF HARDWARE AND TIME COSTS A OF PPA

Let us estimate the hardware and time costs of a PPA in terms of the occupied area and maximum delay. To estimate the area occupied by the adder, it is necessary to sum the areas of all the used logic elements. For the maximum delay, the longest (critical) path along which the signal passes is chosen, and the delay time of all logic elements along this path is added [9]. Assume that α_{NOT} and τ_{NOT} are the occupied area and the delay of the NOT element, respectively; α_{AND} and τ_{AND} are the AND element; α_{OR} and τ_{OR} are the OR element; α_{exc} and τ_{exc} are the Exclusive OR element; and α_{m} and τ_{m} are the multiplexer.

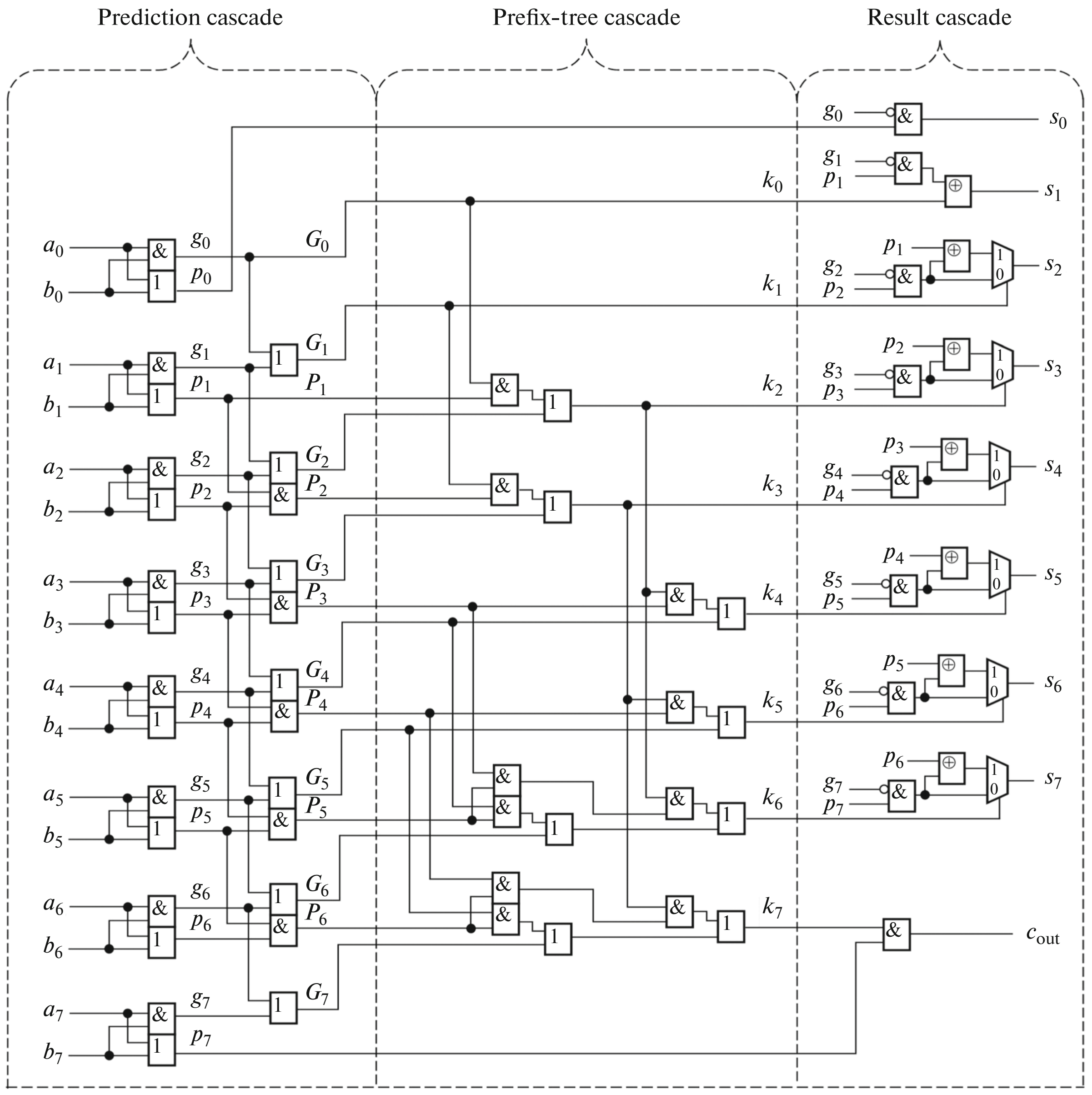
Prediction cascade
Prefix-tree cascade
Result cascade
a_0 b_0 a_1 b_1 a_2 b_2 a_3 b_3 a_4 b_4 a_5 b_5 a_6 b_6 a_7 b_7
g_0 p_0 g_1 p_1 g_2 p_2 g_3 p_3 g_4 p_4 g_5 p_5 g_6 p_6 g_7 p_7
G_0 G_1 P_1 G_2 P_2 G_3 P_3 G_4 P_4 G_5 P_5 G_6 P_6 G_7
k_0 k_1 k_2 k_3 k_4 k_5 k_6 k_7
s_0 s_1 s_2 s_3 s_4 s_5 s_6 s_7
c_{out}

Fig. 3. Logic circuit of an 8-bit modified adder.

After analyzing the structures of the considered PPAs, we obtain the formulas for calculating the occupied area and the maximum delay:

— for the n-bit Kogge–Stone adder

$$\alpha_{\text{KS}} = (2n(\log_2 n) - 2n + 3)\alpha_{\text{AND}} + (n(\log_2 n) - n + 1)\alpha_{\text{OR}} + (2n - 1)\alpha_{\text{exc}},$$

$$\tau_{\text{KS}} = (\log_2 n)\tau_{\text{AND}} + (\log_2 n)\tau_{\text{OR}} + 2\tau_{\text{exc}};$$

— and for the for n-bit modified adder

$$\alpha_{\text{MOD}} = n\alpha_{\text{NOT}} + ((n\log_2 n) + n)\alpha_{\text{AND}} + \left(2n - 1 + \frac{n}{2}(\log_2 n - 1)\right)\alpha_{\text{OR}} + (n-1)\alpha_{\text{EXC}} + (n-2)\alpha_{\text{m}},$$

$$\tau_{\text{mod}} = (\log_2 n)\tau_{\text{AND}} + (\log_2 n)\tau_{\text{OR}} + \tau_{\text{m}}.$$

To quantify the delay, assume that $\tau_{\text{NOT}} = 0.5$, $\tau_{\text{AND}} = \tau_{\text{OR}} = 1$ conventional units, and $\tau_{\text{exc}} = \tau_{\text{m}} = 2$. To estimate the area, we assume that $\alpha_{\text{NOT}} = 0.5$, $\alpha_{\text{AND}} = \alpha_{\text{OR}} = 1$, and $\alpha_{\text{exc}} = \alpha_{\text{m}} = 2$. The obtained values of the occupied area and the maximum delay (operating time) of the PPA for 8, 16, 32, and 64 bits are given in Table 1.

It can be seen from the Table 1 that when increasing the bit depth from the point of view of hardware and time costs, the modified adder is characterized by better performance and a smaller footprint than the Kogge–Stone adder. In particular, the footprint of the 32-bit and 64-bit modified adders is 11 and 16.5% less, respectively, than that of the standard Kogge–Stone

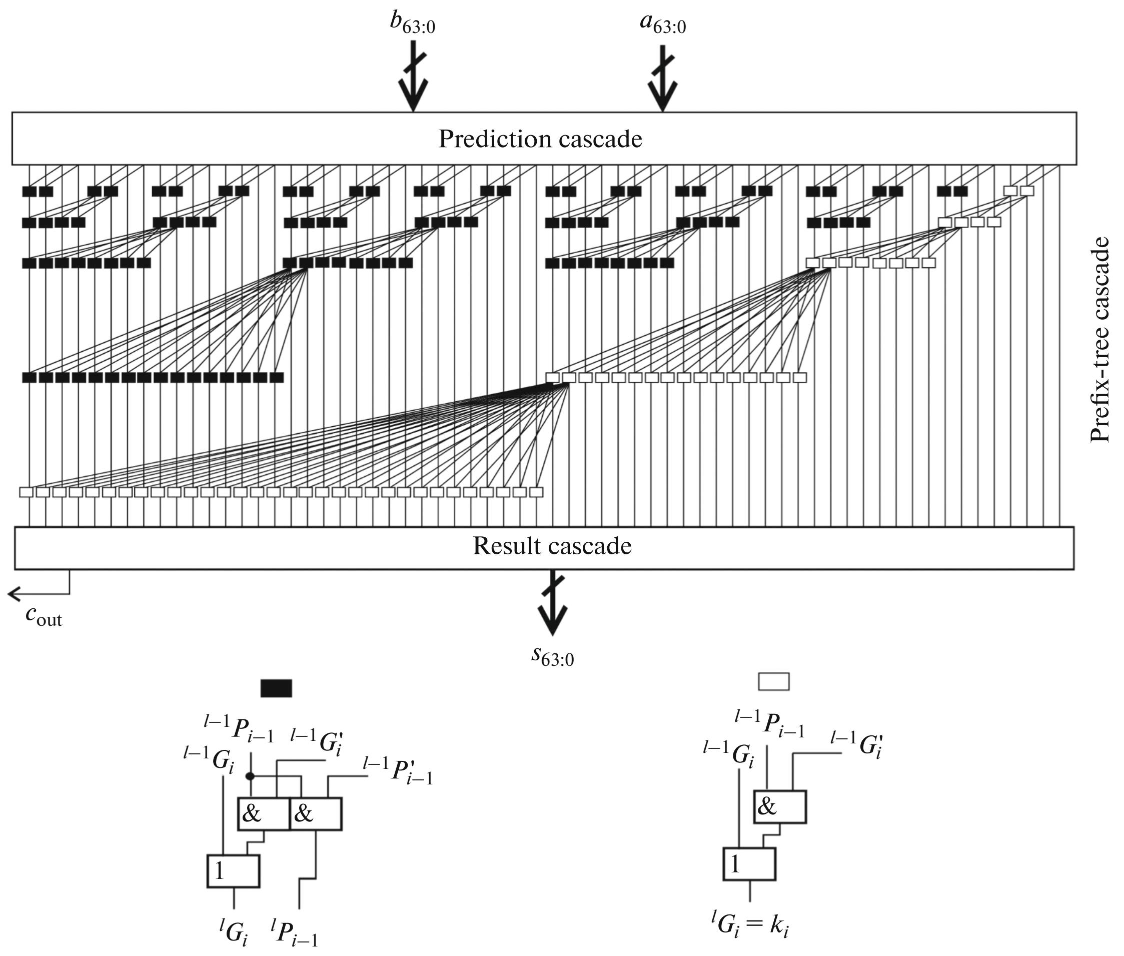

Fig. 4. 64-bit modified adder circuit.

adder. For 32 and 64 bits, the proposed adder gives a decrease in the maximum delay by almost 7% compared to the Kogge–Stone adder.

DEVELOPMENT OF A SCHEME FOR CHECKING THE RESULTS OF THE MODIFIED ADDER

To confirm the reliability of the modified adder's operation, a scheme for checking the reliability of the results is proposed (Fig. 5). The circuit contains an 8-bit modified adder, an 8-bit standard Kogge–Stone adder, a 16-bit synchronous totalizer, a 9-bit digital equality comparator, a four-input AND gate, and two inverters. The synthesis of an adding counter and an equality comparator is considered in [10].

The circuit has inputs—clock signals (*clk*), start (*start*), and reset (*reset*)—and outputs—a 16-bit number ($q_{[15...0]}$), the finish (*Done*), and detected error (*Error*). The input *start* is used to count or stop the counter (1, count; 0, stop). The input *reset* controls the operation of the counter (1 to run, 0 to reset). The calculated counter values are expressed by the output

Table 1. Evaluation of the occupied area and the maximum delay of adders

Number of bits	Occupied area		Maximal delay	
	Kogge–Stone adder	modified adder	Kogge–Stone adder	modified adder
8	82	85	9	8
16	210	201	11	10
32	514	457	13	12
64	1218	1017	15	14

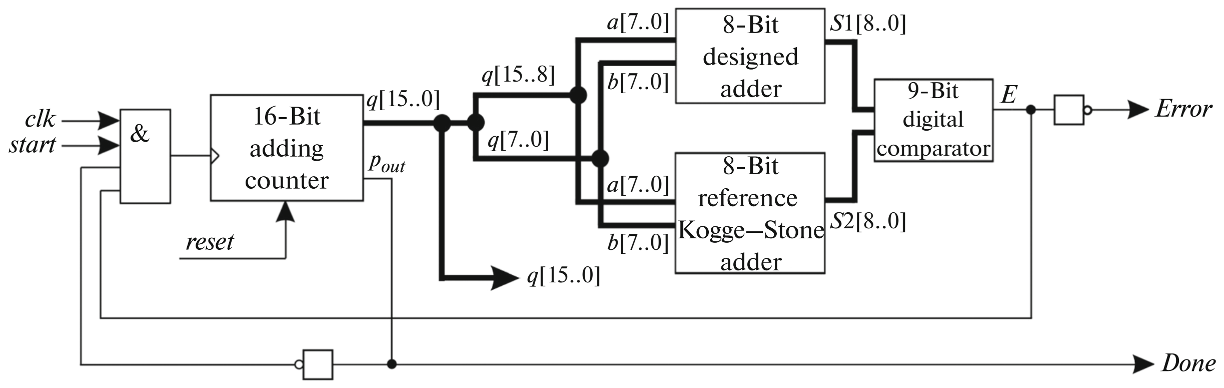

Fig. 5. Circuit for checking the results of the modified adder's operation.

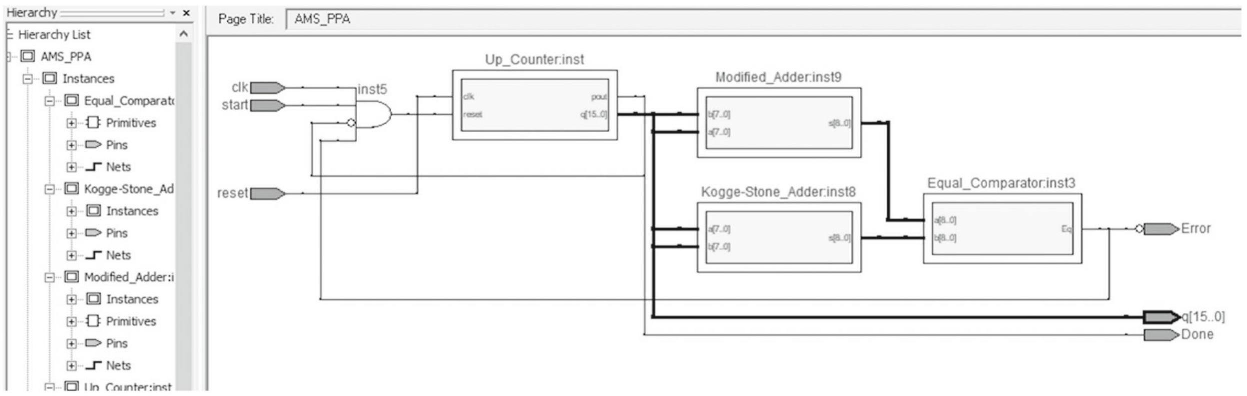

Fig. 6. Scheme of RTL-Viewer results obtained after compilation.

$q_{[15...0]}$. The signal *Done* shows that all possible argument values have been successfully tested as input arguments for two adders. If the signal *Done* = 1, then the process of checking the results was successful; i.e., all the results of the modified adder are correct. The signal *Error* indicates that an error has been detected in some result of the modified adder compared to the result of the reference adder. If there is an error, then the output *Error* = 1. Otherwise the signal *Error* = 0.

Let us consider how the circuit works. Suppose the output signal *E* of the comparator and inversion signal's $\overline{p_{out}}$ output carry counter are 1. In this case *reset* = 1 and *start* = 1, while *clk* enters the clock input of the counter through the AND element. Then the counter calculates all $2^{16} = 65536$ possible values for a 16-bit binary number, and the transition to the next value occurs on the rising edge of the clock pulse. The calculated values of the operands for the counters are defined as $q_{15}...q_8$ for the first term (operand *A*) and $q_7...q_0$ for the second term (operand *B*). The received operands are supplied to the corresponding inputs of the modified and reference adders. These adders perform addition on the input operands in different ways and send 9-bit sums to the outputs. The comparator then compares the two 9-bit binary sums from the adders and provides one output signal indicating whether they are equal or not. If the corresponding digits are equal, then signal *E* = 1 and the process is repeated until the finish is reached. If any discharges *S*1 and *S*2 are unequal, the signal *Error* = 1 and the verification process stops. When the output carry counter P_{out} is logical 1, then the signal *Done* will be set to logical 1; i.e., the verification process ends and the process stops.

MODELING AND VALIDATION OF RESULTS

The proposed scheme was modeled in a graphical editor in the CAD Altera Quartus-II environment. For simulation in the Quartus-II environment, the functional simulation mode was selected using the Cyclone-II-EP2C20F484C7 FPGA family. Figure 6 shows a diagram of the results (version at the level of register transfers of RTL-Viewer) obtained after compiling the diagram. The successful verification of the results of the modified adder operation when adding two 8-bit numbers with all possible values is confirmed by the

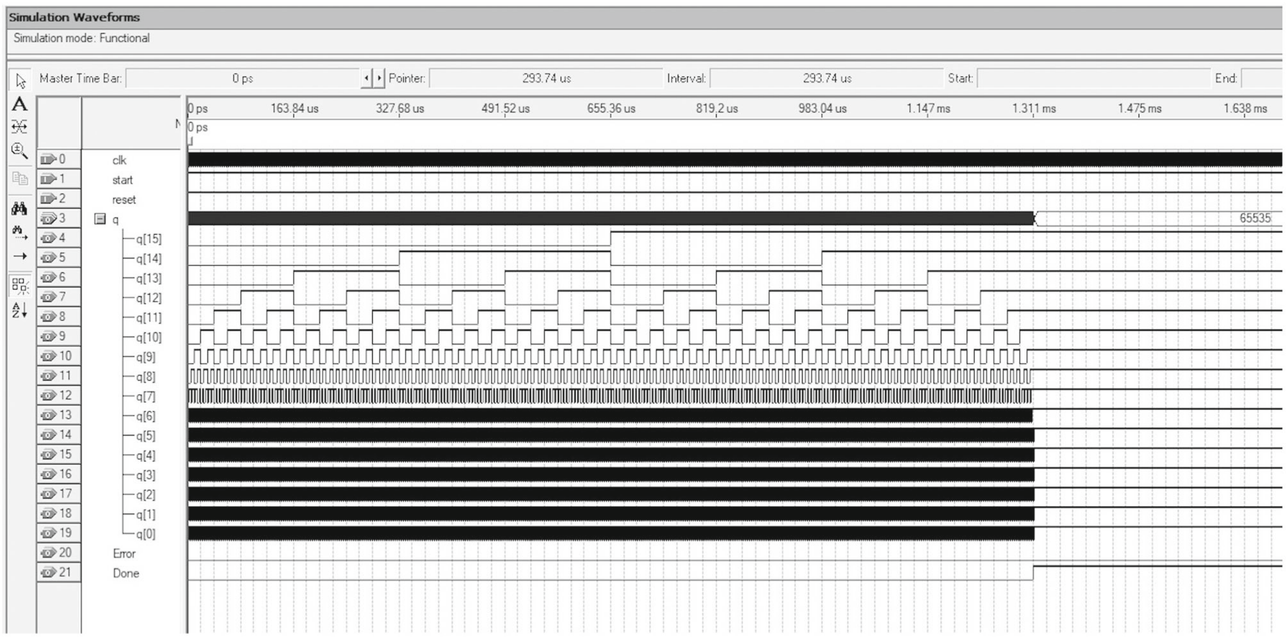

Fig. 7. Time chart of checking the results of the modified adder's operation when adding two 8-bit numbers with all possible values.

timing diagram (Fig. 7) obtained in the Quartus II environment.

CONCLUSIONS

As a result of a comparative analysis of the Kogge–Stone adder and the modified adder, the following points were established: when the number of bits of operands is 16, 32, or 64, the modified adder performs better and occupies a smaller area than the Kogge–Stone adder. The reliability of the results of the modified adder when adding with all possible values of two 8-bit operands was checked using the developed circuit, which was modeled in the CAD Altera Quartus-II environment. The results are confirmed by the time chart.

The proposed modified adder can be used in the design of high-speed operating devices in computer arithmetic.

REFERENCES

1. Yakunin, A.N. and Aung, M.S., Comparative analysis of characteristics of binary multi-bit parallel adders, *Izv. Vyssh. Uchebn. Zaved., Elektron.,* 2018, vol. 23, no. 3, pp. 299–301.
2. Daphni, S. and Vijula Grace, K.S., A review analysis of parallel prefix adders for better performance in VLSI applications, in *Proceedings of the IEEE International Conference on Circuits and Systems, Thiruvananthapuram, India, 2017,* pp. 103–106.
3. Daphni, S. and Vijula Grace, S.K., Design and analysis of 32-bit parallel prefix adders for low power VLSI applications, *Adv. Sci. Technol. Eng. Syst.,* 2019, vol. 4, pp. 102–106.
4. Rahila, KC. and Sajesh Kumar, U., A comprehensive comparative analysis of parallel prefix adders for asic implementation, in *Proceedings of the International Conference on Systems Energy and Environment, GCE Kannur, Kerala, July 2019,* pp. 1–5.
5. Aung, M.S. and Yakunin, A.N., Reduction of the hardware complexity of a parallel prefix adder, in *Proceedings of the International Conference EIConRus-2018, St. Petersburg, Moscow, June 28–31, 2018,* Moscow: MIET, 2018, pp. 1348–1349.
6. Penchalaiah, U. and Siva Kumar, V.G., Design of high-speed and energy-efficient parallel prefix Kogge–Stone adder, in *Proceedings of the IEEE International Conference on System, Computation, Automation and Networking, Pondicherry, India, July 6–7, 2018,* 2018, pp. 1–6.
7. Yakunin, A.N. and Aung, M.S., Increasing the operating speed of a multi-bit binary multiplier, in *Problemy razrabotki perspektivnykh mikro- i nanoelektronnykh sistem MES-2018* (Problems of the Development of Promising Micro- and Nanoelectronic Systems, Proceedings of the 7th All-Russia Conference), 2018, vol. 2, pp. 149–151.
8. Yakunin, A.N. and Aung, M.S., Research and modification of a multi-bit parallel-prefix adder, *Izv. Vyssh. Uchebn. Zaved., Elektron.,* 2019, vol. 24, no. 2, pp. 197–207.
9. Chervyakov, N.I., Lyakhov, P.A., Valueva, M.V., and Krivolapova, O.V., Comparative analysis of adders hardware implementation on FPGA, *Nauka. Innov. Tekhnol.,* 2016, no. 4, pp. 99–108.
10. Harris, D. and Harris, S., *Digital Design and Computer Architecture,* New York: Morgan Kaufmann, 2012.