BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 74(5), 2026, Article number: 1949
DOI: 10.24425/bpasts.2026.1949

Low-power logic function implementation by means of SOP networks

Marcin KUBICA*, Bernard WYRWOŁ, and Dariusz KANIA

Department of Digital Systems, Silesian University of Technology, 44-100 Gliwice, Poland

Abstract. Energy-efficient implementation of digital circuits is a key development direction in modern electronics, especially in the context of the growing demand for mobile and integrated computing systems. Reducing energy consumption not only extends battery life but also reduces heat emission, simplifying the design of cooling systems and increasing system reliability. This article presents a method for designing energy-efficient combinational circuits implemented as SOP networks. The essence of the solution is to modify the classical method known from the literature to minimize the number of switching events across the entire network, considering varying switching probabilities of input signals. Unlike previous works, which assumed identical switching probabilities, the proposed approach incorporates their actual, varying distributions. Additionally, the number of logic levels is accounted for. The main goal of the method is to reduce the switching activity of the SOP network, thus lowering dynamic power consumption. The proposed solution is universal: it allows for the search for energy-efficient configurations regardless of the implementation technology, focusing solely on limiting the number of switching events in the implemented system. The developed algorithm for technology mapping logic functions into energy-efficient SOP networks has been verified using numerous benchmarks. Experimental results confirm its high efficiency and significant potential for dynamic power reduction.

Keywords: low power synthesis; switching activity; SOP; technology mapping.

1. INTRODUCTION

New directions in the development of contemporary digital technology are making certain aspects of digital system design crucial. One of these is the energy efficiency of the designed digital systems. Reducing power consumption extends the operating time of battery-powered devices, increasing their mobility, and also improves the reliability of the systems, particularly important in critical applications such as medicine, the military, or space exploration. Reducing power consumption also reduces the amount of heat generated, mitigating the risk of damage to logical structures. Therefore, in the design of modern systems, including solutions based on artificial intelligence [1], fuzzy logic [2], and IoT technologies [3, 4], it is essential to consider issues related to power reduction.

Given this problem, the question naturally arises as to how power consumption in digital systems can be effectively reduced. Various technological developments aimed at minimizing energy consumption play a significant role here. Undoubtedly, the observed progress in microelectronics, including systematic reductions in supply voltage, contributes to significant reductions in power consumption. However, questions arise regarding the limits of further reductions, as well as high costs and long implementation cycles associated with new technologies. In this situation, the ability to reduce energy consumption at the system design stage becomes particularly important. Therefore, a key question arises: how can digital systems be designed to reduce their energy requirements without interfering with the technological layer? An analysis of the available literature indicates that many interesting and promising solutions have been proposed in this area.

Many works focus on both architecture and technology. A good example is the work demonstrating efficient implementation of arithmetic operations [5]. From a technological perspective, it is possible to reduce static power by using the nonvolatility feature of the magnetic tunnel junction (MTJ) device to implement multi-valued logic [6] and by using carbon nanotube field effect transistors to implement a nonvolatile quaternary flip-flop (NQFF) [7]. This approach can be generalized to the standard quaternary inverter (SQI) [8] or nonvolatile quaternary pre-charged memory [9].

The idea of reducing power consumption is most often associated with the high-level synthesis stage [10–14]. Reducing power consumption at the system level [10, 15–18] yields satisfactory results. Techniques include dividing tasks into software and hardware parts [15, 16, 19] or temporarily disabling selected modules.

Similarly, in the case of low-level synthesis, a number of techniques exist to reduce power consumption. These include methods associated with supply voltage modification, such as power gating [20, 21], or locally reducing the supply voltage [22–25]. Methods that reduce the operating frequency of the system also offer significant benefits: temporary blocking of the clock signal (clock gating) [26–30] and locally reducing the clock signal frequency in selected modules of the designed system [14, 31, 32]. An extension of these ideas is the idea of designing GALS (Glob-

*e-mail: mkubica@polsl.pl

Manuscript submitted 2026-02-01, revised 2026-04-20, initially accepted for publication 2026-06-07, published in September 2026.

ally Asynchronous Locally Synchronous) structures [33–37], where the system is globally asynchronous, with only locally synchronous modules.

From the perspective of low-level synthesis, dividing the implemented system into sequential and combinational parts is essential. In the case of sequential circuits, appropriate encoding of the internal states of the FSM (Finite State Machine) becomes crucial [38–43]. Typically, encoding should be performed in such a way as to reduce the number of memory block switches. Naturally, other concepts for reducing power consumption are also known in the case of circuits, including those presented in [11, 44–47]. The same applies to combinational circuits. Here, the number of switches in the logical network is reduced. This can be achieved through appropriate function decomposition [48–51] or directly by minimizing the circuit area through the output graph [52] and its optimization [53]. In [48], decomposition methods targeted at CPLDs were presented, while in [49], a new method for cutting BDD diagrams that incorporates switching activity was presented, which was generalized using Unicode encoding to a set of functions in [50]. In [51], the earlier idea was supplemented with methods for quickly combining functions into appropriate sets and grouping them into clusters. Satisfactory results can also be obtained by considering specific features of target architectures, as presented by the authors in [54–56] for logic functions implemented based on Sum of Product (SoP) blocks. In [56], the use of output graphs in the mapping process in SOP blocks was proposed, while in [55], a method for optimizing these graphs in terms of switching activity was presented. The paper [54] focuses on the classical method for mapping in SOPs. None of the previous works considered probabilities at the input of the structure other than 0.5, which is of considerable importance, as shown in this paper.

The main contribution of this work is the development of an original algorithm for technology mapping logic functions into a network of SOP blocks, enabling the achievement of a network topology characterized by a minimal number of switching events. Minimizing the number of switching events directly translates into a maximum reduction in dynamic power. The universality of the proposed algorithm stems from its applicability to the implementation of any combinational circuits mapped into a network of SOP blocks, regardless of the varying switching activity of individual inputs. The presented method generalizes previously developed solutions [54,56]. Given the assumed number of products occurring in the SOP block, it leads to a network with a minimal number of switching events, and thus to solutions requiring minimal energy input to implement the logic functions. Unlike previous work, this study includes a detailed analysis of the dynamic properties of the solutions obtained, based on the observation of the number of logical levels in the resulting SOP networks.

The paper is divided into sections. Section 2 discusses the basic concepts associated with dynamic power minimization in digital systems. Additionally, the concept of an SOP block is explained. Section 3 focuses on methods for mapping functions in SOP networks. Section 4 analyzes the impact of individual mappings on the number of logical levels in the SOP network. Section 5 describes the idea of an original mapping that leads to a reduction in network switching activity. This idea is presented in Section 6 in the form of a detailed algorithm. An example explaining the proposed idea is also presented. Section 7 presents experimental results. The paper concludes with a summary and conclusions.

2. THEORETICAL BACKGROUND

The problem of power consumption in digital systems is complex and multifaceted. Total power dissipation consists of two fundamental components: static power (P_{stat}) and dynamic power (P_{dyn}). Static power is closely related to circuit technology, while dynamic power results directly from the implementation and organization of a given project. Further, this article focuses on the analysis of dynamic power, requiring the identification of key factors leading to its minimization. Every digital system has certain constraints related to the supply voltage V_{dd}. Typically, constraints related to the circuit operating frequency (f) can also be considered. A digital system is typically a logical network in which individual nodes (gates, blocks, etc.) are connected by lines characterized by capacitances C_i. Recharging these capacitances requires a certain amount of energy, so it can be concluded that the smaller the logical network, the lower the dynamic power. Considering the above, the dynamic power Pdyn for an n-node logical network can be expressed by equation (1)

$$P_{\text{dyn}} = \sum_{i=1}^{n} \frac{1}{2T} SW_i \cdot C_i V_{dd}^2 . \tag{1}$$

A key factor determining the value of dynamic power is the frequency of capacity reloading in individual network nodes (C_i). A quantity directly related to this phenomenon is switching activity SW_i, which is an element of relationship (1). Switching activity defines the number of logical state changes to the opposite in the i-th node of the analyzed logical network, relative to time. This corresponds to a transition from state 0 to 1 or 1 to 0. The question arises how to determine this parameter. It turns out that knowing the probabilities of the value 1 or 0 occurring in the considered logical network node is crucial. If $p(x)$ denotes the probability of the value 1 occurring, then $1 - p(x)$ will indicate the probability of the value 0 occurring in the considered node. Switching activity can therefore be expressed by relationship (2)

$$SW = 2p(x)(1 - p(x)) \tag{2}$$

Assuming the probabilities p(x) at individual inputs are known and the inputs are independent, it is possible to determine the probability of the value 1 appearing at the output y for basic logic blocks. [54] presents the probabilities $p(y)$ for individual logic gates. It is also possible to determine the probability of the value 1 appearing for more complex functional blocks, such as multiplexers, demultiplexers, etc. One popular way to describe logic functions is the sum-of-products (SOP) form. This form can be represented in SOP blocks. Such a block contains an OR gate with a fixed number of

inputs (expressed by the parameter k), to which AND gates are connected. AND gates also have a fixed number of inputs (expressed by the parameter n). Variables can be connected to the input of the AND gate in either a simple or negated form. The general form of an SOP block is shown in Fig. 1.

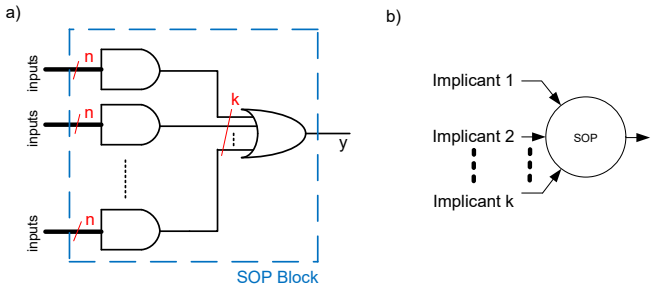


Fig. 1. Sum of products: gate mapping (a), SOP network node (b)

Depending on the values of parameters k and n, the probability of occurrence of the value 1 at the output of the SOP block ($P_{\text{sop}}(y)$) may vary. The same applies to the switching activity of such a block (SW_{sop}). Paper [54] presents the influence of the values of k and n on these parameters (for $k = 3, 5$, and 7). Complex logic functions require implementation not in a single SOP block, but in the form of a network of SOP blocks, which leads to the question of how SOP blocks should be interconnected in the network.

3. SOP NETWORK

There are several ways to create an SOP network, which represents a technological representation of logic functions. A remarkably effective method is the one based on output graphs [55–57]. It provides highly efficient implementations in terms of resource utilization by considering the sharing of logical resources [58]. However, this approach is rather unpopular due to the relatively complex synthesis process. Approaches in which each function is implemented separately are much more common. Two classical methods can be distinguished: cascade and parallel.

The classic cascade method [54] leads to a cascaded block structure (blocks represent SOP vertices), in which the outputs of subsequent blocks are connected to the inputs of subsequent blocks (we assume that blocks have k products). The essence of this implementation is shown in Fig. 2a. The classic parallel method leads to the allocation of blocks in a single logic level. This leads to a situation in which the inputs of a single block are connected to several outputs from blocks located earlier. The essence of this implementation is shown in Fig. 2b.

The properties of such a structure can be described mathematically. Let Δ_f denote the number of implicants of the function $f\colon B^n \to B$ for which the function takes the value 1. In a situation where $\Delta_f > k$, it becomes necessary to create a network of SOP blocks, created in a cascade or parallel form. In each case, the obtained network of SOP blocks containing k-products,

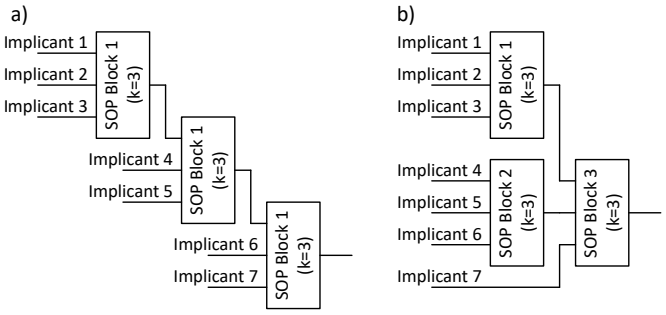


Fig. 2. Implementation of the SOP network: using the classic cascade method (a) and the classic parallel method (b)

forming the function mapping, uses the number of δ_f blocks, which can be determined from the relation (3)

$$\delta_f = \left\lceil \frac{\Delta_f - k}{k-1} \right\rceil + 1, \tag{3}$$

where $\lceil z \rceil$ denotes the smallest natural number not less than z [52].

The classic cascade and parallel methods are easy to implement and are therefore widely used in commercial tools. This raises the question of whether these technological representations are effective in terms of dynamic properties and dynamic power consumption.

4. DYNAMIC PROPERTIES OF THE SOP NETWORK

Let ξ_f denote the number of levels of the SOP network constituting the technology mapping of the function $f\colon B^n \to B$, whose minimal form contains $\Delta_f > k$ implicants.

In the case of mapping logic functions using the classical cascade method, we obtain a structure described by a network of SOP blocks, the number of levels of which is $\xi_f^c = \delta_f$ (Fig. 2a).

The application of the technology mapping of logic functions using the classical parallel method (Fig. 2b) leads to a solution characterized by a significantly smaller number of logic levels. In this case, it is possible to implement it in the form of a structure for which the number of ξ_f levels is the minimal natural number satisfying the inequality $\Delta_f \le k^{\xi_f}$. This allows determining the number of levels of the technology mapping obtained by the ξ_f^p parallel method, based on the relationship (4)

$$\xi_f^p = \lceil \lg_k \Delta_f \rceil. \tag{4}$$

The parallel method of representing logic functions is significantly more dynamic than the cascade method. It leads to a structure with a minimal number of logic levels, making it a viable approach for all other mappings.

Let μ_i be a coefficient indicating how much worse the obtained solution is than the optimal one in terms of dynamic properties. Relation (5) allows for a comparison of different solutions in terms of dynamic properties, where L_i denotes the

number of levels of the mapping considered

$$\mu_i = \frac{L_i}{\lceil \lg_k \Delta_f \rceil}. \tag{5}$$

The resulting solution, an L-level structure, is compared to the technological representation of the logic function using the classical parallel method, which is dynamically optimal, i.e., one that contains the minimum number of levels.

For example, it is possible to compare the technological representation using the classical cascade method with the classical parallel method. The value of the coefficient μ^c, which characterizes the quality of the cascade solution, is then

$$\mu_c = \frac{xi_f^c}{\xi_f^p} = \frac{\left\lceil \frac{\Delta_f - k}{k-1} \right\rceil + 1}{\lceil \lg_k \Delta_f \rceil}. \tag{6}$$

Its value can be interpreted as a measure of how many times greater the number of levels obtained in the mapping using the classical cascade method is compared with the optimal method in terms of the number of levels, namely the classical parallel method. It is worth noting that the value of the coefficient is a function of two arguments: the number of products in the SOP block (k) and the number of implicants of the mapped logic function (Δ_f).

5. TECHNOLOGICAL REPRESENTATION OF COMBINATIONAL CIRCUITS IN THE FORM OF AN ENERGY-SAVING SOP NETWORK

Although the classical methods for implementing combinational circuits in the form of a cascaded or parallel network of SOP blocks, presented in Section 3, utilize a minimal number of blocks, they do not minimize energy consumption. This is because implementing combinational circuits in a structure that utilizes a minimal number of logical resources (minimum number of SOP blocks) does not minimize dynamic power. In this situation, when seeking an energy-efficient solution, it becomes necessary to look not only for an implementation that utilizes the minimal number of SOP blocks but also ensures the minimal number of switches in the resulting network nodes. With dynamic power minimization in mind, it becomes necessary to minimize the number of switches in each node of the system. Considering the entire system implemented in a specific technology, the optimization process comes down to minimizing the total value of the SW coefficient, which is the sum of the SWi coefficients determined for each internal signal in the resulting SOP block network.

In the process of technology mapping of combinational circuits, which is focused on minimizing switching activity (SW), the location of signals with the highest switching capacity in the immediate vicinity of the logical network output plays a key role [54]. This strategy allows for limiting the propagation of logical changes throughout the circuit, thus eliminating unnecessary switching. Therefore, in the case of synthesis of structures based on SOP blocks, the first step should be to identify and map function implicants characterized by low switching activity, which translates into a reduction in the total number of switching events in the logical network.

For any node x in the network of SOP blocks, the probability of the appearance of the logical value "1", denoted as $p(x)$, satisfies the condition: $p(x) \in \langle 0, 1 \rangle$. According to equation (2), the value of the switching ratio SW in node x reaches its maximum for $p(x) = 0.5$.

The function $SW = f(p(x))$ is an even function with respect to the argument shifted to the value 0.5, which can be expressed as $SW\big(p(x)\big) = SW\big(p(x) - 0.5\big)$. The value of SW therefore depends only on the absolute deviation of the probability $p(x)$ from the value 0.5, and not on its direction. In this situation, in the process of logic synthesis of functions implemented as a network of SOP blocks, priority should be given to mapping those implicants or outputs of blocks for which the value $|p(x) - 0.5|$ is the largest, as this leads to minimizing the switching activity of the entire network. To unify the deviation measure, let us introduce the coefficient λ

$$\lambda = \big|p(x) - 0.5\big|. \tag{7}$$

The λ coefficient defines a measure of the switching activity of the analyzed point of the system by determining its distance from the point of maximum switching activity. The value of the λ coefficient is therefore a criterion for ordering individual implicants of the logic function in terms of their impact on the SW value of the final technological representation. In the first phase of the SOP block network construction process, the SOP block should be appended with products associated with implicants for which the maximum values of the λ coefficients occur. In subsequent stages, the SOP block should be appended with products associated with implicants or outputs of already created SOP blocks for which the associated λ coefficients reach maximum values. This type of procedure enables the creation of an SOP block network characterized by the minimum value of the SW switching index.

When creating a technological representation of logic functions in the form of a network of SOP blocks with a specified number of products, the issue of how to manage unused products arises. Since an unused product does not generate network switching, from an energy perspective, it is worth ensuring that all products contained in blocks close to the circuit outputs are used. A consequence of this type of procedure is the need to leave unused products in the block created in the first step of the representation. Therefore, it becomes necessary to determine the number of unused products at the beginning of the representation process. In the case of a technological representation of a function that is the sum of Δ_f – implicants, the number of unused products ε can be determined from equation (8)

$$\varepsilon = \left[k + \left(\delta_f - 1\right)(k-1)\right]. \tag{8}$$

In this situation, the first SOP block should map $k - \varepsilon$ implicants of the implemented function. In light of the above considerations, it becomes possible to formulate an algorithm for mapping

the logic function as an energy-efficient network of SOP blocks. It is evident that the mapping produces a structure with μ_e logical levels, where $\mu_p \leq \mu_e \leq \mu_c$. Considering the above, we can propose an algorithm for technology mapping of the SOP network.

6. ALGORITHM FOR TECHNOLOGY MAPPING OF A LOGICAL NETWORK IN THE FORM OF AN ENERGY-SAVING NETWORK OF SOP BLOCKS

The pseudocode presented below describes the technology mapping algorithm for logic functions represented as a network of SOP blocks containing k-products. Line numbers are provided for reference to specific algorithmic steps.

```
I_or_SOP = espresso(f)                          (1)
foreach i_or_sop in I_or_SOP                    (2)
  λ[i_or_sop] = calculateλ(i_or_sop)
I_or_SOP = sort_max_min(I_or_SOP, λ)            (3)
n = k - ε                                       (4)
sop = selectElements(I_or_SOP, n)               (5)
I_or_SOP = I_or_SOP - inputs(sop)               (6)
I_or_SOP = I_or_SOP + sop                       (7)
while size(I_or_SOP) > 1                        (8)
{
   λsop] = calculateλ(sop)                      (9)
   I_or_SOP = sort_by_insert(sop, λ)           (10)
   sop = selectElements(I_or_SOP, k)           (11)
   I_or_SOP = I_or_SOP - inputs(sop)           (12)
   I_or_SOP = I_or_SOP + sop                   (13)
}
ξ = calculate_levels()                         (14)
```

The **I_or_SOP** array initially contains all implicants and their associated λ coefficients, obtained by minimizing the implemented function **f** using the Espresso method (1). In the next step (2), the λ coefficients are calculated for each implicant, and the entire array is sorted in descending order using the `sort_max_min` function (3). Next, the first **sop** block in the network is implemented; it contains potentially unused products and the first **n** implicants (4) with the largest possible λ coefficients from the **I_or_SOP** array, selected using the **selectElements** function (5). The products used in this first block are then removed from the array (6), and the block itself (the output) is added to it (7). Subsequent **sop** blocks are implemented sequentially in a **while loop** as long as the size of the **I_or_SOP** array – which stores both the original implicants and the outputs of successive blocks – is greater than 1 (it contains the last SOP block) (8). The following operations are performed within the loop: calculating the λ coefficient for the new **sop** block (9), re-sorting the array (using insertion sort, as a full sort is no longer required) (10), and creating another **sop** block from the first **k** elements selected by the **selectElements** function (11). These selected elements are then replaced in the **I_or_SOP** array by the newly created **sop** block (12–13). After exiting the loop, the ξ coefficient, which defines the number of layers, is determined (14).

Example

Let us examine the technological representation of the example function 9 from the 5xp1 benchmark as a network of SOP blocks containing three products (5xp1_f9.pla). Of course, we are interested in the solution for which the *SW* coefficient takes on a minimum value. The result of the function minimization is presented in the file 5xp1_f9.pla.

```
.i 7
.o 1
.ilb a b c d e f g
.ob y
.p 11
11-110- 1
111-10- 1
00-001- 1
000-01- 1
1-1--01 1
11---01 1
0--0-10 1
0-0--10 1
00---10 1
----101 1
----010 1
.e
```

The number of implicants of the minimized function is $\Delta_f = 11$, which makes it necessary to use 5 SOP blocks according to equation (3)

$$\delta_f = \left\lceil \frac{\Delta_f - k}{k-1} \right\rceil + 1 = \left\lceil \frac{11-3}{2} \right\rceil + 1 = 5.$$

According to equation (8), the number of unused products is

$$\varepsilon = \left[k + (\delta_f - 1)\cdot(k-1)\right] - \Delta_f = [3 + 4\cdot 2] - 11 = 0.$$

In this situation, there will be no unused products in the first SOP block, and three implicants will be realized. The question then arises as to which implicants should be represented in the first block.

The columns of Table 1 contain the numbers of implicants (i1, i2, . . . , i11) described in the file 5xp1_f9.pla together with the assigned values of probabilities $p(x)$ for the inputs (a, b, . . . , g) of individual implicants and the values of the coefficient λ characterizing the "distance" of the probability values from the value 0.5.

To generalize the considerations, in the analyzed example, it was assumed that the input signals are associated with different values of the probability of the appearance of the value 1. These values are placed in the row above the designations of the input signals.

According to the algorithm for mapping functions in SOP blocks (p. 3), in the first step of technology mapping, the implicants are sorted according to the decreasing value of the λ coefficient. The sorting result is presented in Table 2.

In the first SOP block, the first three implicants i2, i1, and i6 are implemented, for which the values of the coefficient λ as-

Table 1
Parameter values of the analyzed function used in the process of mapping the function in SOP blocks

$p(x)$	0.1	0.2	0.4	0.5	0.6	0.8	0.9		
	a	b	c	d	e	f	G	$p_{\mathrm{AND}}(y)$	λ
i1	0.1	0.2	1	0.5	0.6	0.2	1	0.00120	0.49880
i2	0.1	0.2	0.4	1	0.6	0.8	1	0.00096	0.49904
i3	0.9	0.8	1	0.5	0.4	0.8	1	0.11520	0.38480
i4	0.9	0.8	0.6	1	0.4	0.8	1	0.13824	0.36176
i5	0.1	1	0.4	1	1	0.2	0.9	0.00720	0.49280
i6	0.1	0.2	1	1	1	0.2	0.9	0.00360	0.49640
i7	0.9	1	1	0.5	1	0.8	0.1	0.03600	0.46400
i8	0.9	1	0.6	1	1	0.8	0.1	0.04320	0.45680
i9	0.9	0.8	1	1	1	0.8	0.1	0.05760	0.44240
i10	1	1	1	1	0.6	0.2	0.9	0.10800	0.39200
i11	1	1	1	1	0.4	0.8	0.1	0.03200	0.46800

Table 2
The result of sorting the implicants of the analyzed function according to the decreasing value of the index λ

$p(x)$	0.1	0.2	0.4	0.5	0.6	0.8	0.9		
	a	b	c	d	E	f	g	$p_{\mathrm{AND}}(y)$	λ
i2	0.1	0.2	0.4	1	0.6	0.8	1	0.00096	0.49904
i1	0.1	0.2	1	0.5	0.6	0.2	1	0.00120	0.49880
i6	0.1	0.2	1	1	1	0.2	0.9	0.00360	0.49640
i5	0.1	1	0.4	1	1	0.2	0.9	0.00720	0.49280
i11	1	1	1	1	0.4	0.8	0.1	0.03200	0.46800
i7	0.9	1	1	0.5	1	0.8	0.1	0.03600	0.46400
i8	0.9	1	0.6	1	1	0.8	0.1	0.04320	0.45680
i9	0.9	0.8	1	1	1	0.8	0.1	0.05760	0.44240
i10	1	1	1	1	0.6	0.2	0.9	0.10800	0.39200
i3	0.9	0.8	1	0.5	0.4	0.8	1	0.11520	0.38480
i4	0.9	0.8	0.6	1	0.4	0.8	1	0.13824	0.36176

sume maximum values. After the first step of technology mapping, the successively created network of SOP blocks is shown in Fig. 3.

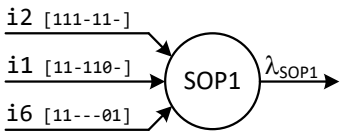


Fig. 3. Technological representation of the analyzed function after creating the first SOP block

Then, after first determining the probability of the value 1 appearing in the output of the created SOP1 block, it becomes possible to determine the value of the λ_{SOP1} coefficient (p. 6), which is 0.00575. In the next step (p. 7), the values of the λ coefficients are sorted, including the λ_{SOP1} coefficient, which represents the output of the previously created block. The sorting result is presented in Table 3. It includes the unmapped implicants and the output of the previously created SOP1 block.

Table 3
Result of sorting the unmapped implicants of the analyzed function according to the decreasing value of the index λ

$p(x)$	0.1	0.2	0.4	0.5	0.6	0.8	0.9		
	a	b	c	d	e	f	g	$p_{\mathrm{AND}}(y)$	λ
sop1								0.00575	0.49425
i5	0.1	1	0.4	1	1	0.2	0.9	0.00720	0.49280
i11	1	1	1	1	0.4	0.8	0.1	0.03200	0.46800
i7	0.9	1	1	0.5	1	0.8	0.1	0.03600	0.46400
i8	0.9	1	0.6	1	1	0.8	0.1	0.04320	0.45680
i9	0.9	0.8	1	1	1	0.8	0.1	0.05760	0.44240
i10	1	1	1	1	0.6	0.2	0.9	0.10800	0.39200
i3	0.9	0.8	1	0.5	0.4	0.8	1	0.11520	0.38480
i4	0.9	0.8	0.6	1	0.4	0.8	1	0.13824	0.36176

Next, another SOP block is added, in which two implicants i5 and i11 are implemented. The third product is used to connect the output of the SOP1 block. The selection of the above signals is a consequence of sorting the λ values, which indicated three signals that guarantee minimal switching of the created block. The current mapping result is shown in Fig. 4. The determined value $\lambda_{\mathrm{SOP2}} = 0.45550$.

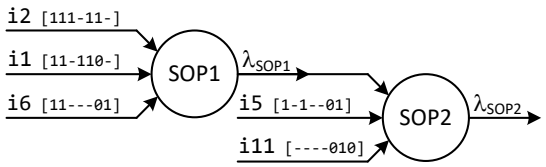


Fig. 4. Technological representation of the analyzed function after creating two SOP blocks (SOP1 and SOP2)

The result after sorting the values of the λ coefficients indicates that in the next step the i7, i8 implicants are realized, while the third product was used to connect the output of the SOP2 block, which is the result of the previously obtained λ_{SOP2} coefficient. The result of the mapping is presented in Table 4 and graphically in Fig. 5, and the determined value $\lambda_{\mathrm{SOP3}} = 0.24378$.

In the next step, three more implicants i9, i10 and i3 are realized. The table justifying the next step of creating the SOP

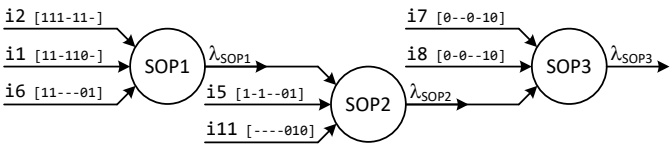


Fig. 5. Technological representation of the analyzed function after creating three SOP blocks (SOP1, SOP2 and SOP3)

Table 4
Sorting result after creating the SOP2 block

$p(x)$	0.1	0.2	0.4	0.5	0.6	0.8	0.9		
	a	b	c	d	e	f	g	$p_{\mathrm{AND}}(y)$	λ
i7	0.9	1	1	0.5	1	0.8	0.1	0.03600	0.46400
i8	0.9	1	0.6	1	1	0.8	0.1	0.04320	0.45680
sop2								0.04449	0.45550
i9	0.9	0.8	1	1	1	0.8	0.1	0.05760	0.44240
i10	1	1	1	1	0.6	0.2	0.9	0.10800	0.39200
i3	0.9	0.8	1	0.5	0.4	0.8	1	0.11520	0.38480
i4	0.9	0.8	0.6	1	0.4	0.8	1	0.13824	0.36176

block network, together with the network structure, is presented in Table 5 and Fig. 6, respectively. The determined value of λ_{SOP4} is 0.36176.

Table 5
Sorting result after creating the SOP3 block

$p(x)$	0.1	0.2	0.4	0.5	0.6	0.8	0.9		
	a	b	c	d	e	f	g	$p_{\mathrm{AND}}(y)$	λ
i9	0.9	0.8	1	1	1	0.8	0.1	0.05760	0.44240
i10	1	1	1	1	0.6	0.2	0.9	0.10800	0.39200
i3	0.9	0.8	1	0.5	0.4	0.8	1	0.11520	0.38480
sop3								0.11868	0.38131
i4	0.9	0.8	0.6	1	0.4	0.8	1	0.13824	0.36176

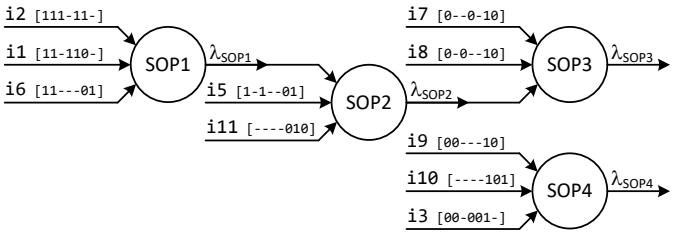


Fig. 6. Technological representation of the analyzed function after creating four SOP blocks (SOP1, SOP2, SOP3 and SOP4)

In the last stage, the last implicant i4 is realized in the SOP5 block, in which the remaining products are used to connect the outputs of the SOP3 and SOP4 blocks as shown in Table 6.

Table 6
Sorting result after creating the SOP4 block

$p(x)$	0.1	0.2	0.4	0.5	0.6	0.8	0.9		
	a	b	c	d	e	f	g	$p_{\mathrm{AND}}(y)$	λ
sop3								0.11868	0.38131
i4	0.9	0.8	0.6	1	0.4	0.8	1	0.13824	0.36176
sop4								0.25622	0.24378

The final form of the technological representation of the function, for which the total value of the SW coefficient is 2.84837, is presented in Fig. 7.

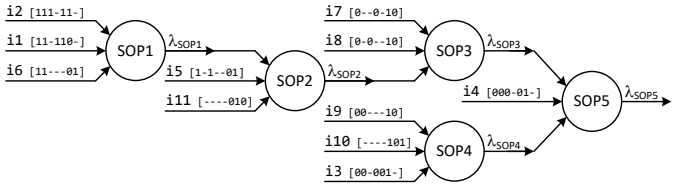


Fig. 7. Network of SOP blocks representing an example function aimed at minimizing switching activity

Comparing the obtained solution with alternatives, it is worth noting that much depends on the initial ordering of the implicants. In the case of the classic cascade implementation, which is a 5-level structure, obtained directly after function minimization (Cl_cascade), i.e., without initial ordering of the implicants (the implicant system is presented in Table 1), we obtain a value of the coefficient $SW = 4.32887$, which results in dynamic power losses being 51.2% higher than the proposed energy-efficient solution. A slightly better result was obtained for the parallel mapping (Cl_parallel), with the total value of the coefficient $SW = 3.75092$ (dynamic power losses are 31.7% higher). Significantly better results are obtained after initial ordering of the implicants (the ordering is presented in Table 2). This time, for the cascade structure (Cl_cascade_sort), which is of course also a 5-level structure, the total SW value = 2.85404 (only a 0.2% increase compared to the energy-efficient implementation). Unlike in the previous sorting, the parallel structure (Cl_parallel_sort), which is, as before, a 3-level structure, is much worse in terms of energy, because the value of the SW coefficient is 3.16779, which causes dynamic power losses to be 11.2% higher compared to the obtained energy-efficient structure.

7. EXPERIMENTAL RESULTS

The effectiveness of the proposed algorithm was verified by performing a series of experiments. These were conducted using a popular benchmark set [59] and dedicated software developed for determining switching activity. To maintain reliability and enable comparison of the obtained results with previous works [54–56], only selected functions (as in previous works) were used in each benchmark. The experiments were divided into three series for different numbers of SOP block products. We selected blocks with 3 products ($k = 3$), 5 products ($k = 5$), and 7 products ($k = 7$).

First, we analyzed the strategies for assigning the probability $p(x_i)$ of state 1 for given inputs x_i for the proposed approach. Three probability distribution approaches $p(x_i)$ were adopted:

- $\mathrm{P}_{[0.5]}$ – the probability of state 1 for each input is constant and equal to 0.5; for the remaining inputs, the probabilities are selected from the range [0.1, 0.9], with the probabilities for subsequent inputs.
- $\mathrm{P}_{[0.1,0.9]}$ – ordered in ascending order (as in the example from Section 6).

- $P_{[0.9,0.1]}$ – ordered in descending order.

Tables 7, 8, and 9 summarize the results obtained for SOP blocks with 3, 5, and 7 products, respectively.

Table 7
ummary of network parameters for SOP blocks with $k = 3$ and selected probability assignment strategies

	Benchmark			$P_{[0.5]}$					$P_{[0.1\text{-}0.9]}$					$P_{[0.9\text{-}0.1]}$				
	Name	In	Out	SW	ξ_f^p	ξ_f^c	ξ_f^e	μ_e	SW	ξ_f^p	ξ_f^c	ξ_f^e	μ_e	SW	ξ_f^p	ξ_f^c	ξ_f^e	μ_e
1	5xp1	7	10	28.957	3	9	4	1.33	20.432	3	9	7	2.33	19.133	3	9	7	2.33
2	bw_21_11_9	5	3	6.098	2	3	3	1.50	4.181	2	3	2	1.00	4.082	2	3	3	1.50
3	bw_27_26_25	5	3	5.059	2	2	2	1.00	3.416	2	2	2	1.00	4.232	2	2	2	1.00
4	bw_321	5	3	7.315	2	2	2	1.00	6.172	2	2	2	1.00	4.810	2	2	2	1.00
5	bw_432	5	3	5.987	2	2	2	1.00	5.197	2	2	2	1.00	3.509	2	2	2	1.00
6	clip_432	10	3	26.152	4	21	5	1.25	18.066	4	21	8	2.00	21.526	4	21	7	1.75
7	ldd_mno	9	3	5.960	2	3	3	1.50	11.360	2	3	2	1.00	0.855	2	3	3	1.50
8	misex1_3456	8	4	7.283	2	3	3	1.50	6.688	2	3	3	1.50	2.584	2	3	3	1.50
9	misex1_356	7	3	6.215	2	3	3	1.50	6.268	2	3	3	1.50	2.166	2	3	3	1.50
10	misex1_3567	7	4	7.921	2	3	3	1.50	7.225	2	3	3	1.50	3.177	2	3	3	1.50
11	misex1_367	7	3	5.623	2	3	3	1.50	4.749	2	3	3	1.50	2.265	2	3	3	1.50
12	misex1_4567	8	4	7.590	2	3	3	1.50	6.378	2	3	3	1.50	3.115	2	3	3	1.50
13	misex1_567	8	3	6.522	2	3	3	1.50	5.904	2	3	3	1.50	2.585	2	3	3	1.50
14	rd53	5	3	10.062	3	8	3	1.00	6.420	3	8	6	2.00	6.496	3	8	6	2.00
15	rd73	7	3	29.199	4	32	4	1.00	18.838	4	32	9	2.25	19.795	4	32	9	2.25
16	sao2_210	10	3	15.852	3	10	3	1.00	5.266	3	10	6	2.00	15.628	3	10	6	2.00
17	sao2_310	10	3	15.802	3	11	3	1.00	5.273	3	11	5	1.67	15.511	3	11	5	1.67
18	sao2_320	10	3	5.968	3	10	3	1.00	1.537	3	10	6	2.00	6.340	3	10	6	2.00
19	sao2_321	10	3	10.444	3	10	3	1.00	3.791	3	10	6	2.00	9.954	3	10	6	2.00
20	sao2	10	3	16.022	3	10	3	1.00	5.289	3	10	6	2.00	15.811	3	10	6	2.00
21	sqn	7	3	13.301	3	9	4	1.33	5.171	3	9	5	1.67	13.800	3	9	7	2.33
	Sum:			243.332	54	160	65		157.621	54	160	92		177.374	54	160	95	
	Mean:							1.23					1.62					1.68

Table 8
Summary of network parameters for SOP blocks with $k = 5$ and selected probability assignment strategies

	Benchmark			$P_{[0.5]}$					$P_{[0.1\text{-}0.9]}$					$P_{[0.9\text{-}0.1]}$				
	Name	In	Out	SW	ξ_f^p	ξ_f^c	ξ_f^e	μ_e	SW	ξ_f^p	ξ_f^c	ξ_f^e	μ_e	SW	ξ_f^p	ξ_f^c	ξ_f^e	μ_e
1	5xp1	7	10	21.798	2	5	3	1.50	15.918	2	5	4	2.00	14.790	2	5	4	2.00
2	bw_21_11_9	5	3	4.704	2	2	2	1.00	3.595	2	2	2	1.00	3.672	2	2	2	1.00
3	bw_27_26_25	5	3	4.249	1	1	1	1.00	2.901	1	1	1	1.00	3.944	1	1	1	1.00
4	bw_321	5	3	5.041	1	1	1	1.00	4.466	1	1	1	1.00	3.714	1	1	1	1.00
5	bw_432	5	3	4.100	1	1	1	1.00	3.918	1	1	1	1.00	2.638	1	1	1	1.00
6	clip_432	10	3	17.744	3	11	4	1.33	12.284	3	11	6	2.00	14.577	3	11	5	1.67
7	ldd_mno	9	3	4.413	2	2	2	1.00	8.381	2	2	2	1.00	0.782	2	2	2	1.00
8	misex1_3456	8	4	5.012	2	2	2	1.00	4.697	2	2	2	1.00	1.935	2	2	2	1.00
9	misex1_356	7	3	4.278	2	2	2	1.00	4.377	2	2	2	1.00	1.547	2	2	2	1.00
10	misex1_3567	7	4	5.404	2	2	2	1.00	5.072	2	2	2	1.00	2.223	2	2	2	1.00
11	misex1_367	7	3	3.917	2	2	2	1.00	3.382	2	2	2	1.00	1.604	2	2	2	1.00
12	misex1_4567	8	4	5.162	2	2	2	1.00	4.513	2	2	2	1.00	2.220	2	2	2	1.00
13	misex1_567	8	3	4.428	2	2	2	1.00	4.143	2	2	2	1.00	1.796	2	2	2	1.00
14	rd53	5	3	7.023	2	4	2	1.00	4.549	2	4	4	2.00	4.625	2	4	4	2.00
15	rd73	7	3	20.104	3	16	3	1.00	12.804	3	16	6	2.00	13.458	3	16	6	2.00
16	sao2_210	10	3	10.910	2	6	3	1.50	3.643	2	6	4	2.00	10.651	2	6	4	2.00
17	sao2_310	10	3	10.890	2	6	3	1.50	3.647	2	6	4	2.00	10.588	2	6	4	2.00
18	sao2_320	10	3	4.057	2	5	2	1.00	1.065	2	5	3	1.50	4.315	2	5	4	2.00
19	sao2_321	10	3	7.266	2	6	3	1.50	2.619	2	6	4	2.00	6.828	2	6	4	2.00
20	sao2	10	3	11.041	2	6	3	1.50	3.658	2	6	4	2.00	10.794	2	6	4	2.00
21	sqn	7	3	9.165	2	5	3	1.50	3.627	2	5	4	2.00	9.867	2	5	4	2.00
	Sum:			170.706	41	89	48		113.259	41	89	62		126.568	41	89	62	
	Mean:							1.16					1.45					1.46

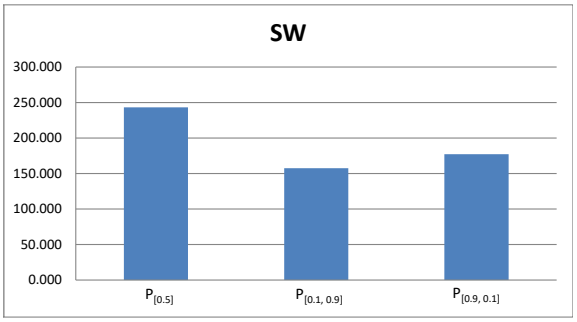
Table 9
Summary of network parameters for SOP blocks with $k = 7$ and selected probability assignment strategies

	Benchmark			$P_{[0.5]}$					$P_{[0.1\text{-}0.9]}$					$P_{[0.9\text{-}0.1]}$				
	Name	In	Out	SW	ξ_f^p	ξ_f^c	ξ_f^e	μ_e	SW	ξ_f^p	ξ_f^c	ξ_f^e	μ_e	SW	ξ_f^p	ξ_f^c	ξ_f^e	μ_e
1	5xp1	7	10	18.883	2	3	3	1.50	14.389	2	3	3	1.50	13.461	2	3	3	1.50
2	bw_21_11_9	5	3	4.278	1	1	1	1.00	3.420	1	1	1	1.00	3.579	1	1	1	1.00
3	bw_27_26_25	5	3	4.249	1	1	1	1.00	2.901	1	1	1	1.00	3.944	1	1	1	1.00
4	bw_321	5	3	5.041	1	1	1	1.00	4.466	1	1	1	1.00	3.714	1	1	1	1.00
5	bw_432	5	3	4.100	1	1	1	1.00	3.918	1	1	1	1.00	2.638	1	1	1	1.00
6	clip_432	10	3	14.654	2	4	3	1.50	10.168	2	4	4	2.00	12.039	2	4	4	2.00
7	ldd_mno	9	3	3.620	1	1	1	1.00	6.332	1	1	1	1.00	0.763	1	1	1	1.00
8	misex1_3456	8	4	4.587	1	1	1	1.00	4.557	1	1	1	1.00	1.909	1	1	1	1.00
9	misex1_356	7	3	3.853	1	1	1	1.00	4.225	1	1	1	1.00	1.530	1	1	1	1.00
10	misex1_3567	7	4	4.979	1	1	1	1.00	4.920	1	1	1	1.00	2.206	1	1	1	1.00
11	misex1_367	7	3	3.492	1	1	1	1.00	3.230	1	1	1	1.00	1.587	1	1	1	1.00
12	misex1_4567	8	4	4.737	1	1	1	1.00	4.373	1	1	1	1.00	2.194	1	1	1	1.00
13	misex1_567	8	3	4.003	1	1	1	1.00	4.003	1	1	1	1.00	1.770	1	1	1	1.00
14	rd53	5	3	6.079	2	3	2	1.00	4.058	2	3	3	1.50	4.095	2	3	3	1.50
15	rd73	7	3	16.520	3	11	3	1.00	10.518	3	11	5	1.67	11.043	3	11	5	1.67
16	sao2_210	10	3	9.304	2	4	2	1.00	3.015	2	4	4	2.00	8.970	2	4	3	1.50
17	sao2_310	10	3	9.273	2	4	2	1.00	3.018	2	4	3	1.50	8.925	2	4	3	1.50
18	sao2_320	10	3	3.625	2	4	2	1.00	0.896	2	4	4	2.00	3.577	2	4	3	1.50
19	sao2_321	10	3	6.034	2	4	2	1.00	2.155	2	4	4	2.00	5.837	2	4	3	1.50
20	sao2	10	3	9.412	2	4	2	1.00	3.028	2	4	4	2.00	9.103	2	4	3	1.50
21	sqn	7	3	7.569	2	3	3	1.50	3.070	2	3	3	1.50	8.125	2	3	3	1.50
	Sum:			148.292	32	55	35		100.660	32	55	48		111.009	32	55	44	
	Mean:							1.07					1.37					1.27

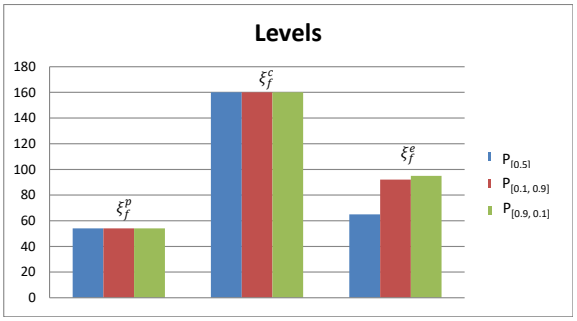
Tables 7–9, in the first columns, contain information about individual benchmarks (name, number of inputs, number of outputs). The following columns group the switching activity values and parameters regarding the number of levels of the structures created for three probability assignment strategies ($P_{[0.5]}$, $P_{[0.1-0.9]}$, $P_{[0.9-0.1]}$). The parameters describing the number of levels include: the number of levels for the classical parallel structure (ξ_f^p), the classical cascade structure (ξ_f^c), the structure created according to the proposed algorithm (ξ_f^e), and the parameter μ_e. The last rows of each table contain the sum or average value of the individual parameters.

The individual sums or average values for Tables 7, 8 and 9 are presented as graphs in Figs. 8–10.

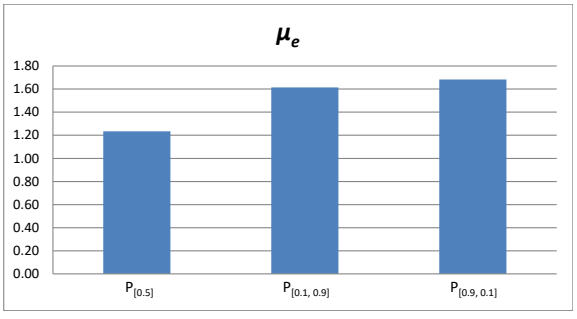
Analyzing the graphs in Figs. 8a, 9a, and 10a, it is evident that, regardless of the number of products, the worst results are obtained for the probability distribution $P_{[0.5]}$. This confirms



(a)



(b)



(c)

Fig. 8. Obtained results of the SOP network mapping in 3-product blocks: total *SW* value (a), number of logical levels for the classic parallel method, cascade method, and the proposed solution (b), average value of the μ_e coefficient (c)

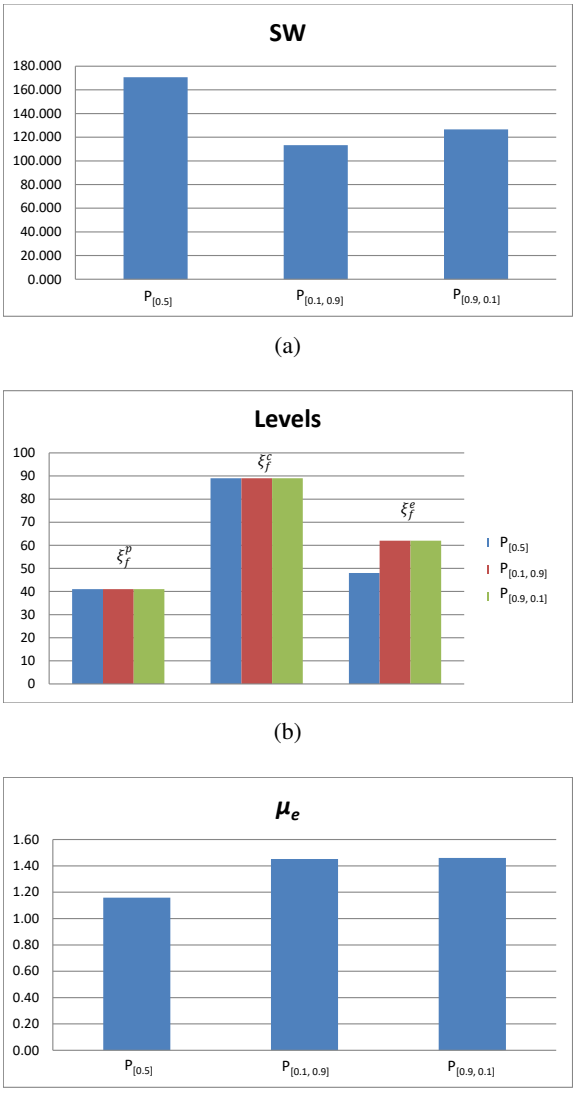


Fig. 9. Obtained results of the SOP network mapping in 5-product blocks: total *SW* value (a), number of logical levels for the classic parallel method, cascade method, and the proposed solution (b), average value of the μ_e coefficient (c)

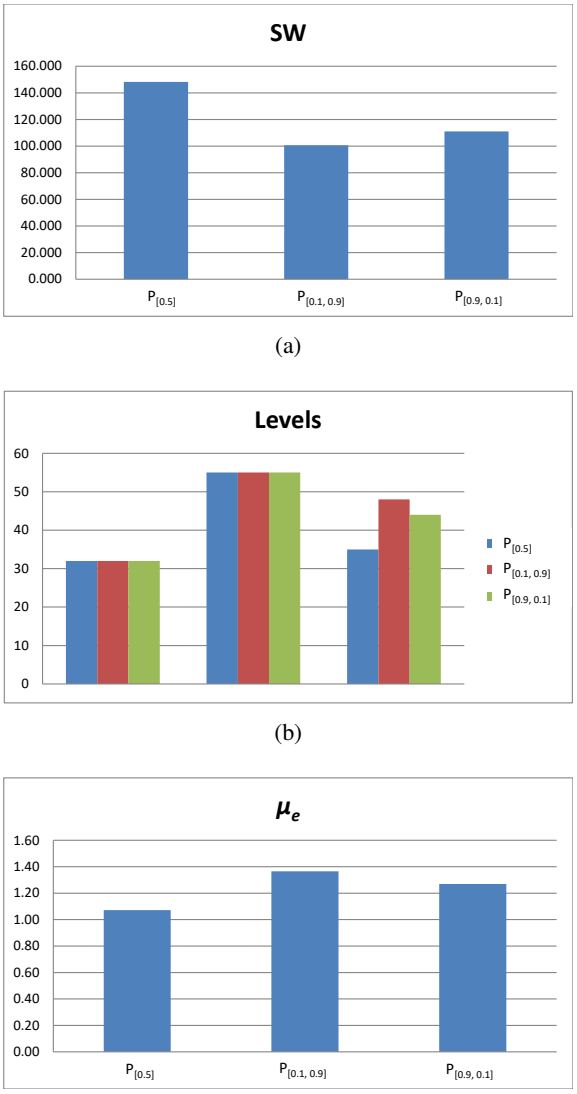


Fig. 10. Obtained results of the SOP network mapping in 7-product blocks: total *SW* value (a), number of logical levels for the classic parallel method, cascade method and the proposed solution (b), average value of the μ_e coefficient (c)

the original thesis, which is a key element of the developed algorithm, that input signal probability values closer to 0.5 cause more frequent switching of the entire network, which undoubtedly leads to higher values of the summary *SW* coefficient. In each case, the application of the proposed algorithm led to a reduction in the summary value of the *SW* coefficient for the probability distributions consistent with $P_{[0.1-0.9]}$, $P_{[0.9-0.1]}$.

For Figs. 8b, 9b, and 10b representing the number of levels, it can be seen that for the $P_{[0.5]}$ distribution, the number of levels of the proposed approach (ξ_f^e) is closer to the solutions obtained for the classical parallel method (ξ_f^p) (i.e., the optimal solution).

In the case of the $P_{[0.1,0.9]}$ and $P_{[0.9,0.1]}$ distributions, the obtained number of levels for the proposed algorithm (ξ_f^e) leads to solutions intermediate between the solutions for the classical parallel method (ξ_f^p) and the classical cascade method (ξ_f^c). It is difficult to indicate the advantage of one distribution over the other. However, the obtained minimization of dynamic power is associated with the need to increase the number of levels.

Figures 8c, 9c and 10c show that for the $P_{[0.5]}$ distribution the mean value of the μ_e coefficient is always the smallest, so this distribution leads to the best solutions in terms of dynamic properties.

After confirming the influence of the input signal probability distribution on the *SW*, it is necessary to evaluate the effectiveness of the proposed algorithm in reducing the *SW* value. For this purpose, Table 10 compares the summed *SW* values (the last row of Tables 7, 8, and 9) with the results for other approaches. The following notations are introduced:

- Proposed approach – function mapping using the presented method.

- Cl_cascade – function mapping using the classical cascade method (immediately after minimization – without sorting the implicants).
- Cl_cascade_sort – function mapping using the classical cascade method with sorting the implicants.
- CL_parallel – function mapping using the classical parallel method (immediately after minimization – without sorting the implicants).
- Cl_parallel_sort – function mapping using the classical parallel method with sorting the implicants.

Table 10 also includes the number of SOP block products (k) and the input probability distribution.

Table 10
Synthetic comparison of the proposed solution with alternative solutions for different SOP sizes in terms of SW

		SUM of SW				
p(xi)	k	proposed approach	Cl_cascade sort	Cl_cascade	Cl_parallel sort	Cl_parallel
$P_{[0.5]}$	3	243.332	306.503	319.101	251.471	255.272
	5	170.706	192.605	197.988	181.864	183.552
	7	148.292	157.959	160.473	154.354	155.397
$P_{[0.1, 0.9]}$	3	157.621	238.952	253.789	192.141	195.703
	5	113.259	149.986	158.623	142.132	143.372
	7	100.66	121.980	125.299	116.843	117.631
$P_{[0.9, 0.1]}$	3	177.374	277.198	285.202	215.644	217.415
	5	126.568	172.868	174.539	154.112	155.352
	7	111.009	140.356	143.376	132.967	134.968
Total SW:		1348.821	1758.407	1818.390	1541.528	1558.662

The last row of Table 10 shows the sum of the *SW* coefficient values for each method (regardless of the k value and the probability distribution). These values are presented in the form of a graph in Fig. 11.

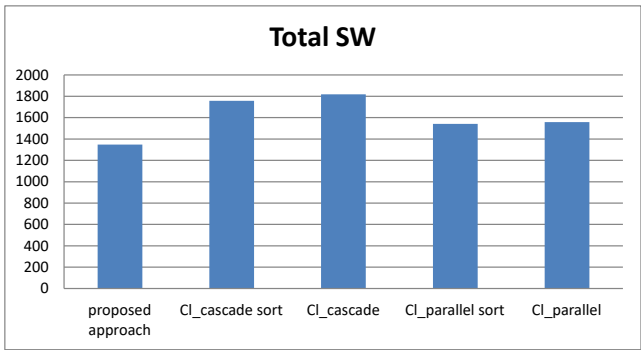


Fig. 11. Comparison of SW for different mapping strategies in SOP networks

Looking at the graph in Fig. 11, it is clear that the proposed solution is the most efficient in terms of dynamic power (minimum *SW* value). Significantly worse solutions were obtained for the classical parallel method, and the worst for the classical cascade method. It is also clear that this tendency is independent of the sorting of input variables. Pre-sorting the implicants leads to a slight reduction in dynamic power, which, however, does not change the overall energy-saving characteristics of the mapping method.

8. CONCLUSIONS

The experimental results clearly indicate that the proposed method of technology mapping logic functions in the form of an SOP network leads to a significant reduction in switching activity (SW) compared to other, commonly used classical methods. Importantly, the improvement achieved in energy efficiency does not come at the cost of significantly degrading the network dynamic properties. The results obtained for the developed mapping algorithm are significantly better dynamically than the classical cascade method and are often comparable to those achieved using the dynamically optimal classical parallel method.

An additional advantage of the proposed method is its algorithmic simplicity. It does not require complex optimization procedures or a significant increase in computational cost, making it relatively easy to implement into existing logic function mapping algorithms in CAD-based synthesis support systems. This makes it attractive not only from a theoretical perspective but also from a practical perspective, especially in the context of designing low-power systems.

It should be noted, however, that other approaches are also known to reduce switching activity in SOP networks through direct minimization, such as methods based on output graphs. Integrating these methods with the observations and conclusions presented in this paper creates a realistic possibility of obtaining near-optimal SOP network solutions. The authors see significant development potential in this area of further research and plan to work in the near future on combining both approaches within a coherent process of optimizing multi-output functions described by output graphs.

REFERENCES

[1] T. Joshi, S. Chakraborty, S. Ambuskar, and A.K. Singh, "Automated Formal Verification Methodology for Digital Circuits," in *Proc. 14th Int. Conf. Inf. Commun. Technol. Syst.* (ICTS), Surabaya, Indonesia, Oct. 2023, pp. 277–282, doi: 10.1109/icts58770.2023.10330830.

[2] B. Wyrwoł and E. Hrynkiewicz, "Implementation of a microcontroller-based simplified FITA-FIS model," *Microprocessors Microsyst.*, vol. 44, pp. 22–27, Jul. 2016, doi: 10.1016/j.micpro.2015.11.012.

[3] Y. Sasaki, "A Survey on IoT Big Data Analytic Systems: Current and Future," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1024–1036, Jan. 2022.

[4] J.S. Yalli, M.H. Hasan, and A.A. Badawi, "Internet of Things (IoT): Origin, Embedded Technologies, Smart Applications and its Growth in the Last Decade," *IEEE Access*, vol. 12, pp. 91357–91382, 2024.

[5] F. Baraati, A. Amirany, M.T. Nasab, K. Jafari, R. Ghaderi, "A novel approach for designing high-accuracy approximate signed multipliers," *Design+*, vol. 1, no. 1, p. 3882, 2024, doi: 10.36922/dp.3882.

[6] M. Bahman Abadi, A. Amirany, and M.H. Moaiyeri, "Non-volatile Quaternary Pre-Charged MRAM Cell Design based on Emerging Technologies," *32nd International Conference on*

Electrical Engineering (ICEE), Tehran, Iran, 2024, pp. 1–5, doi: 10.1109/ICEE63041.2024.10668224.

[7] M. Bahman Abadi, A. Amirany, M.H. Moaiyeri, and K. Jafari, "Towards Nonvolatile Spintronic Quaternary Flip-Flop and Register Design," *SPIN*, vol. 13, no. 3, p. 2350015, 2023, doi: 10.1142/S2010324723500157.

[8] A. Amirany, K. Jafari, and M.H. Moaiyeri, "An Investigation of Hardware Implementation of Multi-Valued Logic Using Different Nanodevices," *31st International Conference on Electrical Engineering* (ICEE), Tehran, Iran, 2023, pp. 426–431, doi: 10.1109/ICEE59167.2023.10334810.

[9] M. Rezaei, A. Amirany, M.H. Moaiyeri, and K. Jafari, "A High-Accuracy and Low-Power Emerging Technology-Based Associative Memory," *IEEE Trans Nanotechnol.*, vol. 23, pp. 293–298, 2024, doi: 10.1109/TNANO.2024.3380368.

[10] H. Ali and B.M. Al-Hashimi, "Architecture Level Power-Performance Tradeoffs for Pipelined Designs," in *Proc. IEEE Int. Symp. Circuits Syst.* (ISCAS), New Orleans, USA, 2007, pp. 1791–1794.

[11] S. Bard and N.I. Rafla, "Reducing power consumption in FPGAs by pipelining," in *Proc. 51st Midwest Symp. Circuits Syst.* (MWSCAS), Knoxville, USA, 2008, pp. 173–176.

[12] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," in *Proc. 27th Int. Symp. Comput. Archit.* (ISCA), Vancouver, Canada, 2000, pp. 83–94.

[13] M. Kuc, W. Sułek, and D. Kania, "Low power QC-LDPC decoder based on Token Ring architecture," *Energies*, vol. 13, no. 23, p. 6310, Dec. 2020.

[14] M. Kuc, W. Sułek, and D. Kania, "FPGA-oriented LDPC decoder for Cyber-Physical System," *Mathematics*, vol. 8, no. 5, p. 723, May 2020.

[15] R. Bondade, "Hardware-software co-design of an embedded power management module with adaptive on-chip power processing schemes," in *Proc. IEEE Int. Symp. Circuits Syst.* (ISCAS), Paris, France, 2010, pp. 617–620.

[16] P.A. Hsiung and C.W. Liu, "Exploiting Hardware and Software Low power Techniques for Energy Efficient Co-Scheduling in Dynamically Reconfigurable Systems," in *Proc. Int. Conf. Field Programmable Logic Appl.* (FPL), 2007, pp. 165–170.

[17] O.S. Unsal and I. Koren, "System-level power-aware design techniques in real-time systems," *Proc. IEEE*, vol. 91, no. 7, pp. 1055–1069, Jul. 2003.

[18] A. Raghunathan, N.K. Jha, and S. Dey, High-Level Power Analysis and Optimization. Boston, MA, USA: Springer, 1998.

[19] L. Benini and G. De Micheli, "System-Level Power Optimization: Techniques and Tools," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 5, no. 2, pp. 115–192, Apr. 2000.

[20] Y. Shin, J. Seomun, K. Choi, and T. Sakurai, "Power gating: Circuits, design methodologies, and best practice for standard-cell VLSI designs," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 15, no. 4, pp. 1–37, 2010.

[21] N. Wang *et al.*, "Power-gating-aware scheduling with effective hardware resources optimization," *Integr. VLSI J.*, vol. 61, pp. 167–177, 2018.

[22] C. Chen, A. Srivastava, and M. Sarrafzadeh, "On gate level power optimization using dual supply voltages," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 9, no. 5, pp. 616–629, Oct. 2001.

[23] J.C. Chi, H.H. Lee, S.H. Tsai, and M.C. Chi, "Gate Level Multiple Supply Voltage Assignment Algorithm for Power Optimization Under Timing Constraint," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 6, pp. 637–648, Jun. 2007, doi: 10.1109/TVLSI.2007.898650.

[24] A. Manzak and C. Chakrabarti, "A low power scheduling scheme with resources operating at multiple voltages," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 10, no. 1, pp. 6–14, Feb. 2002.

[25] R. Mukherjee and S.O. Memik, "Evaluation of Dual VDD Fabrics for Low Power FPGAs," in *Proc. Asia South Pacific Des. Autom. Conf.*, vol. 2, 2005, pp. 1240–1243.

[26] M.R. Alam, M.E.S. Nasab, and S.M. Fakhraie, "Power Efficient High-Level Synthesis by Centralized and Fine-Grained Clock Gating," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 34, no. 12, pp. 1954–1963, Dec. 2015, doi: 10.1109/TCAD.2015.2445734.

[27] E. Bezati, S. Casale-Brunet, M. Mattavelli, and J.W. Janneck, "Clock-Gating of Streaming Applications for Energy Efficient Implementations on FPGAs," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 36, no. 4, pp. 699–703, Apr. 2017, doi: 10.1109/TCAD.2016.2597215.

[28] R. Mullins and S. Moore, "Demystifying Data-Driven and Pausible Clocking Schemes," in *Proc. 13th IEEE Int. Symp. Asynchronous Circuits Syst.* (ASYNC), 2007, pp. 175–185.

[29] Y.S. Park, Y. Tao, and Z. Zhang, "A Fully Parallel Nonbinary LDPC Decoder With Fine-Grained Dynamic Clock Gating," *IEEE J. Solid-State Circuits*, vol. 50, no. 2, pp. 464–475, Feb. 2015, doi: 10.1109/JSSC.2014.2362854.

[30] F. Xin, M. Krstić, and E. Grass, "Improvements of Pausible Clocking Scheme for High-Throughput and High-Reliability GALS Systems Design," in *Advanced Techniques in Logic Synthesis, Optimizations and Applications*, New York, NY, USA: Springer, 2011, pp. 401–417.

[31] R.S. Shelar, H. Narayanan, and M.P. Desai, "Orthogonal partitioning and gated clock architecture for low power realization of FSMs," in *Proc. 13th Annu. IEEE Int. ASIC/SOC Conf.*, 2000, pp. 266–270.

[32] W. Shen, Y. Cai, X. Hong, and J. Hu, "Activity-Aware Registers Placement for Low Power Gated Clock Tree Construction," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, 2007, pp. 383–388.

[33] E. Amini, M. Najibi, and H. Pedram, "A Novel Clock Generation Scheme for Globally Asynchronous Locally Synchronous Systems: An FPGA-Validated Approach," in *Proc. 15th Great Lakes Symp. VLSI* (GLSVLSI), 2005, pp. 296–301.

[34] E. Amini, M. Najibi, and H. Pedram, "Globally Asynchronous Locally Synchronous Wrapper Circuit Based on Clock Gating," in *Proc. IEEE Comput. Soc. Annu. Symp. Emerging VLSI Technol. Archit.*, 2006, pp. 193–199.

[35] M. Krstic, E. Grass, F.K. Gurkaynak, and P. Vivet, "Globally Asynchronous, Locally Synchronous Circuits: Overview and Outlook," *IEEE Des. Test Comput.*, vol. 24, no. 5, pp. 430–441, Sep. 2007.

[36] A. Kulmala, T.D. Hamalainen, and M. Hannikainen, "Reliable GALS Implementation of MPEG-4 Encoder with Mixed Clock FIFO on Standard FPGA," in *Proc. Int. Conf. Field Programmable Logic Appl.* (FPL), 2006, pp. 1–6.

[37] Z. Yu and B.M. Baas, "High Performance, Energy Efficiency, and Scalability With GALS Chip Multiprocessors," *IEEE Trans.*

Very Large Scale Integr. (VLSI) Syst., vol. 17, no. 1, pp. 66–79, Jan. 2009, doi: 10.1109/TVLSI.2008.2001947.

[38] S. Chattopadhyay, P. Yadav, and R. Singh, "Multiplexer Targeted Finite State Machine Encoding for Area and Power Minimization," in *Proc. IEEE INDICON*, 2004, pp. 12–16.

[39] K. Kajstura and D. Kania, "A decompositional method of encoding the internal states of a finite state machine aimed at minimizing power" *Prz. Elektrotechn.*, vol. 87, no. 6, pp. 146–150, 2011.

[40] K. Kajstura and D. Kania, "Binary Tree-based Low Power State Assignment Algorithm," in *Proc. 12th Int. Conf. Comput. Methods Sci. Eng.* (ICCMSE), 2016, p. 030007.

[41] K. Kajstura and D. Kania, "Low Power Synthesis of Finite State Machines State Assignment Decomposition Algorithm," *J. Circuits Syst. Comput.*, vol. 27, no. 3, p. 1850041, 2018.

[42] L. Mengibar, L. Entrena, M.G. Lorenz, and E.S. Millan, "Partitioned state encoding for low power in FPGAs," *Electron. Lett.*, vol. 41, no. 17, pp. 948–949, 2005.

[43] Y. Xia and A.E.A. Almaini, "Genetic algorithm based state assignment for power and area optimisation," *IEE Proc. – Comput. Digit. Tech.*, vol. 149, no. 4, pp. 128–133, Jul. 2002.

[44] Y. Aghaghiri, F. Fallah, and M. Pedram, "Transition reduction in memory buses using sector-based encoding techniques," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 23, no. 8, pp. 1164–1174, Aug. 2004.

[45] L. Benini *et al.*, "Synthesis of power-managed sequential components based on computational kernel extraction," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 20, no. 9, pp. 1118–1131, Sep. 2001.

[46] A. Barkalov, L. Titarenko, and S. Chmielewski, "Improving characteristics of LUT-based Moore FSMs," *IEEE Access*, vol. 8, pp. 155306–155318, 2020.

[47] A. Barkalov, L. Titarenko, and K. Mielcarek, "Improving characteristic of LUT-based Mealy FSMs," *Int. J. Appl. Math. Comput. Sci.*, vol. 30, no. 4, pp. 745–759, 2020.

[48] D. Kania, "Logic decomposition for PAL-based CPLDs," *J. Circuits Syst. Comput.*, vol. 24, no. 3, pp. 1–27, 2015.

[49] M. Kubica, A. Opara, and D. Kania, "Logic Synthesis Strategy Oriented to Low Power Optimization," *Appl. Sci.*, vol. 11, no. 19, p. 8797, 2021.

[50] A. Opara, M. Kubica, and D. Kania, "Decomposition Approaches for Power Reduction," *IEEE Access*, vol. 11, pp. 29417–29429, 2023.

[51] A. Opara, M. Kubica, "Technology mapping of multi–output functions leading to the reduction of dynamic power consumption in FPGAs," *Int. J. Appl. Math. Comput. Sci.*, vol. 33, no. 2, pp. 267–284, 2023.

[52] D. Kania, "An Efficient Approach to Synthesis of Multi-Output Boolean Functions on PAL-based Devices," *IEE Proc. – Comput. Digit. Tech.*, vol. 150, no. 3, pp. 143–149, May 2003.

[53] M. Kubica and D. Kania, "Graph of Outputs in the Process of Synthesis Directed at CPLDs," *Mathematics*, vol. 7, no. 2, p. 177, 2019.

[54] B. Wyrwoł, M. Kubica, and D. Kania, "SOP network optimization methods aimed at reducing switching activity," *IEEE Access*, vol. 13, pp. 179438–179448, 2025.

[55] M. Kubica and D. Kania, "Multi-level sum of product (SOP) network power optimization based on switching graph," *Electronics*, vol. 13, no. 20, p. 4011, Oct. 2024, doi: 10.3390/electronics13204011.

[56] M. Kubica, B. Pochopien, and D. Kania, "Switching activity reduction of SOP networks," *IEEE Access*, vol. 12, pp. 112984–112994, 2024, doi: 10.1109/ACCESS.2024.3442975.

[57] D. Kania, "A technology mapping algorithm for PAL-based devices using multi-output function graphs," in *Proc. 26th Euromicro Conf.*, Maastricht, Netherlands, 2000, pp. 146–153.

[58] D. Kania, "A new approach to logic synthesis of multi-output Boolean functions on PAL-based CPLDs," in *Proc. ACM Great Lakes Symp. VLSI* (GLSVLSI'07), Stresa, Italy, Mar. 11–13, 2007, pp. 152–155.

[59] Collaborative Benchmarking and Experimental Algorithmics Laboratory, "LGSynth93 Benchmarks," cbl.ncsu.edu. [Online]. Available: https://www.cbl.ncsu.edu:16080/benchmarks/LGSynth93/testcase/ (accessed Oct. 24, 2023).