IEEE Open Journal of the Computer Society

Received 29 January 2023; accepted 14 February 2023. Date of publication 6 March 2023;
date of current version 9 March 2023. The review of this article was arranged by Associate Editor Peng Li.

Digital Object Identifier 10.1109/OJCS.2023.3247752

Towards Area Efficient Logic Circuit: Exploring Potential of Reconfigurable Gate by Generic Exact Synthesis

LIUTING SHANG [1] (Student Member, IEEE), AZAD NAEEMI [2] (Senior Member, IEEE), AND CHENYUN PAN [1] (Senior Member, IEEE)

[1]Department of Electrical Engineering, University of Texas at Arlington, Arlington, TX 76019 USA
[2]School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA

CORRESPONDING AUTHOR: LIUTING SHANG (e-mail: liuting.shang@mavs.uta.edu).

This work was supported in part by the Advanced Scientific Computing Research Program of the Department of Energy under Award DE-SC0022881 and in part by the National Science Foundation under Grant CCF-2219753.

ABSTRACT In this article, we propose a generic design methodology to achieve area-efficient reconfigurable logic circuits by using exact synthesis based on Boolean satisfiability (SAT) solver. The proposed methodology better leverages the high representation ability of emerging reconfigurable logic gates (RLGs) to achieve reconfigurable circuits with fewer gates. In addition, we propose a fence-based acceleration method to provide $>10\times$ speed up for the synthesis without an observable loss of optimality. Furthermore, four sets of RLGs are developed based on a recently proposed valley-spin device as a case study to demonstrate the advantage of the proposed circuit. Simulations have been performed to analyze the impact of the fence searching algorithm and combination of operators. Based on disjoint-support decomposable (DSD) benchmarks, up to 38% and 73% reductions are observed in the area and energy-delay-area product (EDAP), respectively, compared to CMOS counterparts. Compared to the two existing synthesis methods, the proposed scheme provides 40% and 26.3% reduction in EDAP based on MCNC benchmark.

INDEX TERMS Reconfigurable logic circuit, SAT solver, exact synthesis, valley-spin logic, area optimization.

I. INTRODUCTION

Decades of successful CMOS technology scaling have brought exponential growth in transistor density and hardware performance [1]. As geometrical scaling becomes more challenging, several technical routes have been proposed to sustain the historical growth in computation power [2], [3]. In this context, the reconfigurable logic gate (RLG) is one of the promising candidates as it may improve logic circuit area efficiency by increasing the number of functions performed by a fixed number of gates. To achieve this goal, researchers have developed operator-rich RLGs using emerging device technologies, such as quantum, optical, spintronic, ferroelectric, semiconductor, ReRAM, nanotube, and silicon nanowire devices [4], [5], [6], [7], [8], [9], [10], [11], [12]. These RLGs can switch between two or more operators with a small overhead in area and performance. However, most existing research efforts in this area have focused on gate-level design with the assumption that improving circuit area efficiency will motivate their wide adoption. With regard to applications, some hardware security solutions, such as circuit camouflage or obfuscation, also assume that hardware can be reused to realize different functions based on RLGs [13], [14]. Both gate-level design and security applications require area-efficient RLG-based reconfigurable circuits. In addition, with more versatile RLGs being invented, there is also an urgent need for exploring a general method to leverage all kinds of RLGs based on various emerging technologies. It is worth noting that a lookup table (LUT) can be regarded as a special RLG with flexibility that overwhelms conventional RLGs. Although LUT-based Field Programmable Gate Arrays (FPGA) and corresponding synthesis methods are mature reconfigurable solutions, they are only designed

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

for fast and low-cost development instead of area-optimal applications. This type of device and its synthesis methods lead to significant penalties in area and performance due to the fully-reconfigurable interconnection network and the redundant representation ability of the LUT [15], [16]. To achieve area efficiency by leveraging reconfigurable circuits, a reconfigurable circuit scheme with fewer redundant reconfigurability of interconnect/device is needed.

Meanwhile, a systematic design methodology is critical to fully unleash the power of RLG-based circuits for higher area efficiency along with the roadmap of technology scaling [3]. Compared to device research, a limited number of synthesis methods have been proposed to leverage RLGs in logic circuits. Evolutionary synthesis is one of the proposed schemes that intend to automatically design circuits that switch functions (for example, Multiplier/Sorter) according to environmental stimuli, such as temperature. This scheme optimizes a very redundant initialized circuit containing all candidate functions by evolutionary algorithms in terms of the number of gates [17], [18], [19], [20], [21]. Unfortunately, no improvement in terms of the area has been reported compared to the standalone implementation of those functions even in the latest results [21]. Another proposed scheme is technology mapping based on mature and scalable graph methods that aim to leverage RLGs. This scheme is area competitive and uses RLGs as compact logic gates by regarding the RLGs' control (i.e., switch) signals as inputs [9], [22], [23], [24], [25]. Such RLGs are added to the library and deployed in the circuit during logic mapping. This method might not fully leverage the reconfigurability of reconfigurable devices because the RLGs are manually designed. The technology mapping also cannot guarantee the resulted circuit to optimally utilize reconfigurability. Meanwhile, another SAT-based exact synthesis method was proposed to build area-efficient reconfigurable circuits that realize target functions (truth tables). Based on And-Inverter-Graph (AIG), its circuits only involve And and Inverter during the synthesis. This scheme improves the representation ability of the circuits by allowing polymorphic (reconfigurable) edges, i.e., reconfigurable buffers/inverters that are controlled by one of the function inputs [26], [27]. Such a method was then combined with rewriting techniques and applied to large circuits [28], [29]. However, this thread of schemes has two shortcomings. First, different device technologies support different operators of RLGs, while this method cannot customize available operators. Besides, it limits the types of RLGs and cannot fully take advantage of the rich reconfigurability of emerging device technologies. Although logic synthesis and technology mapping are usually separated in modern synthesis flow, the synthesis for reconfigurable circuits should take the technology into account. Nevertheless, SAT-based exact synthesis is still one of the most promising logic synthesis approaches because of its ability to find the area-optimal circuit with a customizable synthesis target [26], [27], [28], [29], [30], [31]. Another great advantage of SAT-based exact synthesis is that users can flexibly describe the desired logic circuit by adding customized constraints.

The advantage of reconfigurable circuits highly depends on the characteristics of RLGs. For example, memristor-based reconfigurable logic gate families are proposed as low-cost and complete RLGs [4], [32], [33], [34]. However, they have complex and multi-gate control mechanisms. Each logic gate needs to be driven by dedicated control signal flow, which induces significant delay and energy penalties. Other RLGs can achieve a variety of cascadable logic operators; however, they generally have limited reconfigurability and/or require changing the input/output or control terminals [5], [6], [7], [8], [9], [10], [11], [12], [35], [36]. Therefore, it is desirable to develop an RLG set that can achieve a wide range of logic operators with a minimum alternation of the input/output/control terminals.

In this article, we propose a generic reconfigurable circuit scheme as well as the corresponding automatic design methodology to better leverage RLGs. The proposed reconfigurable scheme can achieve multiple dedicated functions with a substantial reduction in gate count compared to standalone circuits. An efficient synthesizer for finding an optimum reconfigurable logic network is developed based on an SAT-based exact synthesis library, *percy* [31]. Meanwhile, an accelerated scheme is designed based on an existing fence acceleration method. To demonstrate the effectiveness of the proposed framework, we modify a recently proposed reconfigurable device, the valley-spin (VS) logic device [37], to improve its reconfigurability and use it as a case study. A group of low-cost logic gates is developed to guarantee that any two 2-input operations can be realized in a single gate without switching the input/output or control terminals. Moreover, we perform a comprehensive area/performance benchmarking on the proposed scheme, traditional standalone CMOS-based logic circuit, and other state-of-the-art schemes of leveraging RLGs to demonstrate the benefits of the proposed method. The major contributions of this work are highlighted in the following.

- This work provides a novel reconfigurable scheme that switches circuit functions based on direct control of RLG. The available RLGs in synthesis are arbitrary and customizable for excellent area reduction and compatibility with various emerging device technologies.
- A novel SAT-based design methodology is proposed to optimize the proposed reconfigurable circuit with a significant improvement in area and synthesis time.
- We design complete sets of VS logic gates to achieve all 2-in-1 logic gates without the need of switching the input/output or control terminals.
- We perform a comprehensive benchmarking on the proposed reconfigurable scheme and compare it against non-reconfigurable and existing reconfigurable schemes. Results demonstrate a substantial improvement for the proposed design in the number of gates and EDAP.
- Results from this work offer strong support for the motivation of area efficiency that is widely claimed in RLG research. They also provide valuable insights for device

researchers and technologists to understand the requirements and better design RLGs for maximum circuit performance.

The rest of this article is organized as follows. Section II provides background on SAT solver and SAT-Based exact synthesizer. Section III describes the proposed reconfigurable circuit and synthesis methodology. We discuss the concept of 2-in-1 circuits, design methodology, synthesizer, as well as a scheme of acceleration. Section IV introduces the emerging VS logic device and designs an RLG set with VS technology for the proposed synthesis method. In Section V, we evaluate and analyze the performance of the proposed method based on massive sampyling from the disjoint-support decomposable (DSD) function set. Then, we compare and analyze the proposed method with counterparts by using selected circuits from the MCNC set. Finally, we conclude this article in Section VI.

II. BACKGROUND

A. SAT SOLVER AND SAT-BASED EXACT SYNTHESIZER

SAT is the first proven NP-complete problem, which aims to find if there exists an interpretation that satisfies a given Boolean formula. Exact synthesis is the problem of finding an optimum logic network for a given function. The cost criteria are typically (although not limited to) the size or depth of the network [12], [30], [36], [38], [39]. An SAT-based synthesizer can effectively encode the description of a logical circuit (e.g., target function of the circuit, possible interconnection of nodes, output positions and functions of nodes) into formulas, then find the solution in expression E or declare the non-existence of the solution. Once E is found, the solver finds the logic network that correctly performs the given function. Otherwise, it proves a satisfactory solution does not exist in the search space.

To better illustrate the proposed design methodology, we introduce the following terms:

- **Topology** refers specifically to a directed acyclic graph (DAG), which represents all interconnection information between inputs, gates, and outputs.
- **Fence** is a partition of k gates over l levels, where each level has at least one gate. A fence contains part of the topology information. An example is given in Section II-I-A Fig. 1.
- **Operator** represents the logical relation between the input and output of a gate.
- **Function** represents a complex logic relation between the input and output of a group of gates.

In this article, we adopt the single selection variable (SSV) encoding, a typical method for the synthesis of normal 2-input operator chains [31]. Given an n-input m-output function f f and we assume it can be realized by r gates, the CNF formula F_r consists of the following four variables, for $1 \leq h \leq m,\ n \leq i \leq n+r,$ and $0 < t < 2^n$:

$$x_{it} : t_{th} \text{ bit of } x_i\text{'s truth table} \tag{1}$$

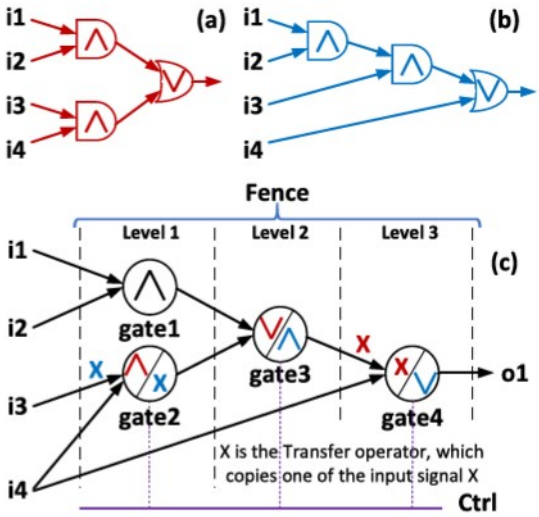


FIGURE 1. Proposed reconfigure scheme with a fixed interconnection network. (a) and (b) show the standalone circuits. (c) represents the reconfigurable logic network.

$$g_{hi} : f_h(x_1, \ldots, x_n) = x_i \tag{2}$$

$$s_{ijk} : x_i = x_j \circ_i x_k \text{ for } 1 \leq j < k < i \tag{3}$$

$$f_{ipq} : p \circ_i q \text{ for } 0 \leq p, q \leq 1, p + q > 0. \tag{4}$$

The g_{hi} variable means the output position of the h^{th} function is gate i. The s_{ijk} variable determines the inputs for each gate i are gates j and k. The f_{ipq} represents a normal Boolean operator of gate i that has p and q as the logical value of its inputs. This Boolean operator could be any 2-variable logical relationship except f_{i00}, because $f_i(0,0) = 0$ according to the definition of normal chains. For a logic circuit, the four variables (1)–(4) describe the expected truth table of each gate, the output position of each function, the topology information, and the operator of each gate, respectively. For example, the description 'gate i has inputs j and k *and* the t^{th} bit of x_i and x_j are a and b, *respectively, and* the t^{th} bit of x_k is c, *then* it must be the case that $b \circ_i c = a$ ' can be represented as the following expression:

$$(s_{ijk} \wedge (x_{it} \oplus \bar{a}) \wedge (x_{jt} \oplus \bar{b}) \wedge (x_{kt} \oplus \bar{c}) \rightarrow (f_{ibc} \oplus \bar{a}) \tag{5}$$

We refer the interested reader to [39] for more details on the encoding method and speed issues. Based on those principles of circuit representation, all possible topologies of a circuit under a given number of gates r can be included in the search space and the area-optimal implementation of different design targets can be found from it. However, when it comes to RLGs-based circuits, the optimality of SAT method highly depends on the way of involving reconfigurability, and a more advanced circuit paradigm improves the upper bound of the performance.

III. PROPOSED DESIGN METHODOLOGY FOR RECONFIGURABLE LOGIC CIRCUITS

In this section, we propose the concept of the function-level reconfigurable logic circuit scheme and develop a generic design methodology. The VS RLG is designed and used as a case study for the functional demonstration.

A. PROPOSED CONCEPT OF RECONFIGURABLE LOGIC CIRCUIT

The proposed methodology aims to design reconfigurable circuits with a common control signal, and the switch of functions mainly relies on gate-level operator reconfiguration. An illustration of the proposed function-level reconfigurable logic circuit is shown in Fig. 1, where two arbitrary 4-input functions are given. Each standalone logic network uses 3 gates with different operators and topologies, as shown in Fig. 1(a) and (b). Fig. 1(c) gives a reconfigurable circuit using the proposed scheme, which only consumes 4 gates. The same circuit performs either function based on the control signal. Note that X (Y) is the 'TRANSFER' operator, which copies the state of the upper (lower) input. For example, to achieve function (a), gate 4 performs the Transfer (X) operator so that its output o1 copies the output of gate 3, and the topology is equivalent to Fig. 1(a); similarly, to achieve function (b), gate 2 performs Transfer (X) operator so that its output copies i3, and the topology is equivalent to Fig. 1(b). This not only provides a mechanism to block the unwanted logic path and reduces RLG usage by allowing minimal local changes in the topology but also enables the circuit to be shared between two functions that have different numbers of inputs. In this brief example of a function-level reconfigurable logic circuit, the number of gates is only increased by 1 compared to the standalone circuit, but the number of realized functions is doubled.

Compared to the baseline synthesis flow of ABC that utilizes reconfigurable gates as part of library [25], the proposed circuit can better leverage the potential of reconfigurable gates. An example of multi-output benchmark 'lion' is provided in Fig. 2 in .blif format, which shows a significant reduction in the number of gates. Compared to the existing synthesizer for reconfigurable circuits [27], the proposed reconfigurable design shown in Fig. 1 is more generic and customizable in the use of RLGs. More details, comprehensive benchmarking, and comparison among the three methods are provided in the rest of the article. Compared to FPGA, the proposed circuit achieves a superior area efficiency due to the fixed interconnection topology (the major area cost in LUT-based circuits) and fewer operators in each gate [40]. Different logic functions can be switched during run-time by switching the common control signals. The simple reconfigurability leads to a significant reduction in the footprint area of the control circuit because if each gate in Fig. 1(c) is realized by a LUT instead of RLG, at least 4 control bits and the corresponding interconnection are required. More detailed performance analyses and trade-offs will be discussed in Section V.

```
# Benchmark Generated by ABC
.model MCNC-Seq_lion
.inputs pi0 pi1 lo0 lo1
.outputs po0 po1 po2
.gate inv1   a=lo0 O=new_n8_
.gate or2_1  a=pi1 b=pi0 O=new_n9_
.gate vsg109 c=pi1 a=lo0 b=lo1 O=new_n10_
.gate vsg109 c=new_n9_ a=new_n10_ b=new_n8_ O=po0
.gate or2_1  a=pi0 b=pi1 O=new_n12_
.gate vsg116 c=pi1 a=lo0 b=lo1 O=new_n13_
.gate vsg109 c=new_n12_ a=new_n13_ b=lo1 O=po1
.gate and2_1 a=pi1 b=lo1 O=new_n15_
.gate vsg109 c=new_n8_ a=new_n15_ b=pi0 O=new_n16_
.gate vsg109 c=new_n9_ a=new_n16_ b=new_n8_ O=po2
.end
# inv1    O=!a
# or2_1 O=(!a+b)
# and2_1 O=(!a*b)
# vsg3_109 O = c*(a^b) + !c*(a*b)
# vsg3_116 O = c*(a^b) + !c*b
```

(a)

```
# Benchmark Generated by the proposed method
.model MCNC-Seq_lion
.inputs pi0 pi1 lo0 lo1
.outputs po0 po1 po2
.gate (X*!Y)/XOR a=pi0 b=lo1 c=pi1 O=newn8_
.gate (X*!Y)/OR a=lo0 b=newn8_ c=pi1 O=new_n9_
.gate XOR a=lo1 b=new_n9_ O=po2
.gate (X+!Y)/(X*!Y) a=lo1 b=new_n9_ c=pi1 O=po1
.gate !X/Y a=lo0 b=po2 c=pi1 O=po0
.end
# XOR: O=a^b
# (X*!Y)/XOR: O=(a*!b)*c + (a^b)*!c
# (X*!Y)/OR: O=(a*!b)*c + (a+b)*!c
# (X+!Y)/(X*!Y): O=(a+!b)*c + (a*!b)*!c
# !X/Y: O=!a*c + b*!c
```

(b)

FIGURE 2. Benchmark 'lion' optimized by (a) ABC and (b) the proposed method. In (b), each reconfigurable gate is controlled by input 'c'.

B. DESIGN METHODOLOGY FOR PROPOSED CIRCUITS

To enable an efficient reconfigurable logic circuit synthesis and benchmark, we propose a design automation framework to generate reconfigurable circuits and analyze their performance.

Currently, the SAT-based exact synthesis mainly processes circuits with a limited number of nodes since further increasing the node number will lead to an explosion of the search space (although techniques like graph cutting, rewriting, and recursive synthesis have enabled exact synthesis to improve large-scale circuit design). To accelerate the synthesis speed, we adopt an existing fence-based method and design the fence-based approach for the proposed reconfigurable circuits [31]. The purpose of introducing the fence is to provide basic structure information and thus shrink the search space. A fence can be expressed as $(n_1, n_2, \ldots n_i, \ldots, l)$, where n_i is the number of gates on the level i, l is the number of levels in a fence. As an example, the fence divided by black dotted lines in Fig. 1(c) can be represented as (2, 1, 1, 3), which means a 3-layer fence with 2, 1, and 1 gate allocated at layers 1, 2, and 3, respectively.

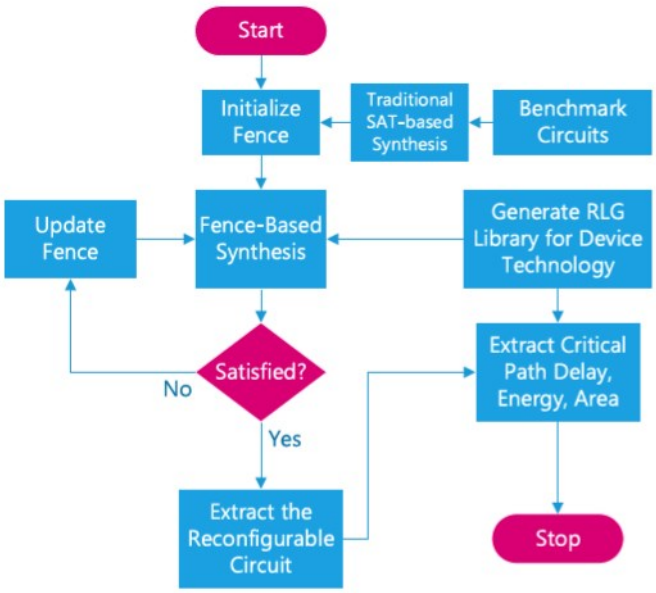


FIGURE 3. Flowchart of the design framework.

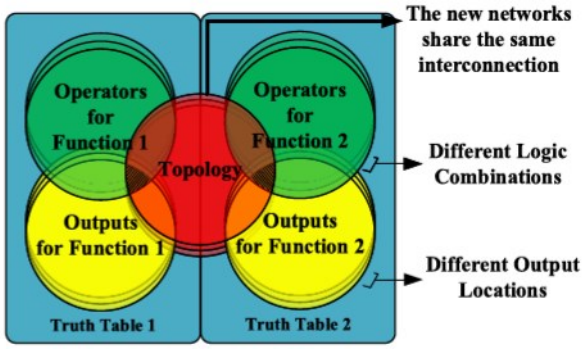


FIGURE 4. Abstract diagram for searching the proposed circuit. Colored circles denote variables.

The overall design framework flowchart is shown in Fig. 3. Here, we briefly describe the overall design flow, and more details for individual modules are described in the following subsections. First, two standalone benchmark circuits are synthesized independently to obtain the initial fence information. Then, the slower fence-based synthesis for the reconfigurable scheme is performed with the operator primitives given by the RLG library based on a given emerging technology. If the initial fence fails to meet a satisfiable state in the SAT solver, the fence is updated to start another synthesis attempt. After the satisfiable expression is found, the logic networks can be extracted, and the performance benchmark can be performed based on the timing, area, and energy information given in the RLG library.

C. SYNTHESIZER FOR RECONFIGURABLE CIRCUIT

SAT-based synthesis is suitable for emerging device/technology exploration because the user can arbitrarily create the search space and manipulate the search rule by adding constraints. However, it is important to find a proper way to describe and translate the synthesis problem into an SAT problem, and an effective translation is critical to determine the area reduction, performance improvement, generality, as well as opportunity for accelerating the synthesis speed. In this case, we describe the synthesis goals as finding two logic networks that (1) perform functions 1 and 2 independently and (2) share identical topology.

First, we perform SSV encoding as the following. The original variables $(x,\ g,\ s,\ f)$ that determine a single logic circuit are replaced with $(x_1,\ x_2,\ g_1,\ g_2,\ s,\ f_1,\ f_2)$, where the subscripts of each variable correspond to different functions. The expression link is changed from $F_r(x,\ g,\ s,\ f)$ to $F_{r1}(x_1, g_1, s, f_1) \wedge F_{r2}(x_2, g_2, s, f_2)$ accordingly. The shared variables s guarantee that the two logic networks have an identical topology. The two independent and complete expressions F_{r1} and F_{r2} with their variables x_1 and x_2 enforce each circuit to perform the desired functions with corresponding operators f_1 and f_2. The synthesizer can enforce all functions have the same output position in the logic network by adding constraints $\wedge_{h=1}^{m} \wedge_{i=n}^{n+r} (\neg g_{1hi} \vee (\vee_{n<j\leq n+r, j\neq i}(\neg g_{2hj})))$, where i and j are the possible output positions of two functions, respectively. The diagram shown in Fig. 4 illustrates the abstractive searching mechanism. Parallel circles denote the potential searching space of variables, which exponentially increases with the scale of the target circuit. The two triangle intersections correspond to two circuit implementations with identical topologies, whose output locations and operators satisfy the truth tables of given functions. The identical topology in the two circuits ensures that they can be implemented by the reconfigurable circuit design shown in Fig. 1. The proposed framework applies to an arbitrary number of functions (x-in-1) for reconfiguration by expanding variables from $(x_1, x_2, g_1, g_2, s, f_1, f_2)$ to $(x_1, x_2, \ldots, x_x, g_1, g_2, \ldots, g_x\ s, f_1, f_2, \ldots, f_x)$ and expression link from $F_{r1}(x_1,\ g_1,\ s,\ f_1) \wedge F_{r2}(x_2,\ g_2,\ s, f_2)$ to $F_{r1}(x_1,\ g_1,\ s,\ f_1) \wedge F_{r2}(x_2,\ g_2,\ s, f_2) \wedge \ldots \wedge F_{rx}(x_x,\ g_x,\ s, f_x)$. In this article, we will discuss the 2-in-1 case.

We develop the encoding method of the reconfigurable scheme based on a mature exact synthesis primitive, *percy*, from an open-source EPFL logic synthesis library [31], which is a header-only exact synthesis tool built on top of *miniSAT* [41]. The remaining design framework and benchmarking are realized in MATLAB.

D. ACCELERATION BY FENCE INITIALIZATION AND SEARCHING

For the SAT-based synthesis, the number of gates determines the topology search space, which grows to a tremendous scale as the gate number increases. To improve the synthesis efficiency, existing work gives a solution by providing topology information (fence) to shrink the search space [31]. Without prior knowledge and the prediction of a certain circuit, the current fence-based synthesis program always starts with one

gate. If the SAT solver cannot find a satisfactory solution after trying all possible fence options under the current gate number, one more gate will be added. This process continues until a satisfactory solution is found. Unfortunately, SAT solving becomes more time-consuming when it is applied to the reconfigurable scheme with multiple logic functions.

In the proposed design flow, we develop a fence initialization and searching method to accelerate the search process. To initialize the fence for synthesizing the reconfigurable circuit, we first independently synthesize the two given functions using the traditional exact synthesis and obtain the fences with minimum gates. The independent synthesis is faster than the synthesis that considers reconfiguration because the latter has more variables as shown in Fig. 4. Meanwhile, the searching space (possible combinations of variable states) grows exponentially instead of linearly with the increase of the number of variables. Second, we combine the two standalone fences to generate an initial fence for the reconfigurable circuit. If we denote the standalone fences as (n_{11}, n_{12}, l_1) and (n_{21}, n_{22}, n_{23}, l_2), the combined/initialized fence will be ($\max(n_{11}, n_{21})$, $\max(n_{12}, n_{22})$, n_{23}, $\max(l_1, l_2)$). n_{ij} is the number of gates of the level j in fence i, l_i is the number of levels in fence i. This method is expected to achieve sub-optimal results based on the fact that 1) the reconfigured fence is typically larger than the standalone fence, and 2) a redundant gate can be assigned a 'TRANSFER/TRANSFER' operator combination, which is removed by the framework after the synthesis. With more variables and clauses involved, each synthesis for the reconfigurable circuit takes more time than the standalone synthesis. A proper initialization can significantly improve the synthesis efficiency thanks to moving partial reconfigurable fence searching to non-reconfigurable fields. For example, over 10× acceleration is observed in synthesizing a circuit with 9 initial gates that achieves two 6-gate standalone circuit functions.

After initializing the fence, we develop two algorithms below to update the fence if the first attempt fails.

Algorithm 1 first tries to add one gate to one of the levels with a priority to the lowest level. If traversing all levels does not find a solution, one gate will be permanently added to a level in a bottom-up sequence. If adding permanent gates to each level cannot provide a solution, one extra level that contains one gate will be added to the top of the fence. The algorithm repeats the process above until success or timeout. This process aims to keep the delay minimum as more gates are added.

The fence searching algorithm determines the efficiency as well as the effectiveness of capturing the minimum circuit topology for the reconfigurable circuit. To find an optimum fence searching algorithm, another promising algorithm candidate is provided below.

Algorithm 2 has the same initializing strategy as Algorithm 1 (i.e., obtaining fence information from the result of standalone synthesis) and also adds one extra node when the current number of nodes cannot satisfy the target circuit. Different from Algorithm 1 which tries to add nodes evenly

Algorithm 1: Fence Searching Algorithm

```
Input  : Initialized fence with r gates and l levels
Output: Successful fence and synthesized reconfigurable circuit
Initialize: Gate index to be added k = r + 1
while True do
    for i ← 1 to l do
        Add Gate k to Level i;
        for j ← i to l do
            Move Gate k to Level j;
            return synthesis result if successful
        end for
        Move Gate k to Level i;
        k + +;
    end for
    l + +;
    Add Gate k to Level l;
    return synthesis result if successful
end while
```

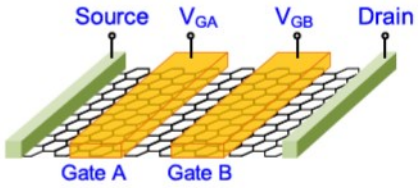


FIGURE 5. VS logic device schematic [37].

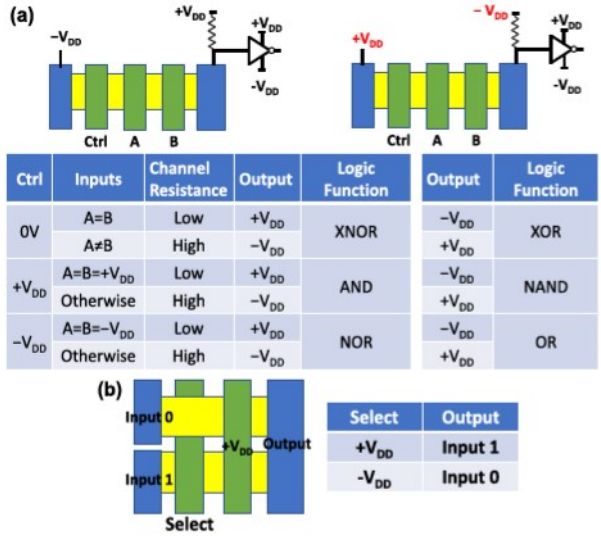
to each layer, Algorithm 2 gives priority to adding nodes to the lower layers based on the fact that many circuits have a cone structure and tend to deploy more nodes at lower layers (the layers closer to the primary inputs). For example, if 6 nodes are to be added to an initialized fence, then the first layer obtains 3 nodes, the second layer obtains 2 nodes, and the third layer obtains 1 node. The fence updating rule can be concluded as: (i) it always adds the node to the lowest possible layer, and (ii) the nodes added to one layer should be equal to or 1 node more than the number of nodes added to its higher neighboring layer. The performance of the proposed two algorithms will be compared with the existing fence searching algorithm in Section V based on the devices and benchmarks introduced as follows.

IV. RECONFIGURABLE VS LOGIC DESIGN

In this section, we propose to use VS devices to create an RLG primitive as well as the 2-in-1 logic sets for any commonly used logic functions.

A. VALLEY-SPIN LOGIC DEVICE

VS logic gates utilize a novel logic device that manipulates the valley-pseudospin degree of freedom [37], [42]. The VS polarization in specific materials can be switched by the applied electric field, which forms a VS polarizer or a VS analyzer. When they are placed in series, a VS valve is built, whose conductance state switches between on or off according to the voltage applied to the gates. As shown in Fig. 5, the VS polarizations controlled by gates A and B form a valve. If V_{GA} has the same polarity as V_{GB}, the valve state is on (high conductance); otherwise, the valve state is off (low conductance). By combining multiple valves, various VS logic gates



Ctrl	Inputs	Channel Resistance	Output	Logic Function	Output	Logic Function
0V	A=B	Low	$+V_{DD}$	XNOR	$-V_{DD}$	XOR
	A≠B	High	$-V_{DD}$		$+V_{DD}$	
$+V_{DD}$	A=B=$+V_{DD}$	Low	$+V_{DD}$	AND	$-V_{DD}$	NAND
	Otherwise	High	$-V_{DD}$		$+V_{DD}$	
$-V_{DD}$	A=B=$-V_{DD}$	Low	$+V_{DD}$	NOR	$-V_{DD}$	OR
	Otherwise	High	$-V_{DD}$		$+V_{DD}$	

Select	Output
$+V_{DD}$	Input 1
$-V_{DD}$	Input 0

FIGURE 6. Schematic and logic function lists of the proposed (a) VS RLG primitive and (b) VS MUX.

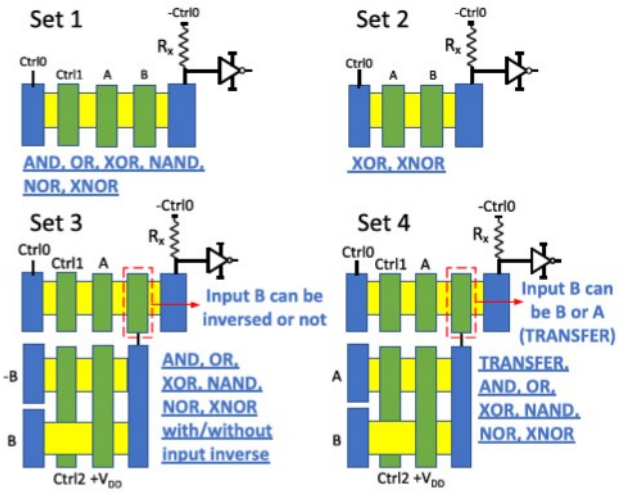


FIGURE 7. Proposed four VS RLG sets to achieve any combinations of two commonly used 2-input logic functions. The underlined texts correspond to the functions that can be realized by each RLG set.

with different available operators can be established. Note that the gate can be placed on either the top or bottom of the 2D channel to create electric fields in opposite directions. The existing work gives two RLGs XNOR/XOR and AND/NAND [37], and we aim to design a set of RLGs to realize arbitrary combinations of two 2-input operators.

B. PROPOSED VALLEY-SPIN LOGIC SETS

The schematic and logic function lists of the proposed VS RLG primitive and VS MUX are shown in Fig. 6. Using the NOR gate in the list as an example, if inputs A and B have the same polarities as Ctrl, i.e., $-V_{DD}$, the valve will be at the 'on' state, and the output of the INVERTER will be $+V_{DD}$. Otherwise, the inconsistent voltage polarity on gates will lead to an 'off' state in the VS valve, and the output of the INVERTER will be $-V_{DD}$ [37], [42]. The RLGs in Fig. 6(a) have excellent reconfigurability because all frequently used operators can be achieved in a single gate by applying different voltages to Ctrl, Supply, and Input terminals. In Fig. 6(b), a low-cost MUX is provided. The states of the two valves are complementary, and only the input with an 'on' state valve will be selected. This component is critical to enabling a low-cost TRANSFER operator.

Based on the VS RLG primitive and MUX, four sets of VS RLG used for the case study are designed and shown in Fig. 7. Since each set is supposed to perform two operations, each set can be easily switched by the binary control signals (associated 'Ctrl0' and 'Ctrl1' for Set1, 'Ctrl0' for Set 2, and 'Ctrl2' for Set 3 and Set 4). If the complementary control gate 'Ctrl0' and '-Ctrl0' are both applied '1' or '0', the output will be constant '0' (FALSE) or '1' (TRUE). Set 1 is the RLG module that performs 6 commonly used logic gates. Set 2 is the simplified Set 1 that achieves XOR and XNOR with a small area. Sets 3 and 4 have the same functions as Set 1, but they include extra low-cost MUXs to achieve the input

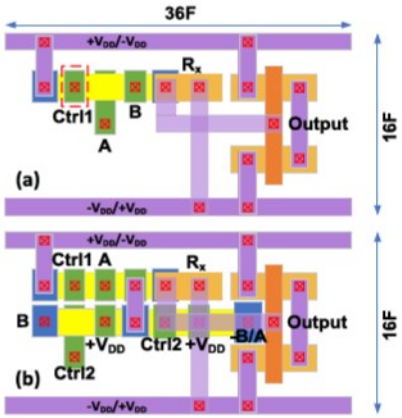


FIGURE 8. Layout for RLGs (a) Set 1 (with the gate in the dashed box) and Set 2 (without gate in the dashed box), and (b) Set 3 and Set 4. Here, F is half of the metal pitch, which is 15 nm.

TABLE 1. Parameters Used for Performance Benchmark

Parameters	Values	Parameters	Values
V_{DD}	0.5V	Node	15nm
R_{von}	7.6kΩ	R_{voff}	10.0MΩ
R_{on}	20kΩ	R_x	30.4-61kΩ
C_{inv}	0.2fF	C_{gate}	0.1fF

inversion or TRANSFER. The Ctrl0 and Ctrl1 in Sets 3 ∼ 4 are fixed because the operator only inverses its input or switches to TRANSFER. All RLGs deployed in the circuit will be controlled by a pair of opposite signals. The layouts for the four RLG sets are illustrated in Fig. 8, which follow the design rules in [43] with a half metal pitch of F.

The parameters used for the performance benchmarking are provided in Table 1. The R_{von} (R_{voff}) is the on (off) resistance

of the VS valve, which is extracted from the channel conductance projected in [42] according to the V_{DD}; the R_{on} is the output resistance of the CMOS INVERTER, and C_{inv} and C_{gate} are the input capacitors of the CMOS INVERTER and VS devices, respectively.

V. SIMULATION, RESULTS, AND DISCUSSIONS

In this section, we will introduce the experimental setup and quantify the impact of the fence searching algorithm and the available types of operators on the synthesis performance. In addition, we will compare the proposed VS reconfigurable logic circuits with normal CMOS circuits, two existing RLG-based circuits, and discuss the simulation results.

A. EXPERIMENTAL SETUP

To evaluate the proposed reconfigurable circuit, we select random function pairs from the following benchmark function suites for simulations:

- FDSD6: 6-input fully disjoint-support decomposable (DSD) functions.
- PDSD5: 5-input partially DSD functions.

DSD benchmarks contain a large number of functions that enable massive sampling. As mentioned in previous sections, exact synthesis applies to circuits on a limited scale, but some technologies, such as rewriting, can be explored to optimize large circuits. Rewriting is a technology to replace small cuts (small single-output function/subcircuit) with area-optimized subcircuit candidates [30], in which the cuts and candidates usually have no more than 6 inputs to avoid the immense growth of the search space. Despite the small size of DSD benchmarks, the simulation results will create a solid foundation for future potential applications in large-scale circuits [44]. Here, we randomly selected 300 pairs of functions from each DSD group for the simulation.

The simulations include three device/circuit configurations:

1) Standalone (STD) CMOS scheme: the given functions are synthesized by using a normal SAT-based exact synthesizer [31], which has been proven to have the ability to find the circuit with a minimal number of 2-input gates. This scheme is used as a baseline to reflect the performance of traditional CMOS devices and circuit schemes.
2) STD VS scheme: use the synthesizer in scheme (1) but replace each logic gate with VS device, i.e., operator primitives in Set 1 and Set 2. Note that the sets in this scheme can realize input inversion by moving the gate to the top or bottom of the channel. This scheme is a counterpart that evaluates whether the main improvement of the proposed VS-based reconfigurable circuit is contributed by the device or circuit.
3) Reconfigurable (REC) VS scheme: use the framework in Fig. 3 to build the proposed reconfigurable logic circuit for each function. All proposed VS RLG sets in Fig. 7 can be utilized.

For the CMOS devices, we assume that n- and p-type devices have identical conductivity, and the XOR gate has a 2× cost compared to NAND/NOR gates with respect to energy, delay, and area. The technology node is 15nm, and the CMOS gate-level performance metrics are adopted from previous work [45]. Under the assumptions for CMOS, VS device, and the VS parameters in Table 1, the gate-level performance metrics are calculated and listed in Table 2. The delay for VS devices is obtained by the Elmore delay model, and the calculated energy for VS devices accounts for both dynamic and static energy components. The dynamic energy equals the average energy consumed per operation and the static energy equals leakage power times delay. The output of VS device is assumed to have 50%/50% probability to be '0'/'1' in the calculation. Note that we assume that power gating technology is applied to the circuit to reduce the leakage during the standby period and thus, the overall power dissipation is more dominated by the dynamic switching power. Because VS gates operate at a much lower voltage, it reduces the switching power substantially and leads to the advantage of the total power compared to the CMOS counterparts. All synthesis simulations in this section are performed on an Intel Core i9-9820X CPU.

TABLE 2. Gate-Level Performance Metrics

CMOS	INV	NOR	NAND	XOR	
Energy (aJ)	25.0	37.5	37.5	75	
Delay (ps)	4.0	6.0	6.0	12.0	
Area (um^2)	0.043	0.072	0.072	0.144	
VS RLG	**Set1**	**Set2**	**Set3**	**Set4**	**MUX**
Energy (aJ)	64.3	87.6	66.0	64.3	25.2
Delay (ps)	14.1	14.1	14.9	14.1	0.76
Area (um^2)	0.129	0.115	0.129	0.129	0.043

B. IMPACT OF OPERATOR TYPE AND FENCE SEARCHING ALGORITHM

In the previous section, we design highly flexible VS RLG sets to efficiently realize a variety of operators, including TRANSFER and MUX. However, emerging technologies rarely support complete operator combinations. Without loss of generality, in this subsection, we study and quantify the impact of the number of available operators on the performance of the proposed reconfigurable logic circuit. Four combinations (CBs) of operators considered in the study are listed below:

- *CB 1:* All 16 operators are available to provide exhaustive exploration for the optimum result, though operators such as NULL and IDENTITY (i.e., the output is always 'FALSE' and 'TRUE') are known to lack logic representation and will not be used in conventional circuits.
- *CB 2:* All operators in CB 1 except the invertible input of operator AND (operator INHIBITION). The reason why input inversion is appended to a certain type of gate is due to the VS RLG's feature of low-cost input-inversion. For technologies that do not have this feature, one can

TABLE 3. Average Number of Gates for Different Operator Combinations and Fence Searching Algorithms

Number of Cells	CB 1	CB 2	CB 3	CB 4
Algorithm 1	6.65	6.65	7.00	7.50
Algorithm 2	6.95	6.95	7.25	7.75

TABLE 4. Average Synthesis Time in Seconds for Different Function CBs and Fence Searching Algorithms

Average Time	CB 1	CB 2	CB 3	CB 4
Algorithm 1	14.43	14.47	19.20	672.45
Algorithm 2	14.66	14.22	13.98	3095

replace it with BUFFER-INVERTOR reconfigurable device or MUX plus INVERTER.

- *CB 3:* All operators in CB 1 except the XOR and XNOR.
- *CB 4:* All operators in CB 1 except the TRANSFER.

As mentioned in the fence searching algorithms in Section III, the reconfigurable synthesis is followed by the standalone synthesis for each operator combination. The combinations for the standalone synthesis do not contain operator TRANSFER, and the reconfigurable synthesis contains TRANSFER (except for CB 4). We perform synthesis for RLG circuits under the four operator combinations and two proposed synthesis algorithms to quantify their impacts on the average number of gates. Function pairs from FDSD6 are used in the simulation. The average number of gates and synthesis time for different fence searching algorithms and operator combinations are shown in Tables 3 and 4, respectively. Note that the synthesis time includes the complete synthesis process, i.e., fence initialization, searching, and the standalone/reconfigurable synthesis.

According to Tables 3 and 4, Algorithm 1 generally outperforms Algorithm 2. This is because the most efficient topologies of the FDSD function are very unbalanced trees (according to the definition of DSD) so the optimal topology of the 2-in-1 FDSD function becomes an unbalanced tree. However, Algorithm 2 leads to a Pyramid-shape topology, which does not match well with the optimal topology of single-output DSD functions. However, Algorithm 2 could be useful for other functions, and we will investigate it in our future research when applying the proposed method to certain large-scale circuits. Regarding the combinations of functions, CB 3 and CB 4 show that lacking operator XOR and TRANSFER will lead to an area penalty. The lack of invertibility of input in operator AND does not lead to a substantial change in the number of gates, which is because the missing input reconfigurability of an operator can be made up by the output reconfigurability of its child operator. In Table 4, the results show that lacking TRANSFER significantly slows down the synthesis. In addition, although all 16 functions are available in CB 1, only operators AND, OR, XOR, TRANSFER, and input-inversed AND (this is the main difference between CB 1 and CB 2) are being used in the simulation. This indicates complete operator sets are not mandatory.

TABLE 5. Performance Benchmark for FullDSD6

Full DSD6	CMOS	STD VS	REC VS	REC VS/ CMOS	REC VS/ STD VS
Energy (fJ)	0.71	0.71	0.42	0.60×	0.59×
Delay (ps)	48.98	63.20	67.06	1.37×	1.06×
Area (um^2)	1.30	1.33	0.84	0.65×	0.63×
EDAP ($fJ \times ps \times um^2$)	44.88	59.88	23.89	0.53×	0.40×
Avg # of Gates	10	10	6.52	0.65×	0.65×

STD: standalone circuit, REC: reconfigurable circuit.

TABLE 6. Performance Benchmark for PartDSD5

Partial DSD5	CMOS	STD VS	REC VS	REC VS/ CMOS	REC VS/ STD VS
Energy (fJ)	0.84	0.77	0.42	0.50×	0.54×
Delay (ps)	58.44	62.92	58.11	0.99×	0.92×
Area (um^2)	1.55	1.36	0.83	0.54×	0.61×
EDAP ($fJ \times ps \times um^2$)	76.29	66.42	20.43	0.27×	0.31×
Avg # of Gates	10.44	10.44	6.45	0.62×	0.62×

Compared to the original state-of-the-art fence searching strategy [31], the proposed fence searching strategy not only accelerates the synthesis but also achieves good performance. Compared to the results in Tables 3 and 4, the synthesis results generated with the original fence searching algorithm and CB 1 show that the average number of gates and synthesis time is 6.61 and 140.7 seconds, respectively. Although the original searching method is performed in parallel with 10 threads simultaneously [31] and we have not applied parallelism to our method, the proposed Algorithm 1 still achieves a very similar number of gates with over 10× improvement in synthesis speed. This is because the original work starts from a circuit with one node and keeps adding nodes to the circuit until the target function is satisfied. For each attempt, the original algorithm tries all possible fences, which is extremely time-consuming. Although the fence-based accelerate method is proven to be able to achieve similar number of gates in DSD benchmarks compared to original method by experiment, a verification of effectiveness will be required before applying this method to larger circuits.

C. PERFORMANCE COMPARISON

Using the experimental setup in Section A and the best deployment found in Section B, i.e., Algorithm 1 and CB 1, we perform simulations for a comprehensive comparison in this subsection. Tables 5 and 6 show the performance results among STD CMOS scheme, STD VS scheme, and REC VS scheme in terms of energy, delay, area, energy-delay-area product (EDAP), and average gate number for two benchmark function sets using Algorithm 1. The delay is calculated by finding the critical path. The last two columns in the tables show the relative performance of REC VS compared to STD

TABLE 7. Performance Comparison Among Proposed Work and Existing Work

		1-bit Adder	2-bit Adder	c17/t	Majority	s27	lion	Daio	Average Improvement of This Work
Energy (fJ)	BUFFER/INV [35]	0.13	0.46	0.32	0.33	0.46	0.39	0.33	10.7
	ABC_SINW [32]	0.26	0.52	0.26	0.26	0.45	0.58	0.26	13.2
	This Work	0.13	0.45	0.26	0.31	0.37	0.31	0.29	
Delay (ps)	Buffer/Inv [35]	14.13	43.14	42.38	58.79	58.03	58.79	29.01	22.6
	ABC_SINW [32]	29.01	29.01	28.25	42.38	42.38	42.38	28.25	8.3
	This Work	14.13	42.38	28.25	44.66	29.01	29.77	29.01	
Area (um^2)	Buffer/Inv [35]	0.26	0.91	0.65	0.65	0.91	0.78	0.65	17.7
	ABC_SINW [32]	0.52	1.04	0.52	0.52	0.91	1.17	0.52	21.7
	This Work	0.26	0.91	0.52	0.54	0.67	0.54	0.44	
EDAP ($fJ \times ps \times um^2$)	Buffer/Inv [35]	0.47	17.81	8.83	12.44	23.96	18.01	6.14	**40.0**
	ABC_SINW [32]	3.92	15.58	3.77	5.65	17.31	28.61	3.77	**26.3**
	This Work	0.47	17.31	3.77	7.58	7.26	5.03	3.73	
Number of Gates	Buffer/inv [35]	2+(4)	7+(5)	5+(3)	5+(3)	7+(4)	6+(5)	5+(4)	
	ABC_SINW [32]	4	8	4	4	7	9	4	
	Proposed	2	7	4	4+1	5+1	4+1	3+2	

# of Gates for [35] is # of normal Gates + (# of zero-cost INV/BUFFER)
# of Gates for proposed method is # of normal Gates + # of MUX

VS and STD CMOS, respectively. Notably, the input/output inversion (INV) is not counted in the average number of gates in Tables 5 and 6 since it is zero-cost for VS gates. However, those INV operators are still considered in calculating the performance in STD CMOS schemes. For large-scale circuits, the usage of inverters might be frequent. If other technologies cannot ignore the cost of inverters, the method proposed in work [46] can be considered to minimize the number of inverters. The average number of INV is 8.49 and 10.21 for FDSD6 and PDSD5, respectively.

The major findings are observed as the following:

1) *Area:* Due to a significant advantage in gate usage (up to 38% reduction), the REC scheme leads to a substantial area reduction (up to 46%) compared to the STD CMOS scheme. STD VS scheme consumes less area than STD CMOS because VS gates have zero-cost INV operation. Nevertheless, according to the advantage of REC VS compared to STD VS, the major area advantage comes from the gate number reduction, i.e., the novel reconfigurable scheme instead of a smaller device.
2) *Energy:* The STD VS logic gate consumes less energy compared to the CMOS due to the efficient XOR implementation and zero-cost INV. Similarly, REC VS consumes much less energy compared to both STD VS and CMOS because fewer RLGs are required to perform each function.
3) *Delay:* The proposed reconfigurable scheme does not result in better delay since it targets the area instead of delay optimization. VS logic gates are in general slower than CMOS gates because of the usage of an inverter at the output stage, while this drawback is compensated by the absence of INV.
4) *EDAP:* The REC VS scheme provides up to 73% improvement in the overall EDAP. In addition, more advantages are observed in the reconfigurable circuits when performing more complex (fewer inputs but similar gate count) PDSD5 functions compared to FDSD6 functions, leading to a 3.2% more usage of XOR in PDSD5.

Gate count and area advantages observed in (1) indicate the proposed function-level REC scheme is promising for improving the area efficiency. The performance advantage of REC VS scheme compared to STD CMOS and STD VS are comparable, demonstrating that the advantage mainly comes from the proposed reconfigurable scheme instead of VS device.

To demonstrate the advantage of our reconfigurable scheme and the synthesis method, we compare this work with two state-of-the-art methods under VS technology. The first work targets area optimization by using input-controlled reconfigurable INV/BUFFER [27]. The second work uses RLGs-based gates by circuit optimization and technology mapping, which can be regarded as the baseline of synthesis that does not involve special design for reconfiguration [25]. The benchmarks are adopted from the first work [27], i.e., selected circuits from the MCNC set [47]. Compared to FDSD/PDSD set that enables massive sampling of single-output circuits, these circuits involve samples of real circuits with multiple outputs. The proposed framework is expanded to support multi-output circuits, and the constraint that enforces all functions to have the same output position is removed. Here, all truth tables of selected circuits are split in half to apply the 2-in-1 method. For the first existing method, we adopt the same setting in the previous work [27] by implementing circuits using non-reconfigurable logic gates AND, XOR, INV, and reconfigurable INV/BUFFER. To ensure a fair comparison, we also deploy post-processing that merges all fixed INV and reconfigurable INV/BUFFER into AND and XOR. This

action makes the available gates for the counterpart similar to CB 4 in Section V-B and thus takes advantage of VS RLGs. For the second method, we adopt the same process in previous work [25] that is based on ABC tool [39]. Except for all INV, BUFFER, and 2-input gates, the library of logic gates for technology mapping also includes VS RLGs-based 3-input gates where one input signal is used to switch the logic function of the gate. Table 7 gives the results of the performance comparison, in which most benchmark circuits implemented by the proposed scheme provide improved EDAP and the number of gates. Note that the number of gates for our scheme includes extra MUXs to select the output based on the output location of the 2-in-1 synthesis.

In Table 7, the proposed method achieves an average improvement of 40% and 26.3% in the overall EDAP compared to the two counterpart methods. The observed advantage is caused by a fundamental difference in methods of realizing and utilizing reconfiguration. Compared to the first existing method, instead of indiscriminately adding INV/BUFFER, our method explicitly deploy reconfigurability by a specified combination of RLGs. This not only allows utilizing TRANSFER, FALSE, TRUE (they are not trivial due to the 2-in-1 mechanism) and arbitrary operator combination (such as AND/XOR, a combination that cannot be realized by adding INV/BUFFER) but also provides customizability of RLGs primitives. This is critical for RLG technologies that are less versatile since they cannot guarantee zero-cost merging between a normal gate and INV/BUFFER. The second existing method can utilize all kinds of reconfigurable gates and achieve better performance than the first method, but it is unable to realize function switch and the proposed scheme still achieves up to 50% advantages in the reduction of gate number.

VI. CONCLUSION

In this article, a generic methodology for designing the function-level reconfigurable logic circuit is proposed, which for the first time, answers the question of how to utilize arbitrary RLGs in function-level reconfigurable circuits. A synthesis framework is proposed to enable design automation based on an SAT-based synthesizer. By properly encoding, initializing, and searching the fence, the synthesizer successfully generates the function-level reconfigurable circuits based on DSD benchmarks. We design four sets of VS RLGs for the case study. An investigation of the operator combinations and algorithms shows operator TRANSFER plays an important role in area control. And the proposed fence searching Algorithm 1 achieves optimum synthesis efficiency ($>10\times$ improvement) and an outstanding average number of gates due to the proper usage of information from standalone synthesis. The experimental results show that the proposed reconfigurable scheme achieves up to 38% and 73% reduction in gate usage and EDAP compared to the standalone CMOS scheme, respectively. The proposed reconfigurable scheme shows an average improvement of 40% and 26.3% in EDAP compared to two existing methods based on MCNC sets.

REFERENCES

[1] C. A. Mack, "Fifty years of Moore's law," *IEEE Trans. Semicond. Manuf.*, vol. 24, no. 2, pp. 202–207, May 2011.

[2] S. Kaxiras, "Architecture at the end of moore," in *Architecture and Design of Molecule Logic Gates and Atom Circuits*. Berlin, Germany: Springer, 2013, pp. 1–10.

[3] L. Wilson, "International technology roadmap for semiconductors (ITRS)," *Semicond. Ind. Assoc.*, vol. 1, pp. 1–17, 2013. [Online]. Available: www.itrs.net

[4] S. Kvatinsky et al., "MAGIC—Memristor-aided logic," *IEEE Trans. Circuits Syst. II: Exp. Briefs*, vol. 61, no. 11, pp. 895–899, Nov. 2014.

[5] G. Berrettini, A. Simi, A. Malacarne, A. Bogoni, and L. Poti, "Ultrafast integrable and reconfigurable XNOR, AND, NOR, and NOT photonic logic gate," *IEEE Photon. Technol. Lett.*, vol. 18, no. 8, pp. 917–919, Apr. 2006.

[6] S. Nishimoto, Y. Yamanashi, and N. Yoshikawa, "Design method of single-flux-quantum logic circuits using dynamically reconfigurable logic gates," *IEEE Trans. Appl. Supercond.*, vol. 25, no. 3, Jun. 2015, Art. no. 1301405.

[7] Y. Zhang, B. Yan, W. Wu, H. Li, and Y. Chen, "Giant spin hall effect (GSHE) logic design for low power application," in *Proc. Des., Automat. Test Europe Conf. Exhib.*, 2015, pp. 1000–1005.

[8] T. Winograd, H. Salmani, H. Mahmoodi, K. Gaj, and H. Homayoun, "Hybrid STT-CMOS designs for reverse-engineering prevention," in *Proc. 53rd Annu. Des. Automat. Conf.*, 2016, pp. 1–6.

[9] S. Angizi, Z. He, A. Chen, and D. Fan, "Hybrid spin-CMOS polymorphic logic gate with application in in-memory computing," *IEEE Trans. Magn.*, vol. 56, no. 2, Feb. 2020, Art. no. 3400215.

[10] R. Zhao et al., "Reconfigurable logic-memory hybrid device based on ferroelectric $Hf_{0.5}Zr_{0.5}O_2$," *IEEE Electron Device Lett.*, vol. 42, no. 8, pp. 1164–1167, Aug. 2021.

[11] Y.-M. Lin, J. Appenzeller, J. Knoch, and P. Avouris, "High-performance carbon nanotube field-effect transistor with tunable polarities," *IEEE Trans. Nanotechnol.*, vol. 4, no. 5, pp. 481–489, Sep. 2005.

[12] C. Murapaka, P. Sethi, S. Goolaup, and W. Lew, "Reconfigurable logic via gate controlled domain wall trajectory in magnetic network structure," *Sci. Rep.*, vol. 6, no. 1, pp. 1–11, 2016.

[13] S. Patnaik, N. Rangarajan, J. Knechtel, O. Sinanoglu, and S. Rakheja, "Spin-orbit torque devices for hardware security: From deterministic to probabilistic regime," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 39, no. 8, pp. 1591–1606, Aug. 2020.

[14] Y. Bi et al., "Emerging technology-based design of primitives for hardware security," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 1, pp. 1–19, 2016.

[15] Z. Marrakchi, H. Mrabet, U. Farooq, and H. Mehrez, "FPGA interconnect topologies exploration," *Int. J. Reconfigurable Comput.*, vol. 2009, 2009, Art. no. 259837.

[16] P.-E. Gaillardon, F. Clermidy, I. O'Connor, J. Liu, M. Amadou, and G. Nicolescu, "Matrix nanodevice-based logic architectures and associated functional mapping method," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 7, no. 1, pp. 1–23, 2011.

[17] L. Sekanina, "Evolutionary design of gate-level polymorphic digital circuits," in *Proc. Workshops Appl. Evol. Comput.*, 2005, pp. 185–194.

[18] W. Luo, Z. Zhang, and X. Wang, "Designing polymorphic circuits with polymorphic gates: A general design approach," *IET Circuits, Devices Syst.*, vol. 1, no. 6, pp. 470–476, 2007.

[19] L. Sekanina, L. Starecek, Z. Kotasek, and Z. Gajda, "Polymorphic gates in design and test of digital circuits," *Int. J. Unconv. Comput.*, vol. 4, no. 2, pp. 125–142, 2008.

[20] H. Liang, W. Luo, and X. Wang, "Designing polymorphic circuits with evolutionary algorithm based on weighted sum method," in *Proc. Int. Conf. Evolvable Syst.*, 2007, pp. 331–342.

[21] Z. Gajda and L. Sekanina, "On evolutionary synthesis of compact polymorphic combinational circuits," *J. Mult. Valued Log. Soft Comput.*, vol. 17, no. 5-6, pp. 607–631, 2011.

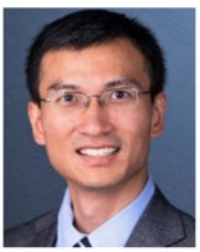
[22] M. H. B. Jamaa, K. Mohanram, and G. De Micheli, "Novel library of logic gates with ambipolar CNTFETs: Opportunities for multi-level logic synthesis," in *Proc. Des., Automat. Test Europe Conf. Exhib.*, 2009, pp. 622–627.

[23] M. H. Ben-Jamaa, K. Mohanram, and G. De Micheli, "An efficient gate library for ambipolar CNTFET logic," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 30, no. 2, pp. 242–255, Feb. 2011.

[24] S. Rai, J. Trommer, M. Raitza, T. Mikolajick, W. M. Weber, and A. Kumar, "Designing efficient circuits based on runtime-reconfigurable field-effect transistors," *IEEE Trans. Very Large Scale Integration Syst.*, vol. 27, no. 3, pp. 560–572, Mar. 2019.
[25] S. Rai, M. Raitza, and A. Kumar, "Technology mapping flow for emerging reconfigurable silicon nanowire transistors," in *Proc. Des., Automat. Test Europe Conf. Exhib.*, 2018, pp. 767–772.
[26] P. Fišer, I. Háleček, J. Schmidt, and V. Šimek, "SAT-based generation of optimum circuits with polymorphic behavior support," *J. Circuits, Syst. Comput.*, vol. 28, no. supp01, 2019, Art. no. 1940010.
[27] P. Fiser and V. Simek, "Optimum polymorphic circuits synthesis method," in *Proc. 13th Int. Conf. Des. Technol. Integr. Syst. Nanoscale Era*, 2018, pp. 1–6.
[28] A. Crha, V. Šimek, and R. Ružicka, "Paig rewriting: The way to scalable multifunctional digital circuits synthesis," in *Proc. IEEE 22nd Euromicro Conf. Digit. Syst. Des.*, 2019, pp. 335–342.
[29] A. Crha, V. Šimek, and R. Růžička, "KL-cuts influence on optimization of polymorphic circuits based on PAIG rewriting," in *Proc. IEEE 23rd Int. Symp. Des. Diagn. Electron. Circuits Syst.*, 2020, pp. 1–6.
[30] W. Haaswijk, M. Soeken, L. Amarú, P.-E. Gaillardon, and G. De Micheli, "A novel basis for logic rewriting," in *Proc. IEEE 22nd Asia South Pacific Des. Automat. Conf.*, 2017, pp. 151–156.
[31] W. Haaswijk, M. Soeken, A. Mishchenko, and G. De Micheli, "SAT-based exact synthesis: Encodings, topology families, and parallelism," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 39, no. 4, pp. 871–884, Apr. 2020.
[32] S. Kvatinsky, G. Satat, N. Wald, E. G. Friedman, A. Kolodny, and U. C. Weiser, "Memristor-based material implication (IMPLY) logic: Design principles and methodologies," *IEEE Trans. Very Large Scale Integration Syst.*, vol. 22, no. 10, pp. 2054–2066, Oct. 2014.
[33] G. Papandroulidakis, A. Serb, A. Khiat, G. V. Merrett, and T. Prodromakis, "Practical implementation of memristor-based threshold logic gates," *IEEE Trans. Circuits Syst. I: Regular Papers*, vol. 66, no. 8, pp. 3041–3051, Aug. 2019.
[34] K. M. Kim and R. S. Williams, "A family of stateful memristor gates for complete cascading logic," *IEEE Trans. Circuits Syst. I: Regular Papers*, vol. 66, no. 11, pp. 4348–4355, Nov. 2019.
[35] F. Parveen, Z. He, S. Angizi, and D. Fan, "Hybrid polymorphic logic gate with 5-terminal magnetic domain wall motion device," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, 2017, pp. 152–157.
[36] C. Pan et al., "Reconfigurable logic and neuromorphic circuits based on electrically tunable two-dimensional homojunctions," *Nature Electron.*, vol. 3, pp. 383–390, 2020.
[37] L. Tao, A. Naeemi, and E. Y. Tsymbal, "Valley-spin logic gates," *Phys. Rev. Appl.*, vol. 13, no. 5, 2020, Art. no. 054043.
[38] N. Li and E. Dubrova, "AIG rewriting using 5-input cuts," in *Proc. IEEE 29th Int. Conf. Comput. Des.*, 2011, pp. 429–430.
[39] R. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool," in *Proc. Comput. Aided Verification: 22nd Int. Conf.*, CAV 2010, Edinburgh, UK, Springer, Berlin, Heidelberg, Jul. 15-19, 2010, pp. 24–40.
[40] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 26, no. 2, pp. 203–215, Feb. 2007.
[41] N. Sorensson and N. Een, "Minisat v1. 13-a sat solver with conflict-clause minimization," *SAT*, vol. 53, pp. 1–2, 2005.
[42] L. Tao and E. Y. Tsymbal, "Two-dimensional spin-valley locking spin valve," *Phys. Rev. B*, vol. 100, no. 16, 2019, Art. no. 161110.
[43] D. E. Nikonov and I. A. Young, "Overview of beyond-CMOS devices and a uniform methodology for their benchmarking," *Proc. IEEE*, vol. 101, no. 12, pp. 2498–2533, Dec. 2013.
[44] A. Mishchenko, S. Chatterjee, and R. Brayton, "DAG-aware AIG rewriting: A fresh look at combinational logic synthesis," in *Proc. IEEE/ACM 43rd Des. Automat. Conf.*, 2006, pp. 532–535.
[45] C. Pan and A. Naeemi, "An expanded benchmarking of beyond-CMOS devices based on boolean and neuromorphic representative circuits," *IEEE J. Explor. Solid-State Comput. Devices Circuits*, vol. 3, pp. 101–110, Dec. 2017.
[46] E. Testa et al., "Inversion optimization in majority-inverter graphs," in *Proc. IEEE/ACM Int. Symp. Nanoscale Architectures*, 2016, pp. 15–20.
[47] S. Yang, *Logic Synthesis and Optimization Benchmarks User Guide: Version 3.0*, Research Triangle Park, NC, USA: Microelectronics Center of North Carolina (MCNC), pp. 502–508, 1991.

LIUTING SHANG (Student Member, IEEE) received the B.S. degree in measurement and control technology and instrument from Shandong University, Jinan, China, in 2016, and the M.S. degree in signal and information processing from Southwest University, Chongqing, China, in 2019. He is currently working toward the Ph.D. degree in EE with the Department of Electrical Engineering, The University of Texas at Arlington, Arlington, TX, USA. His research interests include device- and circuit-level design for in-memory computing circuits and reconfigurable logic circuits based on emerging devices.

AZAD NAEEMI (Senior Member, IEEE) received the B.S. degree in EE from Sharif University, Tehran, Iran, in 1994, and the Ph.D. degree in ECE from the Georgia Institute of Technology, Atlanta, GA, USA, in 2003. He is currently a Professor with the Georgia Institute of Technology, exploring nanotechnology solutions to the challenges facing giga- and tera-scale systems. His research interests include the boundaries of materials, devices, circuits, and systems investigating integrated circuits based on conventional and emerging nanoelectronic and spintronic devices and interconnects. He was the recipient of the IEEE Electron Devices Society Paul Rappaport Award for the best paper that appeared in IEEE Transactions on Electron Devices in 2007. He was also the first recipient of the IEEE Solid-State Circuits Society James D. Meindl Innovators Award in 2022. He was the recipient of NSF CAREER Award, an SRC Inventor Recognition Award, and several best paper awards at international conferences.



CHENYUN PAN (Senior Member, IEEE) received the B.S. degree in microelectronics from Shanghai Jiao Tong University, Shanghai, China, in 2010 and the Ph.D. degree in ECE from the Georgia Institute of Technology, Atlanta, GA, USA, in 2015. He is currently an Assistant Professor with the Department of Electrical Engineering, The University of Texas at Arlington, Arlington, TX, USA. His research interests include device-, circuit-, and system-level modeling and optimization for energy-efficient Boolean and non-Boolean computing systems based on various emerging device and interconnect technologies. He received Research Spotlight Award in 2018, when he was a Research Engineer with the Institute for Electronics and Nanotechnology, Georgia Institute of Technology. He was the recipient of the Early Career Research Program Award from the Department of Energy in 2022.