Optimization of Time-multiplexed Constant Multiplication via Ising Model

Wenhui Zhang[1], Hui Wang[1], Jie Han[2] and Honglan Jiang[1]

***Abstract*—In many applications, such as digital signal processing, multiple-constant multiplications are commonly required, where time-multiplexed constant multiplication (TMCM) is typically employed for efficient hardware utilization. Although an optimal design for a constant multiplication can be easily obtained by using shift-add operations, it is of great difficulty to find an optimal architecture supporting the multiplication of an input with multiple constants due to the exponential increase of the search space. To address this issue, in this work, we propose an efficient Ising model-based TMCM optimization framework. Specifically, the directed acyclic graphs (DAGs) for each constant multiplication are generated under the minimum adder constraint. TMCM is then designed by merging the DAGs for all constants. To achieve an optimal design, we convert the DAG merging to a Quadratic Unconstrained Binary Optimization (QUBO) problem that can be solved by the Ising model. Experimental results show that when applied to several popular fast Fourier transform (FFT) and discrete cosine transform (DCT) coefficient sets, the proposed framework can obtain TMCMs with fewer multiplexers, leading to up to 12.76%, 8.02%, and 18.48% savings in area, power, and power-delay product (PDP), compared to state-of-the-art works.**

***Index Terms*—time-multiplexed constant multiplication, Ising model, resource minimization**

I. Introduction

Constant multiplication is a key operation in many digital signal processing (DSP) systems [1]–[3]. When one operand of the multiplication is a predetermined constant, a general-purpose multiplier can be replaced by a shift-add-based implementation composed of adders, subtractors, and shifters, which significantly reduces hardware cost. According to the number of involved constants, existing design approaches are generally classified as single constant multiplication (SCM) [4], [5] and multiple constant multiplication (MCM) [6]–[9].

Further hardware reduction can be achieved by introducing multiplexers into SCM and MCM architectures, leading to time-multiplexed constant multiplication (TMCM) [6]–[12]. In a typical TMCM flow, optimized directed acyclic graphs (DAGs) are first constructed for individual constants, then merged into a shared datapath through multiplexers. Existing methods mainly focus on minimizing adders, often introducing a significant number of multiplexers [6], [7], [11]. Various heuristic algorithms, exact search methods, and genetic algorithms [9], [10], [12] have been explored for designing low-complexity shift-add-based constant multipliers. Nevertheless, the search space grows exponentially with the number of constants, causing excessive solving time and limiting scalability. Consequently, adapting to different coefficient sets and architectural constraints becomes difficult.

These limitations indicate the need for a formulation capturing the global structure of the optimization problem. The processes of DAG fusion and resource sharing in TMCM involve numerous binary decisions, including the selection of adder graphs as well as node and edge matching. Such characteristics indicate that TMCM optimization can be formulated as a combinatorial optimization problem with strong dependencies among variables. The Ising model has recently emerged as an effective mathematical framework for representing and solving NP-hard combinatorial optimization problems through Quadratic Unconstrained Binary Optimization (QUBO) formulations [13]. Its capability of modeling binary variables and pairwise interactions makes it well-suited for representing relationships among DAG fusion, adder reuse, and multiplexer allocation in TMCM architectures.

Based on this observation, this work proposes an Ising-model-based optimization framework for TMCM. The method first constructs minimum-adder DAGs for individual constants and then formulates DAG fusion and hardware resource allocation as a unified QUBO problem, enabling global optimization of the overall architecture. The main contributions of this work are summarized as follows.

- An Ising-model-based formulation for TMCM is developed, enabling global optimization of architectural decisions in the design process.
- A TMCM search framework is proposed by enumerating candidate DAG structures, where adaptive threshold pruning (ATP) is applied to maintain tractable computation.
- Experimental evaluation on widely used benchmark coefficient sets demonstrates the effectiveness of the proposed framework. The synthesis results show that the obtained TMCMs reduce the structural complexity, area, power, and power-delay product (PDP) compared with existing state-of-the-art designs.

II. Preliminaries

A. Ising Model

The Ising model is widely used to represent combinatorial optimization problems. An M-spin Ising problem determines

*This work was supported in part by the National Natural Science Foundation of China under Grant 62374108, and in part by the Natural Sciences and Engineering Research Council of Canada under Grant RES0048688, Grant RES0051374, and Grant RES0054326. (Corresponding authors: Honglan Jiang)

[1]W. Zhang, H. Wang and H. Jiang are with the School of Integrated Circuits, Shanghai Jiao Tong University, Shanghai, China, 200240. (e-mail: whyl0809@sjtu.edu.cn, ihuiwang@sjtu.edu.cn, and honglan@sjtu.edu.cn)

[2]J. Han is with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada. (e-mail: jhan8@ualberta.ca)

the spin configuration $\boldsymbol{s}$ that minimizes the Hamiltonian energy [14], which is defined as

$$H(\boldsymbol{s}) = -\sum_{i=1}^{M} h_i s_i - \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M} J_{ij} s_i s_j, \tag{1}$$

where $s_i \in \{-1, +1\}$ denotes the state of the i_{th} spin, h_i denotes the local field associated with s_i, and J_{ij} represents the coupling coefficient between s_i and s_j.

For optimization purposes, the spin variables can be rewritten using binary variables $\sigma_i \in \{0, 1\}$, where $s_i = 2\sigma_i - 1$. This transformation leads to a QUBO formulation

$$H(\boldsymbol{\sigma}) = a_0 - \sum_{i=1}^{M} h'_i \sigma_i - \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M} J'_{ij} \sigma_i \sigma_j, \tag{2}$$

where $h'_i = 2(h_i - \sum_{j=1}^{M} J_{ij})$, $J'_{ij} = 4J_{ij}$ and $a_0 = \sum_{i=1}^{M} h_i - \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M} J_{ij}$. Since a_0 does not affect the optimal configuration, minimizing $H(\boldsymbol{\sigma})$ is equivalent to solving the original Ising problem. This QUBO representation provides a convenient framework for modeling binary design decisions in hardware optimization problems.

B. Constant Multiplication

Constant multiplication problems are commonly modeled using DAGs. In this representation, nodes correspond to intermediate values generated by shift and add operations, while edges indicate arithmetic dependencies. The objective is to construct a minimum-adder graph (MAG) by identifying shared intermediate values, referred to as *fundamentals*. Typically, a normalized or non-normalized form of DAG representations is considered. In the normalized form, intermediate values are restricted to odd numbers, which avoids redundant representations and reduces the search space. In contrast, non-normalized DAGs allow both odd and even intermediates, thereby relaxing this restriction.

Although normalized graphs are sufficient for SCM, they may limit the sharing opportunities when merging the graphs of multiple constants. Non-normalized intermediates enlarge the design space and can improve structural alignment between DAGs, as illustrated in Fig. 1. The normalized DAGs for the constants 149 and 142 are shown in Fig. 1(a) and (b), respectively, while Fig. 1(c) presents a non-normalized representation for 142. When the graphs are fused for time-multiplexed implementation, the non-normalized DAG in Fig. 1(e) enables greater sharing than the normalized fusion shown in Fig. 1(d). This observation motivates our exploration of a larger DAG search space by using non-normalized DAGs when optimizing TMCM architectures.

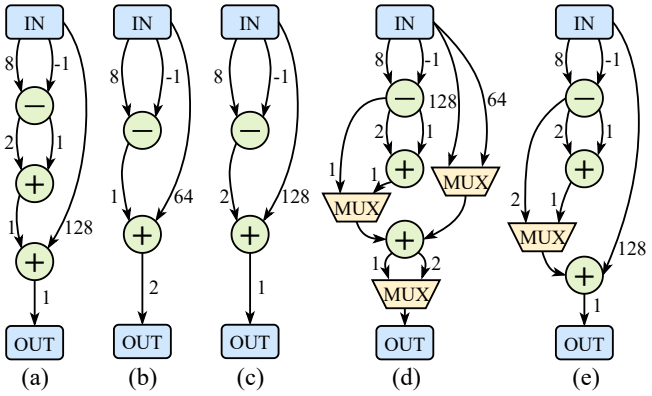

Fig. 1: (a) Normalized DAG for 149. (b) Normalized DAG for 142. (c) Non-normalized DAG for 142. (d) Time multiplexed DAG for normalized 149 and 142. (e) Time multiplexed DAG for non-normalized 149 and 142.

Algorithm 1: TMCM Optimization Framework

Input: Coefficient set $C = \{c_i\}, 1 \le i \le N$
Output: G_{out}: optimized DAG for C

// Step 1: Candidate DAG generation
for $i = 1$ **to** N **do**
  $G_{c_i} = \text{MAG}(c_i)$
  $G_{c_i} = \text{EWT}(G_{c_i})$
$G_C = \text{CartesianProduct}(G_{c_1}, ..., G_{c_N})$
// Step 2: Adaptive Threshold Pruning (ATP)
$G_C = \text{ATP}(G_C)$
// Step 3: Negative Node Transformation (NFT)
$G_C = \text{NFT}(G_C)$
// Step 4: Ising-Based DAG Fusion and Selection (IDFS)
$G_{out} = \text{IDFS}(G_C)$

III. Proposed Ising Model-based TMCM Framework

This section presents an Ising-model-based optimization framework that jointly selects and fuses DAG structures for TMCM implementations with reduced hardware cost. The overall procedure is summarized in Algorithm 1.

A. Optimization Framework Overview

The proposed framework optimizes TMCM architectures through a multi-stage procedure that generates candidate DAGs, prunes inefficient combinations, and finally performs global optimization via an Ising formulation.

First, candidate DAGs are generated for each constant using the MAG construction method [4], [15]. To enlarge the design space, the EWT expansion technique [8] is applied to each generated DAG, enabling additional intermediate nodes that relax the normalization constraint and improve structural flexibility for subsequent fusion.

The candidate DAGs for different constants are then combined to form a global DAG combination for the TMCM. A DAG combination is constructed by selecting a candidate DAG for each constant. Many DAG combinations lead to structurally unreasonable TMCM implementations, which should not be considered in the final optimization stage. Thus, an ATP strategy is introduced to eliminate such clearly implausible configurations, which can accelerate the searching process.

ATP prunes the candidate combinations by using a multi-level search tree, where each level corresponds to the DAG selection for a constant. During traversal, a lightweight structural estimate is computed for each partial combination to assess its feasibility. The branches whose estimated structural cost exceeds the adaptive threshold are pruned early, preventing the

TABLE I: Description of the Ising model: constants (upper section) and decision variables (lower section).

Constant/variable	Explanation
$N \in \mathbb{N}$	number of target constants
$T \in \mathbb{N}$	number of adders used in DAG fusion
$K^t \in \mathbb{N}$	number of possible (weight, source) combinations for the inputs of node t
$BW_A^t \in \mathbb{N}$	bit width of node t
$BW_L^t \in \mathbb{N}$	bit width of left-input MUX of node t
$BW_R^t \in \mathbb{N}$	bit width of right-input MUX of node t
$W_L^t \in \{0,1\}^{N \times K^t}$	one-hot encoding of left-input (weight, source) combinations of node t
$W_R^t \in \{0,1\}^{N \times K^t}$	one-hot encoding of right-input (weight, source) combinations of node t
$P_L^t \in \{0,1\}^N$	1 if left-input weight of node t is positive, 0 otherwise
$P_R^t \in \{0,1\}^N$	1 if right-input weight of node t is positive, 0 otherwise
$x^t \in \{0,1\}^N$	port swap indicator at node t (1 if the two input weights of adder ports are swapped)
$y_{add}^t \in \{0,1\}$	adder indicator for node t (1 if adder)
$y_{sub}^t \in \{0,1\}$	subtractor indicator for node t (1 if subtractor)
$y_{as}^t \in \{0,1\}$	add/subtract indicator for node t (1 if add/subtract)
$u^t \in \{0,1\}^{K^t}$	activation of corresponding (weight, source) combination at left-input of node t
$v^t \in \{0,1\}^{K^t}$	activation of corresponding (weight, source) combination at right-input of node t
$\gamma^t \in \{0,1\}$	1 if at least one left-input weight of node t is positive
$\delta^t \in \{0,1\}$	1 if at least one left-input weight of node t is negative
$\zeta^t \in \{0,1\}$	1 if at least one right-input weight of node t is positive
$\eta^t \in \{0,1\}$	1 if at least one right-input weight of node t is negative

exploration of clearly inefficient configurations. The remaining candidates are retained as potentially feasible implementations. In this framework, ATP functions only as a filtering stage that removes structurally implausible DAG combinations rather than identifying the optimal solution.

To further enlarge the feasible design space, a Negative Node Transformation (NFT) stage is applied to the remaining DAG combinations. Prior studies [16], [17] indicate that implementations using negative *fundamentals* may result in fewer hardware resources. In addition, introducing negative intermediate nodes increases the likelihood of structural alignment across different DAGs, thereby improving hardware-sharing opportunities during DAG fusion.

Finally, the fusion and selection of the candidate DAGs are formulated as an Ising optimization problem. In this formulation, binary variables represent DAG selection decisions, while pairwise interaction terms model the sharing relationships between intermediate nodes across DAGs. This representation enables joint optimization of DAG selection and hardware sharing. A detailed formulation of the Ising model is provided in Section III-B.

B. Ising Formulation of TMCM design

The optimization objective of the proposed TMCM formulation is to minimize the hardware implementation cost of the fused shift-add graph. To enable a hardware-aware optimization, the circuit cost is first expressed using platform-independent parameters α that represent the relative area contributions of arithmetic and routing components. Each hardware component, including adder (ADD), subtractor (SUB), add/subtract unit (AS), and n : 1 multiplexer (MUX), is associated with an area coefficient α equal to 67, 75, 93, and $14 \cdot n$, respectively. These coefficients are derived from a commercial $0.18\mu m$ standard cell library [6], [8], [9] and are used to provide relative cost scaling among different circuit components. The total hardware cost is therefore defined as

$$\begin{aligned} COST = &\sum_{i=1}^{N_A} 67 \cdot BW_A(i) + \sum_{i=1}^{N_S} 75 \cdot BW_S(i) \\ &+ \sum_{i=1}^{N_{AS}} 93 \cdot BW_{AS}(i) + \sum_{i=1}^{N_M} 14 \cdot n_i \cdot BW_M(i) \end{aligned}, \quad (3)$$

where N_A, N_S, N_{AS}, and N_M denote the numbers of ADD, SUB, AS, and MUX units, respectively. $BW_A(i)$, $BW_S(i)$, $BW_{AS}(i)$, and $BW_M(i)$ denote the bit widths of the corresponding components, while n_i denotes the number of input ports of the i_{th} multiplexer.

Let Q be the number of candidate DAG combinations for a given set of N constants. After selecting a specific DAG combination, the fused graph contains T adder nodes. At each node, every constant contributes a pair of input weights corresponding to the two operands of the shift-add operation. The hardware cost of the fused implementation depends on how these weights are assigned to the left and right input ports of each node, i.e., the port assignments. This is because different port assignments lead to different multiplexer configurations and arithmetic unit types. To perform global optimization over these port assignments, the TMCM design problem is formulated as an Ising model. This representation enables compact encoding of pairwise sharing relationships arising from DAG fusion, which naturally introduces quadratic interaction terms in the optimization formulation. In this formulation, binary spin variables represent port selection, multiplexer activation, and arithmetic unit types, while pairwise interactions encode structural consistency constraints during graph fusion. Table I summarizes the constants and decision variables used in the formulation.

For a given DAG combination, the circuit implementation cost in (3) is directly incorporated into the objective Hamiltonian as

$$\begin{aligned} H_{obj} = &\sum_{t=1}^{T} (H_{add,obj}^t + H_{mux,obj}^t) + H_{mux,obj}^O \\ = &\sum_{t=1}^{T} (67 \cdot BW_A^t \cdot y_{add}^t + 75 \cdot BW_A^t \cdot y_{sub}^t + 93 \cdot BW_A^t \cdot y_{as}^t, \\ &+ 14 \cdot BW_L^t \cdot \sum_{k=1}^{K^t} u_k^t + 14 \cdot BW_R^t \cdot \sum_{k=1}^{K^t} v_k^t) + 14 \cdot n_O \cdot BW_O \end{aligned} \quad (4)$$

where n_O and BW_O denote the number of input ports and the bit width of the output multiplexer, respectively.

To guarantee valid circuit configurations during optimization, additional penalty terms are introduced to enforce consistency among port assignments, multiplexer activations, operand sign configurations, and arithmetic unit types. For each node t, the spin variable $x^t = \{x_1^t, x_2^t, \ldots, x_N^t\}$ determines whether the operand pair of the i_{th} constant is swapped between the two input ports. If a specific (weight, source) combination is assigned to a port according to x^t, the

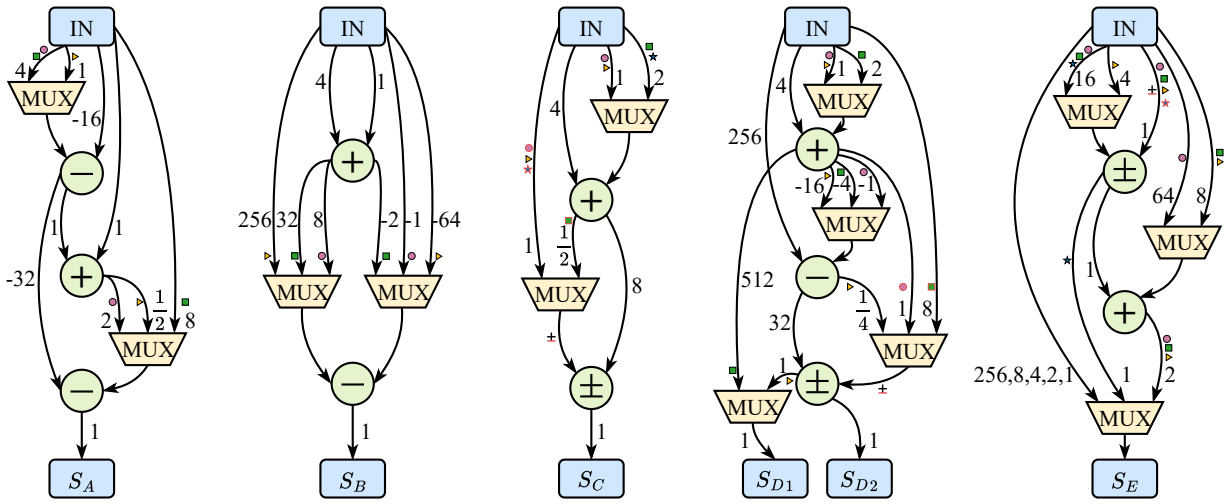

Fig. 2: Proposed time-multiplexed constant multiplication architectures for various coefficient sets with select signals, where $S_A = [362, 392, 473]$, $S_B = [39, 150, 192]$, $S_C = [39, 45, 41, 47]$, $S_D = [8027, 7416 + j3072, 5676 + j5676]$ and $S_E = [162, 50, 26, 15, 256, 8, 4, 2, 1]$. (magenta circle: $sel = 0$, green square: $sel = 1$, orange triangle: $sel = 2$, blue star: $sel = 3$.)

corresponding multiplexer input defined by u^t or v^t must be activated. This consistency constraint is enforced by[1]

$$\begin{aligned} H^t_{mux,pen} = \lambda_M \cdot \sum_{i=1}^{N}\sum_{k=1}^{K^t} (\overline{x}^t_i \cdot W^t_{L,ik} \cdot \overline{u}^t_k + x^t_i \cdot W^t_{R,ik} \cdot \overline{u}^t_k) \\ + \lambda_M \cdot \sum_{i=1}^{N}\sum_{k=1}^{K^t} (x^t_i \cdot W^t_{L,ik} \cdot \overline{v}^t_k + \overline{x}^t_i \cdot W^t_{R,ik} \cdot \overline{v}^t_k) \end{aligned} . \quad (5)$$

The arithmetic type of an adder node depends on the sign distribution of its input weights. Four spin variables, γ^t, δ^t, ζ^t, and η^t, indicate the presence of positive and negative operands on the left and right ports of node t, as summarized in Table I. Invalid configuration in which the left or right input of node t contains no valid sign indicator is penalized by

$$H^t_{port,zero} = \lambda_E \cdot (\overline{\gamma}^t \cdot \overline{\delta}^t + \overline{\zeta}^t \cdot \overline{\eta}^t). \quad (6)$$

Additionally, γ^t, δ^t, ζ^t, and η^t must remain consistent with the weights assigned to each port at node t, which is defined as

$$\begin{aligned} H^t_{port,sign} = \frac{\lambda_P}{4} \cdot (\gamma^t + \delta^t + \zeta^t + \eta^t) \\ + \lambda_P \cdot \sum_{i=1}^{N} (P^t_{L,i} \cdot \overline{x}^t_i \cdot \overline{\gamma}^t + P^t_{R,i} \cdot x^t_i \cdot \overline{\gamma}^t) \\ + \lambda_P \cdot \sum_{i=1}^{N} (N^t_{L,i} \cdot \overline{x}^t_i \cdot \overline{\delta}^t + N^t_{R,i} \cdot x^t_i \cdot \overline{\delta}^t) \\ + \lambda_P \cdot \sum_{i=1}^{N} (P^t_{L,i} \cdot x^t_i \cdot \overline{\zeta}^t + P^t_{R,i} \cdot \overline{x}^t_i \cdot \overline{\zeta}^t) \\ + \lambda_P \cdot \sum_{i=1}^{N} (N^t_{L,i} \cdot x^t_i \cdot \overline{\eta}^t + N^t_{R,i} \cdot \overline{x}^t_i \cdot \overline{\eta}^t)) \end{aligned} , \quad (7)$$

where $N^t_{L,i} = 1 - P^t_{L,i}$ and $N^t_{R,i} = 1 - P^t_{R,i}$.

Each adder node can be implemented using one of three arithmetic units, i.e., ADD, SUB, or AS. Three spin variables y^t_{add}, y^t_{sub}, and y^t_{as} represent the selected unit type. A one-hot constraint ensures that exactly one arithmetic unit is selected, which is given by

$$H^t_{type,onehot} = \lambda_O \cdot (y^t_{add} + y^t_{sub} + y^t_{as} - 1)^2. \quad (8)$$

Furthermore, the selected arithmetic unit must remain consistent with the operand sign configuration. This constraint is enforced by

$$\begin{aligned} H^t_{type,sign} = \lambda_T \cdot y^t_{add} \cdot (\overline{\gamma}^t \cdot \overline{\eta}^t + \overline{\delta}^t \cdot \overline{\zeta}^t + \gamma^t \cdot \delta^t + \zeta^t \cdot \eta^t) \\ + \lambda_T \cdot y^t_{sub} \cdot (\overline{\gamma}^t \cdot \overline{\zeta}^t + \overline{\delta}^t \cdot \overline{\eta}^t + \gamma^t \cdot \delta^t + \zeta^t \cdot \eta^t) \\ + \lambda_T \cdot y^t_{as} \cdot (\overline{\gamma}^t \cdot \overline{\zeta}^t + \overline{\delta}^t \cdot \overline{\eta}^t + \overline{\gamma}^t \cdot \overline{\eta}^t + \overline{\delta}^t \cdot \overline{\zeta}^t) \end{aligned} . \quad (9)$$

By combining the objective cost in (4) with all constraint penalties, the Hamiltonian for q_{th} DAG combination fusion is described by

$$\begin{aligned} H_q = H_{obj} + \sum_{t=1}^{T} (H^t_{mux,pen} + H^t_{port,zero} \\ + H^t_{port,sign} + H^t_{type,onehot} + H^t_{type,sign}) \end{aligned} . \quad (10)$$

The spin variables $z = \{z_1, z_2, \ldots, z_Q\}$ indicate which fused DAG combination is selected. The final Ising Hamiltonian is formulated as

$$H = \sum_{q=1}^{Q} z_q \cdot H_q + \lambda_Q \Big(\sum_{q=1}^{Q} z_q - 1 \Big)^2. \quad (11)$$

Minimizing this Hamiltonian simultaneously determines the optimal fused DAG combination (via the selection variables z_q) and the associated port assignments, multiplexer configurations, and arithmetic unit selections, thereby minimizing the overall hardware cost of the fused implementation. The resulting optimization problem is solved using the Gurobi Optimizer (version 13.0.0) with multi-threading enabled [18]. The solver is configured with a relative optimality gap of 10^{-2} and a time limit of 300 seconds when necessary. While Gurobi is employed here as an efficient heuristic backend for solving the formulated QUBO instances, the proposed model itself is solver-agnostic and can be directly deployed on standard Ising-based optimization platforms.

[1]In this paper, different penalty coefficients, denoted by λ_*, are used to control the relative importance of the penalty terms in the Ising model. $\overline{b}$ denotes the complement of the binary spin variable b, i.e., $1 - b$.

IV. Experimental Results

To evaluate the effectiveness of the proposed TMCM framework, several coefficient sets derived from representative DSP applications are considered. The sets $S_A = [362, 392, 473]$, $S_B = [39, 150, 192]$ and $S_C = [39, 45, 41, 47]$ are taken from previously reported benchmark designs associated with DCT implementations and related signal processing kernels [6], [8], [9], [19]. In addition, the coefficient set $S_D = [8027, 7416 + j3072, 5676 + j5676]$, which appears in FFT rotator implementations, is used to construct a double-output TMCM architecture [6], [9]. Another coefficient set, $S_E = [162, 50, 26, 15, 256, 8, 4, 2, 1]$, is obtained from a digital FIR filter design reported in [20]. These coefficients provide a representative and realistic evaluation scenario for assessing the hardware efficiency of the proposed TMCM framework. Note that the bit-width of the input x is set to the commonly used 8-bit in our experiment.

A. Hardware Cost Evaluation

The optimized TMCM architectures of the considered coefficient sets are shown in Fig. 2. Table II presents the hardware estimation results of different TMCM designs for coefficient sets S_A-S_E across different optimization methods according to (3). Table II shows that the proposed framework achieves the highest hardware efficiency for all evaluated coefficient sets. For S_B, S_D, and S_E, the proposed design provides the minimum hardware cost among all compared methods [6], [8], [9], while it achieves the same optimal cost as the best existing designs for S_A and S_C. The advantage of the proposed framework becomes more apparent for coefficient sets with larger dynamic ranges or more complex sharing structures. For S_D, the estimated hardware cost is reduced from 5833 in [8], [9] to 5691, corresponding to a reduction of approximately 2.43%. For S_E, the cost decreases from 4439 to 4210, achieving an improvement of about 5.16% compared with [9].

The main reason for this improvement lies in the fact that the proposed framework explores different input port assignments during the optimization process based on the Ising model, which allows more efficient sharing of intermediate nodes among multiple constants. Specifically, the proposed method carefully determines the types and bit widths of addition operators according to the input operand characteristics, favoring simpler ADD or SUB units while avoiding the more expensive AS units. In addition, the allocation of left or right-input weight for each adder node is considered during the optimization process of the Ising model, providing greater possibilities for resource-sharing. As a result, the number and size of multiplexers can be reduced or better balanced. For example, in coefficient set S_B, the proposed design replaces several small multiplexers with two balanced 3:1 multiplexers, leading to a lower overall cost. For S_D, the proposed design adopts a smaller 8-bit ADD unit instead of the 10-bit AS unit used in [8], [9], which significantly reduces the arithmetic cost.

TABLE II: Hardware cost estimation results of different TMCM designs for coefficient sets S_A-S_E.

Coef. Sets	Method	ADD/SUB	MUX	Cost
S_A	[6]	12-bit AS 14-bit SUB 17-bit ADD	10-bit 2:1 11-bit 2:1 15-bit 2:1 12-bit 3:1 17-bit 3:1	5531
	[8]	8-bit SUB 12-bit ADD 12-bit SUB	10-bit 2:1 13-bit 3:1	3130
	[9]	8-bit SUB 12-bit ADD 12-bit SUB	10-bit 2:1 13-bit 3:1	3130
	Prop.	**8-bit SUB** **12-bit ADD** **12-bit SUB**	**10-bit 2:1** **13-bit 3:1**	**3130**
S_B	[8]	10-bit ADD 15-bit SUB	9-bit 2:1 11-bit 2:1 16-bit 3:1	3027
	[9]	10-bit ADD 15-bit SUB	9-bit 2:1 11-bit 2:1 16-bit 3:1	3027
	Prop.	**8-bit ADD** **16-bit SUB**	**13-bit 3:1** **14-bit 3:1**	**2870**
S_C	[6]	10-bit ADD 14-bit AS	9-bit 2:1 2*(11-bit 2:1)	2840
	[8]	8-bit ADD 14-bit AS	9-bit 2:1 10-bit 2:1	2370
	[9]	12-bit AS 13-bit ADD	9-bit 2:1 12-bit 2:1	2575
	Prop.	**8-bit ADD** **14-bit AS**	**9-bit 2:1** **10-bit 2:1**	**2370**
S_D	[8]	10-bit AS 16-bit SUB 21-bit AS	15-bit 3:1 14-bit 3:1 19-bit 2:1	5833
	[9]	10-bit AS 16-bit SUB 21-bit AS	15-bit 3:1 14-bit 3:1 19-bit 2:1	5833
	Prop.	**8-bit ADD** **16-bit SUB** **21-bit AS**	**9-bit 2:1** **15-bit 3:1** **14-bit 3:1** **19-bit 2:1**	**5691**
S_E	[6]	12-bit AS 14-bit AS	12-bit 3:1 16-bit 7:1	4490
	[9]	12-bit ADD 15-bit AS	12-bit 3:1 14-bit 2:1 16-bit 6:1	4439
	Prop.	**12-bit AS** **14-bit ADD**	**10-bit 2:1** **11-bit 2:1** **16-bit 7:1**	**4210**

B. Synthesis Results

To further assess the circuit characteristics, the proposed TMCM framework and other state-of-the-art multiplierless architectures [6], [8], [9] for all considered coefficient sets are implemented using Synopsys Design Compiler (DC) and the 45nm NangateOpenCell library in the typical condition with 25°C and 0.9V supply voltage. Taking switching activity into account, a waveform file containing one million random input combinations with a frequency of 500MHz is generated by the Verilog Compile Simulator (VCS), which is used as the stimulus of the TMCM designs to estimate the power dissipation.

Table III shows detailed information on area, power, and delay for the ASIC implementation, where DW-TMCM is the TMCM design based on the Synopsys DesignWare (DW) library [21]. Compared with DW-TMCM, the proposed approach achieves lower area, power, ADP and PDP for all coefficient sets. However, the delay of the proposed design

TABLE III: Synthesis results for coefficient sets S_A-S_E.

Coef. Sets	Method	Area (μm^2)	Power (μW)	Delay (ns)	ADP ($ns \cdot \mu m^2$)	PDP (fJ)
S_A	DW-TMCM	274.51	258.02	1.35	371.75	349.42
	DAG Fusion [6]	258.82	228.39	2.10	544.06	480.11
	TMCCM [8]	160.93	131.03	1.67	268.12	218.31
	TMCM-AM [9]	160.93	131.03	1.67	268.12	218.31
	Prop.	**160.93**	**131.03**	**1.67**	**268.12**	**218.31**
S_B	DW-TMCM	209.34	195.81	1.08	226.67	212.02
	TMCCM [8]	134.86	119.78	1.41	190.42	169.13
	TMCM-AM [9]	139.12	124.91	1.44	199.80	179.40
	Prop.	**135.13**	**107.97**	**1.47**	**199.04**	**159.03**
S_C	DW-TMCM	266.53	247.47	1.23	328.95	305.42
	DAG Fusion [6]	134.06	125.53	1.52	203.86	190.88
	TMCCM [8]	106.13	90.45	1.41	150.01	127.84
	TMCM-AM [9]	125.82	104.81	1.31	164.84	137.32
	Prop.	**106.13**	**90.45**	**1.41**	**150.01**	**127.84**
S_D	DW-TMCM	427.20	394.24	1.78	758.60	700.09
	TMCCM [8]	280.36	235.79	2.84	795.08	668.67
	TMCM-AM [9]	289.67	238.18	2.90	839.38	690.17
	Prop.	**252.70**	**219.08**	**2.57**	**648.95**	**562.62**
S_E	DW-TMCM	329.84	255.89	1.85	609.51	472.86
	DAG Fusion [6]	219.18	177.00	1.70	372.12	300.50
	TMCM-AM [9]	213.60	179.47	1.55	332.02	278.97
	Prop.	**202.96**	**169.97**	**1.50**	**303.50**	**254.17**

is relatively high, which is a common limitation of the shift-add-based TMCM implementations. The comparative analysis shows that, for most coefficient sets, the proposed algorithm significantly outperforms the DAG Fusion [6], especially for area and power consumption, but exhibits slightly higher delay than [8], [9] for S_B and S_C. For complex coefficient sets like S_D and S_E, the proposed method obtains a circuit architecture with more obvious hardware advantages, leading to up to 12.76%, 8.02%, 18.48% reductions in area, power and PDP. These results demonstrate that the proposed optimization framework can effectively explore the design space of arithmetic unit types, bit widths, and multiplexer configurations, leading to more hardware-efficient TMCM implementations.

V. Conclusion

In this paper, we propose a new TMCM framework based on the Ising model to enhance the resource efficiency in DSP applications. By converting the DAG fusion and selection into the optimization of the Ising model, the proposed framework significantly reduces the number of multiplexers utilized, which improves the resource utilization compared with state-of-the-art works. Moreover, compared to the heuristic algorithms and exact search techniques, the proposed Ising model-based formulation scales more effectively for large search spaces, providing greater flexibility in exploring feasible solutions. The evaluation results demonstrate that the proposed framework effectively decreases the use of adders and multiplexers and achieves significant area savings for widely used benchmark coefficient sets reported in other literature [6], [8], [9], [19], [20]. These improvements are further validated through synthesis results.

References

[1] S. Demirsoy, A. Dempster, and I. Kale, "Design guidelines for reconfigurable multiplier blocks," in *Proceedings of the International Symposium on Circuits and Systems (ISCAS)*, vol. 4, 2003, pp. IV–IV.

[2] M. Garrido and P. Malagón, "The Constant Multiplier FFT," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 1, pp. 322–335, 2021.

[3] Z. Fu and W. Ye, "A New Design of Low Hardware Cost and Low Power Programmable FIR Filters," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 71, no. 4, pp. 2314–2318, 2024.

[4] A. Dempster and M. Macleod, "Multiplication by an integer using minimum adders," in *IEE Colloquium on Mathematical Aspects of Digital Signal Processing*, 1994, pp. 11/1–11/4.

[5] F. Al-Hasani, M. P. Hayes, and A. Bainbridge-Smith, "A Common Subexpression Elimination Tree Algorithm," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 9, pp. 2389–2400, 2013.

[6] P. Tummeltshammer, J. C. Hoe, and M. Puschel, "Time-Multiplexed Multiple-Constant Multiplication," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 9, pp. 1551–1563, 2007.

[7] K. Möller, M. Kumm, M. Kleinlein, and P. Zipf, "Reconfigurable Constant Multiplication for FPGAs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 6, pp. 927–937, 2017.

[8] C. Eleftheriadis and G. Karakonstantis, "Optimal Adder-Multiplexer Co-Optimization for Time-Multiplexed Multiplierless Architectures," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 70, no. 9, pp. 3598–3611, 2023.

[9] H. Sun, M. Kumm, L. Liu, Z. Zhang, and Y. Peng, "Resource-optimized Time-multiplexed Constant Multiplication via Adjacency Matrix Modeling," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2025.

[10] M. Kumm, M. Hardieck, and P. Zipf, "Optimization of Constant Matrix Multiplication with Low Power and High Throughput," *IEEE Transactions on Computers*, vol. 66, no. 12, pp. 2072–2080, 2017.

[11] K. Möller, M. Kumm, M. Garrido, and P. Zipf, "Optimal Shift Reassignment in Reconfigurable Constant Multiplication Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 3, pp. 710–714, 2018.

[12] M. Kumm, "Optimal Constant Multiplication Using Integer Linear Programming," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 5, pp. 567–571, 2018.

[13] T. Zhang, S. Liu, H. Jiang, W. J. Gross, F. Lombardi, and J. Han, "Approximate and Stochastic Ising Machines," *IEEE Nanotechnology Magazine*, vol. 19, no. 4, pp. 44–53, 2025.

[14] A. Lucas, "Ising formulations of many NP problems," *Frontiers in Physics*, vol. 2, no. 5, 2014.

[15] A. Dempster and M. Macleod, "Multiplication by two integers using the minimum number of adders," in *2005 IEEE International Symposium on Circuits and Systems*, 2005, pp. 1814–1817 Vol. 2.

[16] O. Gustafsson, K. Johansson, and L. S. DeBrunner, "Techniques for avoiding sign-extension in multiple constant multiplication," 2009, pp. 740–743.

[17] N. Fiege, M. Kumm, and P. Zipf, "Bit-Level Optimized Constant Multiplication Using Boolean Satisfiability," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 71, no. 1, pp. 249–261, 2024.

[18] Gurobi Optimization, LLC, "Gurobi optimizer 13.0.0," https://www.gurobi.com/downloads/gurobi-software/, 2026.

[19] J. Chen and C.-H. Chang, "High-Level Synthesis Algorithm for the Design of Reconfigurable Constant Multiplier," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 12, pp. 1844–1856, 2009.

[20] L. Aksoy, P. Flores, and J. Monteiro, "Optimization of design complexity in time-multiplexed constant multiplications," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2014, pp. 1–4.

[21] Synopsys, "DesignWare Building Block IP User Guide," Synopsys, Mountain View, Technical Report J-2014.09, 2014.