EVSR: Automatic Sizing Optimization of Digital Comparator based on Extended Variable Self-Built References

Shangwei Xie[1,2*], Yi Zhan[1], Shushan Qiao[1,2]
[1]Institute of Microelectronics of the Chinese Academy of Sciences, Beijing, 100029
[2]University of Chinese Academy of Sciences, Beijing, 100049, China
Correspondence: xieshangwei@ime.ac.cn

Abstract
The emergence of threshold-based digital comparators has revolutionized mixed-signal circuit systems, notably in high-speed ADCs. These comparators generate internal reference voltage autonomously, eliminating reliance on external sources. However, different voltages require varied internal logic gate sizes, impacting reference voltage accuracy, power, latency, and area. This divergence prevents direct application of conventional digital synthesis to these comparators. In this study, EVSR (Extended Variable Self-built References) is proposed for automatic sizing optimization. A non-linear programming model is introduced to minimize internal voltage error, solved by an efficient single comparator sizing (SCS) algorithm based on integer differential evolution and Nelder-Mead mechanisms. Additionally, for multi-bit flash ADCs, the comparator is refined for a more uniform distribution of internal voltage. The optimization of both error and energy-delay product through optimal SCS and dynamic programming (OPTSCS-DP) is accomplished by the multi-comparator sizing algorithm. Experimental results confirm the SCS-based digital comparator reaches a step threshold of 10mV. Compared to the best existing solution at the same 55nm process, the proposed design reduces power consumption by 72.25% and area by 41.18%. And our proposed OPTSCS-DP demonstrates a 4 × enhancement in the Figure of Merit (FoM) compared to iterative SCS. (Code is available at https://github.com/ucas-xsw/DigitalCompapratorAlgorithm.)

Keywords: Digital comparators, built-in reference model, different evolution, Nelder-Mead method, non-linear integer programming, multi-comparator sizing algorithm

1 Introduction

Comparators have significant importance in circuit systems, particularly in the context of mixed-signal designs. Traditional analog comparators depend on the knowledge of manual design and are susceptible to fluctuations caused by the manufacturing process, which restricts their efficiency in advanced process. As circuit complexity continues to grow, the dependence on human expertise for the automation of circuit design becomes inadequate. As a result, there is an increasing inclination towards improving comparator designs to possess more digital characteristics and be amenable to synthesis. Digital comparators rely on digital logic gates, hence removing the need for analog components. The aforementioned method has many advantages, including its ability to be compatible with advanced technological nodes, its reduced operating voltage, and its capability to facilitate thorough and automated design processes.

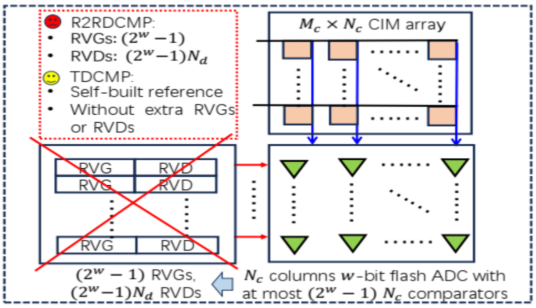

Figure 1: A traditional high speed computing-in-memory (CIM) macro [1, 2, 3] with M_c rows and N_c columns arrays, needs N_c columns flash ADC with at most $N_c \times (2^w - 1)$ comparators. Using analog R2R differential comparators(R2RDCMP) will brings extra reference voltage generators (RVG) and drivers (RVD), causing large

area and power cost, while using TDCMPs, it can be efficiently avoided.

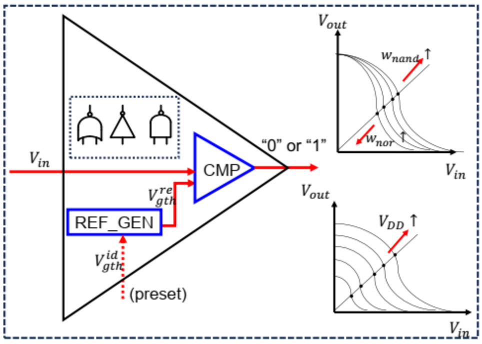

Figure 2: The digital comparator based on threshold can be equivalent to the comparator with the internal reference voltage generator.

Among digital comparators, threshold-based digital comparators (TDCMP) [4] have gained significant attention. They not only offer these aforementioned advantages but also enable internal reference voltage generation. Through continuous adjustment of transistor sizes [5, 6, 7], or discrete manipulation of logic gate numbers [8-10], the internal reference voltage in the comparator can be readily modified without external reference voltage sources. An illustrative example in Fig. 1 highlights the superior adaptability of TDCMPs. In essence, TDCMPs can be considered comparable to comparators with internal reference voltage generators, as depicted in Fig. 2. The focus of this work is the optimization of digital comparators via the use of threshold approaches, specifically using discrete standard cells. This approach aims to maximize the benefits of the complete standard cell library in order to facilitate automated design processes.

The derivation of the comparator threshold model poses a major difficulty in the design of digital comparators, as it has a notable influence on the precision and range of the internally generated reference voltage. Previous literature has explored various threshold models. Njinowa et al. [11] propose a simple yet restrictive threshold model and circuit structure. This model, based on assumptions, limits the comparator's threshold range. Khalapure et al. [12] introduce an improved digital comparator threshold model, offering more precise threshold step size adjustment by enabling the mixing of transistors in series and parallel. However, it is important to acknowledge that these concepts and their accompanying designs do include some limits. The focus of their analysis is only on variations in the internal configuration of the comparator, while disregarding the potential impact of varying amounts of standard cells. In addition, there is a disregard for the quantitative relationship between several performance measures of comparators, such as area, power consumption, latency, and the choice of standard cells. This oversight is particularly concerning considering the substantial influence that internal cell selection has on these metrics. The absence of theoretical reasons for the advantages of various systems applied hinders their scalability when addressing a bigger quantity of digital comparators. Furthermore, due to the limitation of the required threshold value of the digital comparators, existing research can only use manual sizing methods by trying and testing for internal logic gates before implementing digital synthesis, which differs from the direct synthesis process for general digital circuits. They overlook the opportunity to fully automate design through the exploration of optimization algorithm potential.

Besides, The optimization of digital comparator automatic sizing presents a significant challenge to overcome. It mirrors the conventional discrete gate sizing for combinational circuits in its nature. Many studies have tackled discrete gate sizing optimization through diverse methodologies. [13] illustrates that discrete gate sizing poses an NP-hard problem. Meanwhile, [14] introduces a convex optimization framework using 0-1 variables to solve gate sizing, employing geometric programming to address relaxation forms, ultimately achieving low-latency combination circuits. In addressing power consumption, [15] adopts the branch and bound method, specifically targeting dynamic power consumption. Others like [16-18] use dynamic programming to optimize power, area, and timing violations. Meanwhile, [19] employs a rule-guided genetic algorithm, enhancing the speed of a two-stage rail-to-rail operational amplifier. Additionally, [20] models discrete cell sizing as a minimum cost flow problem, proposing a time-driven discrete cell sizing algorithm, resulting in a 60-fold increase in sizing speed. Furthermore, [21] tackles discrete gate sizing and threshold allocation problems using an optimization algorithm based on simulated annealing to minimize leakage power. The lookup table method, as implemented in [22], reconstructs the gate delay model. This method employs modified Lagrangian relaxation to attain optimal gate sizing solutions, ultimately achieving lower delay and reduced power consumption. Additionally, improved Lagrangian-based approaches, as proposed in [23-28], have effectively addressed discrete gate sizing and threshold voltage Assignment issues observed in the ISPD 2013 gate sizing contest. These enhancements have resulted in notable improvements in both timing and runtime. Nevertheless, these algorithms are primarily tailored for linear or separable non-linear integer programming problems. Subsequent modeling reveals the automatic sizing problem for digital comparators takes the form of an inseparable constrained non-linear integer programming (CNLIP), necessitating more efficient algorithm designs.

Furthermore, prior studies mainly focused on singular objectives like power consumption or delay [14,20-22] or a linear combination of both [15,28,29]. Different from the above research, when building a single digital comparator, accuracy of the self-built voltage should be taken into consideration in addition to power consumption, area, and latency. In the case of multi-bit flash ADCs, it is necessary to take into account the nonlinear error between the comparators.

To cope with these issues, this work presents the important contributions as follows:

1) An improved threshold model is introduced, allowing for structural adjustment and discrete cell number selection. Additionally, a threshold error square index is designed to measure the discrepancy between the actual comparator threshold and the theoretical value.

2) Theoretical analyses are conducted on various performance indicators of digital comparators, including delay, power consumption, area, and EDP. Quantitative relationships between these indicators and the decision variables (the number of different standard cells) are derived.

3) A non-linear integer programming model is formulated to describe the single comparator sizing (SCS) problem. To obtain the optimal internal structure and standard cell selection scheme, an SCS algorithm is developed, employing a combination of differential evolution and discrete Nelder-Mead hybrid methods.

4) For the multiple digital comparators jointly sizing problem, it is decomposed into two stages of optimization, including optimization inside the comparators and joint optimization between comparators. Furthermore, a two-stage algorithm is proposed. In the first stage, a modified SCS algorithm is implemented for each comparator to obtain a group of approximate solution sets. In the second stage, based on the full connected network (FCN), a dynamic programming approach is utilized to search out the optimal solution path.

2 Threshold based Comparator Sizing Optimization

The proposed comparator structure is primarily composed of different types of NAND gates, inverters (INV), and NOR gates. An improved threshold model, incorporating a discrete cell vector, is initially introduced. The predetermined threshold values in comparators impose limitations on the number of transistors that can be utilized. Subsequently, significant comparator performance metrics, including delay, power consumption, and area, are derived, establishing constraints for gate sizing optimization. An integer programming-based gate sizing model for a single threshold-based comparator is then developed. Finally, an algorithm is proposed to attain the optimal gate combination. Detailed explanations of these steps for designing threshold-based digital comparators are provided in the following sections.

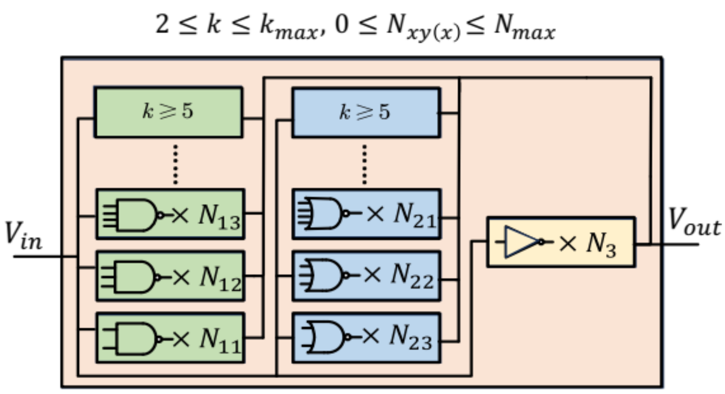

Figure 3: The improved digital comparator structure with more flexible configration of gates

2.1 Threshold based Self-built Reference Voltage Model

The limitations regarding the number and type of gates, as well as the internal structure of the digital comparators depicted in Fig. 3, are relaxed, broadening the search scope within the standard cell library for potential expansion. Defining the variable m as the maximum count of standard cell types with varying fan-in, spanning from k_1 to k_m. Two vectors, K_u and K_d, each with dimensions of $(2m+1)\times 1$, collect equivalent width expansion factors for specific gate types in the PMOS network (P-NET) and NMOS network (N-NET), respectively.

The vector x, with dimensions of $(2m+1)\times 1$, comprises m integers, from x_1^{nand} to x_m^{nand}, representing the count of NAND gates with varying fan-in from k_1 to k_m. x^{inv} denotes the quantity of inverters, while the m integers from x_1^{nor} to x_m^{nor} indicate the count of NOR gates with differing fan-in. Consequently, the $2m+1$ variables are integers, encapsulating the quantities of NAND, inverter, and NOR gates with various fan-in from the standard cell library. The pursuit is to determine the optimal x and gate sizing scheme. For a fundamental inverter gate, the overdrive voltages of the PMOS and NMOS transistors can be defined and calculated as follows table 1.

Table 1: Vector Definitions

	Vector expression	Meaning
x	$\left[\{x_j^{nand}\}_{j=1}^{j=m}, x^{inv}, \{x_j^{nor}\}_{j=1}^{j=m}\right]^T$	Gate Number
K_u	$\left[\{K_j^{-1}\}_{j=1}^{j=m}, 1, \{K_j\}_{j=1}^{j=m}\right]^T$	Width expansion (P)*
K_d	$\left[\{K_j\}_{j=1}^{j=m}, 1, \{K_j^{-1}\}_{j=1}^{j=m}\right]^T$	Width expansion (N)**
G_{pu}	$\left[\{g_{pu,j}^{nand}\}_{j=1}^{j=m}, G_{pu}^{inv}, \{g_{pu,i}nor\}_{j=1}^{j=m}\right]^T$	Conductence (P)
G_{nd}	$\left[\{g_{nd,j}^{nand}\}_{j=1}^{j=m}, G_{nd}^{inv}, \{g_{nd,i}nor\}_{j=1}^{j=m}\right]^T$	Conductence (N)
C_{pu}	$\left[\{c_{pu,j}^{nand}\}_{j=1}^{j=m}, c_{pu}^{inv}, \{c_{pu,i}nor\}_{j=1}^{j=m}\right]^T$	Capacitence (P)
C_{nd}	$\left[\{c_{nd,j}^{nand}\}_{j=1}^{j=m}, C_{nd}^{inv}, \{c_{nd,i}nor\}_{j=1}^{j=m}\right]^T$	Capacitence (N)
P_g	$\left[\{p_{g,j}^{nand}\}_{j=1}^{j=m}, p_g^{inv}, \{p_{g,i}nor\}_{j=1}^{j=m}\right]^T$	Logic gate power
P_{st}	$\left[\{p_{st,j}^{nand}\}_{j=1}^{j=m}, p_{st}^{inv}, \{p_{st,i}nor\}_{j=1}^{j=m}\right]^T$	Static power
P_{dp}	$\left[\{p_{dp,j}^{nand}\}_{j=1}^{j=m}, p_{dp}^{inv}, \{p_{dp,i}nor\}_{j=1}^{j=m}\right]^T$	Dynamic power
A_g	$\left[\{a_{g,j}^{nand}\}_{j=1}^{j=m}, a_g^{inv}, \{a_{g,i}nor\}_{j=1}^{j=m}\right]^T$	Area

[*] indicates PMOS type. [**] indicates NMOS type.

$$V_{ov,p} = V_{sg} - V_{th} = V_{dd} - V_g - V_{th}, \tag{1}$$

$$V_{ov,n} = V_{gs} - V_{th} = V_g - V_{th}. \tag{2}$$

Then we can derive the mathematical expressions of the saturation current I_{dp} in the P-NET and I_{nd} in the N-NET seperately[30]:

$$I_{dp}(x) = K_p V_{ov,p}^2 K_u^T x, \tag{3}$$

$$I_{dn}(x) = K_n V_{ov,n}^2 K_d^T x. \tag{4}$$

And the coeffients above K_p and K_n are shown:

$$K_p = \frac{1}{2}\mu_p C_{ox}\left(\frac{W}{L}\right)_p, K_n = \frac{1}{2}\mu_n C_{ox}\left(\frac{W}{L}\right)_n, \tag{5}$$

where μ_p, μ_n and C_{ox} are transistor parameters that can be seen as constants. W and L seperately represent the least width and length of the transistors in the standard cell library. In general, it is reasonable to assume that $\mu_p = \mu_n$, $\left(\frac{W}{L}\right)_P = \beta\left(\frac{W}{L}\right)_N$, so $K_p = \beta K_n$. When the input voltage V_{in} of the comparator is equal to the threshold of the comparator, the transistors in P-NET and N-NET are all saturated, so $I_{dp}(x) = I_{dn}(x)$. We have the following derivation:

$$I_{dp}(x) = I_{dn}(x), K_p = \beta K_n,$$

$$\Rightarrow \beta V_{ov,p}^2 K_u^T x = V_{ov,n}^2 K_d^T x. \tag{6}$$

We define the real comparator threshold value based on x as $V_{gth}^{re}(x)$ which is solved by (6). (7) is deduced by substituting (1) and (2) to (6) as follows:

$$V_{gth}^{re}(x) = \frac{V_{dd}-V_{th}+V_{th}\cdot m_f^{re}(x)}{m_f^{re}(x)}, \tag{7}$$

$$m_f^{re}(x) = \sqrt{\beta \cdot \frac{K_u^T x}{K_d^T x}}, \tag{8}$$

where $m_f^{re}(x)$ represents the overall equivalent width expansion factor based on x. Fig. 1 illustrates the relationship between the single comparator's ideal threshold voltage V_{gth}^{id} and β across various supply powers V_{dd} and gate sizes. When V_{dd} is reduced to 0.4V in Fig. 1, the impact of β on V_{gth}^{re} may diminish.

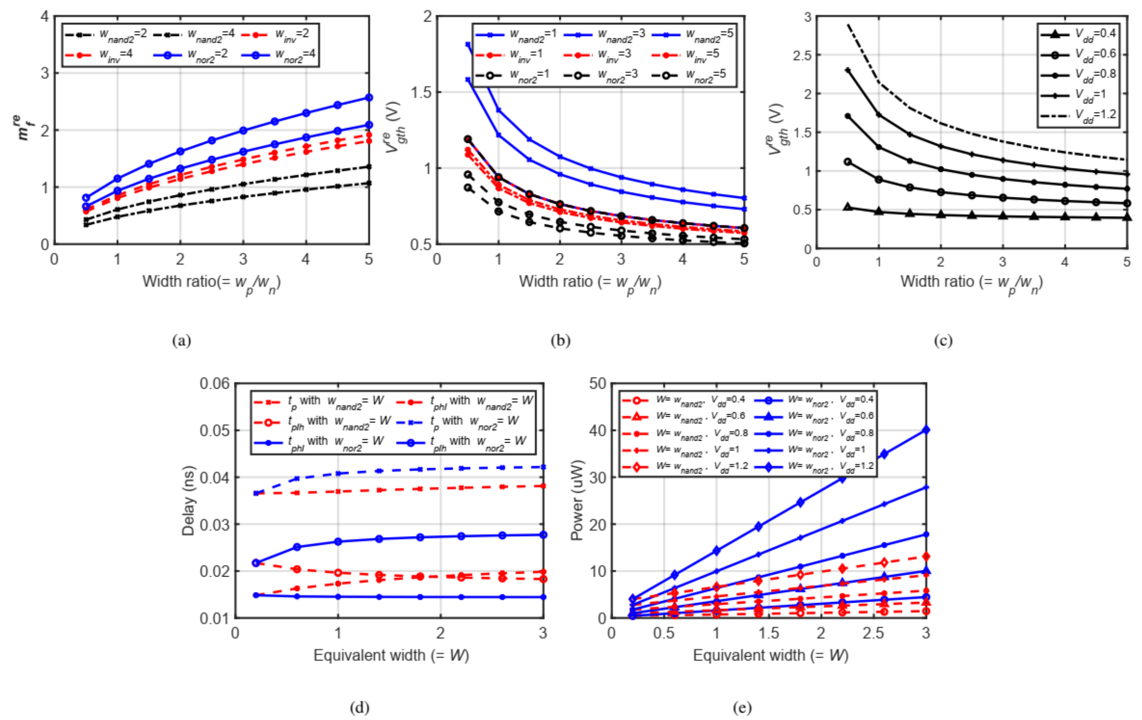

Figure 4: (a) m_f^{re} versus β with different size value of nand, nor and inverter gates (b) V_{gth}^{re} versus $\beta(w_p/w_n)$ with different supply values (c) V_{gth}^{re} versus $\beta(w_p/w_n)$ with different size value of nand, nor and inverter gates (d) Propagartion delay versus the total equivalent width of nand or nor gates (e) Power versus the total equivalent width of nand or nor gates with different supply values

Typically, the ideal comparator threshold values (also indicating the ideal reference voltage), labeled as V_{gth}^{id}, are predetermined. They serve as inputs for the threshold model to ascertain the sizing of the digital comparators. Subsequently, the conversion of (8) is conducted to yield the following form:

$$[(m_f^{id})^2 K_d^T - \beta K_u^T] x_{id} = 0, \tag{9}$$

where m_f^{id} is a constant and computed based on V_{gth}^{id}:

$$m_f^{id} = \frac{V_{dd}-V_{th}}{V_{gth}^{id}-V_{th}}. \tag{10}$$

The vector x_{id} represents a solution derived from (9). Our objective is to identify a suitable x that minimizes the difference between $V_{gth}^{re}(x)$ and the ideal comparator threshold V_{gth}^{id}. This necessitates measuring the gap between V_{gth}^{id} and $V_{gth}^{re}(x)$, referred to as the comparator threshold accuracy η. Similar to [9], we adopt an error-squared criterion to define the error loss function:

$$\eta = \eta(x) = \left[V_{gth}^{re}(x) - V_{gth}^{id}\right]^2. \tag{11}$$

A lower value of η indicates higher comparator threshold accuracy, prompting us to minimize η considerably. Nevertheless, (9) often offers an infinite array of solutions, each corresponding to distinct gate sizing schemes.

Additionally, x_{re} might not exclusively contain integer-type elements. To address this, we require supplementary performance metrics as constraints, thereby constructing an optimization model. This model aims to derive the optimal x and gate sizing scheme for a single comparator.

2.2 Propagation Delay

An essential performance metric for digital comparators is the propagation delay T_p. Our goal is to establish the relationship between T_p and x. T_p doesn't solely equate to the average of the rising time T_{plh} and falling time T_{phl} but is also proportionate to the product of the equivalent resistance R_{eq} and capacitance C_{eq}, as demonstrated in [30]:

$$T_p(x) = \frac{T_{phl}(x)+T_{plh}(x)}{2} = K_{tp} R_{eq}(x) C_{eq}(x)$$
$$= K_{tp}\left[R_{eq,n}(x) + R_{eq,p}(x)\right] C_{eq}(x), \tag{12}$$

where $R_{eq,p}$ and $R_{eq,n}$ denote the equivalent resistance of P-NET and N-NET respectively. As all gates within the comparator are connected in parallel, the computation actually involves the determination of the equivalent conductance $G_{eq,n}$ and $G_{eq,p}$, which are inverses of $R_{eq,p}$ and $R_{eq,n}$ respectively.

$$R_{eq,n}(x) = G_{eq.n}(x)^{-1}, R_{eq,p}(x) = G_{eq.p}(x)^{-1}. \tag{13}$$

$G_{eq,p}$ and $G_{eq,n}$ can be seperately expressed by the cumulative sum of each branch equivalent conductance in the P-NET and N-NET as follows:

$$G_{eq,p}(x) = G_{pu}^T x, G_{eq,n}(x) = G_{nd}^T x. \tag{14}$$

Therefore, the relation between R_{eq} and x can be further deduced below:

$$R_{eq}(x) = \left(G_{pu}^T x\right)^{-1} + (G_{nd}^T x)^{-1}, \tag{15}$$

in whcih, the value of R_{eq} is affected by the selection of the vector x. As for the equivalant capacitance C_{eq}, it can be divided to two parts: intrinsic capacitence C_{int} from the P-NET ($C_{int\ ,p}$) and N-NET ($C_{int\ ,n}$), and external capacitence which is mainly wire capacitence C_w from the P-NET ($C_{w,p}$) and N-NET ($C_{w,n}$).

$$C_{eq}(x) = C_{\text{int}}(x) + C_w = C_{eq,p}(x) + C_{eq,n}(x) \tag{16}$$
$$= C_{int,p}(x) + C_{\text{int},n}(x) + C_{w,p}(x) + C_{w,n}(x).$$

Subsequently, the computing formulas for the total equivalent capacitance in the P-NET ($C_{eq,p}$) and N-NET ($C_{eq,n}$) can be derived:

$$C_{eq,p}(x) = C_{\text{int},p}(x) + C_{w,p}(x) = C_{pu}^T x, \tag{17}$$

$$C_{eq,n}(x) = C_{\text{int},n}(x) + C_{w,n}(x) = C_{nd}^T x. \tag{18}$$

The relation between C_{eq} and x is represented concisely as:

$$C_{eq}(x) = \left(C_{pu}^T + C_{nd}^T\right)x. \tag{19}$$

Since we have known how to express R_{eq} and C_{eq}, we deduce the final formulars of T_p, T_{plh} and T_{phl} about x:

$$T_p(x) = K_{tp}\left(C_{pu}^T + C_{nd}^T\right)x\left[\left(G_{pu}^T x\right)^{-1} + (G_{nd}^T x)^{-1}\right], \tag{20}$$

$$T_{plh}(x) = K_{tp}\left(C_{pu}^T + C_{nd}^T\right)x\left(G_{pu}^T x\right)^{-1}, \tag{21}$$

$$T_{phl}(x) = K_{tp}\left(C_{pu}^T + C_{nd}^T\right)x(G_{nd}^T x)^{-1}. \tag{22}$$

2.3 Dynamic and Static Power

Power stands as another crucial metric for the digital comparator. Typically, the total power P within stems from the cumulative sum of all gate powers. The logic gate power vector P_g, with dimensions $(2m+1) \times 1$, encompasses the total power of each logic gate type with differing fan-in values sourced from the standard cell library. The power of each gate comprises three components as outlined in [30, 31]: static power, short-circuit power, and dynamic power from switching. Lower supply-voltage values ($V_{dd} < 2V_{th}$) can eliminate short-circuit power. Hence, we arrive at the following expression for the gate power vector:

$$P_g = P_{dp} + P_{st}, \tag{23}$$

$$P_{dp} = \alpha V_{dd}^2 f_{clk} C_{eq} = K_{dp} C_{eq}, \tag{24}$$

$$P_{st} = (1-\alpha)V_{dd}I_{leak}, \tag{25}$$
$$= (1-\alpha)V_{dd}e^{\frac{-q_c V_{th}}{\varsigma K_b T}}(I_{0,p}K_u + I_{0,n}K_d) \tag{26}$$
$$= K_{st,p}K_u + K_{st,n}K_d, \tag{27}$$

where the static power vector P_{st} and the switching power vector P_{dp} are further expressed as [32]:

$$I_{0,p(n)} = (\varsigma - 1)\left(\frac{W}{L}\right)_{p(n)} \mu_{p(n)} C_{ox} \left(\frac{K_b T}{q_c}\right)^2. \tag{28}$$

Here, we deduce the final mathematical relation between P and x:

$$P(x) = P_g^T x = \left(P_{dp} + P_{st}\right)^T x \tag{29}$$
$$= K_{dp}{C_{eq}}^T x + K_{st,p}{K_u}^T x + K_{st,n}{K_d}^T x.$$

2.4 Area

The comparator's area holds significance as it directly impacts manufacturing costs, because a smaller area implies reduced expenses. Let A denote the total area occupied by all gates within the comparator:

$$A(x) = A_g^T x \tag{30}$$

2.5 Problem Formulation

With the systematic derivation and analysis of the expressions for the comparator threshold vector model and other performance metrics, a comparator sizing normalization model for the threshold-based comparator is constructed:

$$2x^* = \underset{x}{\operatorname{argmin}} \quad \left[V_{gth}^{re}(x) - V_{gth}^{id}\right]^2 \tag{31}$$
$$s.t. \quad T_{phl}(x) \le \tau_{max}, \tag{32}$$
$$T_{plh}(x) \le \tau_{max}, \tag{33}$$
$$P(x) \le P_{max}, \tag{34}$$
$$A(x) \le A_{max}, \tag{35}$$
$$m, x_j^{nand}, x_j^{nor}, x^{inv} \in N_p, j \in [1, m] \tag{36}$$

where the objective is to minimize the error loss function while considering constraints derived from the performance metrics. x^* denotes the optimal solution of x that simultaneously satisfies the objective and all constraints. Constants τ_{max}, P_{max}, and A_{max} represent the extremities of the performance metrics, predetermined prior to the design process. Among the aforementioned constraints, (34) and (35) constitute linear inequalities, whereas (32) and (33) do not. The conversion of (32) and (33) into linear forms yields (37) and (38) respectively:

$$\left[K_{tp}\left(C_{pu}^T + C_{nd}^T\right) - \tau_{\max} G_{pu}^T\right]x \le 0, \tag{37}$$
$$\left[K_{tp}\left(C_{pu}^T + C_{nd}^T\right) - \tau_{\max} G_{nd}^T\right]x \le 0. \tag{38}$$

Upon observation, the entire model can be summarized into the following inseparable constrained non-linear integer programming (CNLIP) problem, as noted in [33], which is NP-hard. Therefore, the development of efficient algorithms is imperative to attain the optimal solution.

3 Differential evolution and Nelder-Mead hybrid method based Discrete Gate Sizing Algorithm for the Single Comparator

In this section, the aim is to acquire the best discrete global optimal solution. To achieve this, the evolutionary algorithm, renowned for its effectiveness in handling complex programming problems [34], is employed. Specifically, the gate sizing algorithm for the single comparator is designed using the differential evolution algorithm in conjunction with the Nelder-Mead method. The specifics of the gate sizing algorithm are now explored in the following delineation.

3.1 α-Constrained Discrete Differential Evolution (α-$CDDE$)

The proposed model is initially solved under an unconstrained condition by employing the differential evolution method with continuous relaxation to generate N non-integer solutions after N generations. This process comprises four critical operations: population initialization, mutation, crossover, and individual selection for the subsequent generation, which are introduced as follows.

3.1.1 Population Initialization

The np-th individual vector in the ge-th generation is defined as follows:

$$x_{np}^{g} = \left[\{x_{i,np,g}^{nand}\}_{i=1}^{i=m}, x_{np,g}^{inv}, \{x_{j,np,g}^{nor}\}_{j=1}^{j=m}\right]^T, \tag{39}$$

where $ge = 1,2,\dots,GE_{max}$ and $np = 1,2,\dots,NP_{max}$. Each individuals in the population are randomly initialized by following function:

$$x_{np}^{g} = \text{initialization}(x_{np}^{g}[\min], x_{np}^{g}[\max]), \tag{40}$$
$$x_{np}^{g}[l] = x_{np}^{g}[\min] + \text{rnd}[0,1] \cdot \left(x_{np}^{g}[\max] - x_{np}^{g}[\min]\right),$$

where $l = 1,2,..,2m+1$, and $x_{np}^{g}[\min], x_{np}^{g}[\max]$ separately represent the lower and upper bound of $x_{np}^{g}[l]$.

3.1.2 Mutation

Mutation denotes a change occurring at the individual level. Utilizing a ring topology [35], two individuals x_{rd1}^{g}, x_{rd2}^{g}, are selected from the neighborhood of the np-th individual. A differential linear combination is then constructed to execute the mutation:

$$v_{np}^{g} = \text{mutation}(x_{np}^{g}, x_{rd1}^{g}, x_{rd2}^{g}) \tag{41}$$
$$= x_{np}^{g} + \lambda_1\left(x_{np,opt}^{g} - x_{np}^{g}\right) + \lambda_2\left(x_{rd1}^{g} - x_{rd2}^{g}\right),$$

where $x_{np,opt}^{g}$ is the optimal individual in the neighborhood.

3.1.3 Crossover

Crossover, another transformative process occurring at the inner element level, aids in enhancing individual diversity. It is implemented by exchanging inner elements between xnp^{g} and vnp^{g}.

$$u_{np}^{g} = \text{crossover}(v_{np}^{g}, x_{np}^{g}),$$

$$u_{np}^{g}[l] = \begin{cases} v_{np}^{g}[l], & \text{if } u_0[1] \le cor, \text{or} l = l_{rand} \\ x_{np}^{g}[l], & \text{otherwise.} \end{cases} \tag{42}$$

where the crossover ratio cor determines whether the element exchanges happen.

3.1.4 Individual Selection

Following the mutation and crossover, the current optimal individuals are selected for the subsequent generation:

$$x_{np}^{g} = \text{selection}(u_{np}^{g}, x_{np}^{g}),$$

$$x_{np}^{g} = \begin{cases} u_{np}^{g}, & u_{np}^{g} \text{isbetterthan} x_{np}^{g} \\ x_{np}^{g} & \text{otherwise.} \end{cases} \tag{43}$$

The complete algorithm procedure is shown in **Algorithm 1**.

Algorithm 1 α-Constrained Discrete Differential Evolution

```
x_np^g ← initialization(x_np^g[min], x_np^g[max])
for g ← 1 to gmax do
  for np ← 1 to np_max do
    v_np^g ← mutation(x_np^g, x_rd1^g, x_rd2^g)
    u_np^g ← crossover(v_np^g, x_np^g)
    x_np^g ← selection(u_np^g, x_np^g)
  end for
end for
```

3.2 α-Constrained Discrete Nelder Mead (α-CDNM)

The Nelder-Mead method (NM) [36] is instrumental in solving the unconstrained non-linear minimization function $\text{OBJ}(*)$. In this subsection, akin to [37], the application of the discrete Nelder-Mead method [38] to solve the unconstrained non-linear integer problem is demonstrated. A N_p-dimensional space is considered, requiring $N_p + 1$ discrete test points within it. The N_p optimal solutions obtained after N_p runs of DE serve as the initial input variables for NM. NM primarily comprises four types of operations: reflection, expansion, contraction, and shrinkage. The central idea of NM revolves around iteratively finding superior test points to replace previous ones. Specifically, the reflection point p_{ref} of the centroid p_{avg} (the average of the previous N_p points) is computed. If the reflection point outperforms the current point $point_i$ $(i = 1,2,..,N_p)$, exploration along the expanded direction takes place, resulting in expansion points p_{exp}. Conversely, if the reflection point is inferior, all points are contracted towards a better direction, referred to as contraction points p_{con}. If these operations fail to yield improved test points, a shrinkage operation is employed. Refer to **Algorithm 2** for detailed algorithmic steps.

Algorithm 2 α-Constrained Discrete Nelder-Mead Method

```
repeat
  p_avg ← (1/N_p) Σ_{i=1}^{N_p} point_i
  p_ref ← p_bst + K_α K_β · sign(p_avg − p_bst)
  if obj(p_ref) < obj(p_wst) then
    p_exp ← p_bst + K_β · sign(p_avg − p_bst)
    if obj(p_exp) < obj(p_wst) then
      p_bst ← p_exp
    else
      p_bst ← p_ref
    end if
  else
    if obj(p_ref) > obj(point_i), i ≠ bst then
      if obj(p_ref) > obj(p_bst) then
        p_con ← p_bst + K_χ K_β · sign(p_avg − p_bst)
      else
        p_bst ← p_ref
        p_con ← p_bst + K_χ K_β · sign(p_avg − p_bst)
      end if
      if p_con > p_bst then
        point_i ← [(Point_i + P_wst) / 2]
      else
        p_bst ← p_con
      end if
    else
      p_bst ← p_ref
    end if
  end if
until stop condition is matched
```

The integer coefficients of reflection, expansion, and contraction, denoted as K_α, K_γ, and K_χ respectively, are involved in the operations. Additionally, the coefficient K_β signifies the distance, rounded to the next integer, between the centroid and the current best test point p_{bst} with respect to $\mathrm{OBJ}(*)$. The function $\mathrm{sign}(*)$ represents the signed symbol function.

3.3 α-Constrained Discrete Differential Evolution and Nelder Mead Method (α-CDDENM)

The differential evolution algorithm is well-suited for handling non-linear, multi-peak, and high-dimensional problems, offering a higher level of solution accuracy. However, this algorithm tends to consume considerable computational resources. Conversely, the Nelder-Mead algorithm excels in addressing high-dimensional problems with remarkable convergence speed but falls short in solution accuracy.

In a previous study [37], a combination of the two algorithms was explored. This hybrid approach utilized the differential evolution algorithm to construct the initial simplicity of the vertex, followed by employing the Nelder-Mead algorithm for further optimization. The goal was to enhance search efficiency and convergence speed. However, this method faced a trade-off: while the Nelder-Mead algorithm prioritized solution accuracy, the differential evolution algorithm focused on solution time. Consequently, the approach in [37] required extended computational time to obtain an approximate solution and failed to effectively enhance both accuracy and speed simultaneously.

In light of this, an improved alternative hybrid method is proposed. Initially, the Nelder-Mead algorithm efficiently obtains an approximate solution. Subsequently, this approximate solution serves as the initial solution for the differential evolution algorithm, leading to significantly improved accuracy within a shorter runtime. The specific flow of this hybrid algorithm is detailed in $Algorithm$3.3. Within this algorithm, f represents the objective function to be optimized, x_0 denotes the initial solution, δ indicates the step size during simplex initialization, tol signifies the convergence accuracy, maxiter represents the maximum number of iterations, N_p defines the population size, F and CR denote the scaling factor and intersection probability for the differential evolution algorithm, respectively. Additionally, S denotes the simplex vertices initialized using the Initialize-Simplex-Vertices function, while x_{nm} and $f(x_{nm})$ refer to the initial approximate solution and approximate target value obtained through the Nelder-Mead algorithm. Lastly, x^* and f^* represent the final solution obtained using the differential evolution algorithm and the corresponding target value.

Algorithm 4 α-CDDENM

Require: Objective function f, starting point x_0, step size δ, tolerance ϵ, maximum iterations M, population size N, scaling factor F, crossover rate CR

Ensure: Optimal solution x^*, optimal value $f(x^*)$

```
1: S ← Initialize-Simplex-Vertices(f, x_0, δ)
2: x_nm ← α-Constrained-Discrete-Nelder-Mead(f, S, ε, M)
3: x* ← α-Constrained-Discrete-Differential-Evolution(f, x_nm, f(x_nm), N, F, CR, M)
4: f* ← f(x*)
5: return x*, f*
```

Within the algorithmic framework, the first step involves the utilization of the Initialize-Simplex-Vertices function to initialize the simplex vertices. Subsequently, these vertices are used by the Nelder-Mead algorithm, generating an initial approximate solution and approximate target value. These outputs are then passed as inputs to the differential evolution algorithm, which iteratively optimizes to produce the final solution and target value. This approach presents a notable advantage: by leveraging the Nelder-Mead algorithm to obtain the initial approximate solution and approximate target value and employing simplex vertex initialization, the algorithm's global search capability is enhanced. Consequently, this results in significant improvements in both the efficiency and accuracy of the algorithm.

3.4 Violation Check of Constraints and Stop Condition

Upon completion of the unconstrained non-linear integer optimization, a set of candidate optimal solutions is obtained. Subsequently, in this subsection, the verification of whether these solutions violate the specified set of constraints is carried out. To streamline this process, a violation level function is defined as follows:

$$vio(x, g_i, b_i) = \begin{cases} 1 & \text{if } g_i(x) \le 0 \\ 1 - \frac{g_i(x)}{b_i}, & \text{if } 0 \le g_i(x) \le b \\ 0, & \text{otherwise.} \end{cases}$$

where each contraint $\{g_i, bi\}$ will be checked individually. If the violation happens, $vio(x, g_i, b_i)$ will be reset to 0.

Subsequently, the combination of all test results is achieved using the min operator:

$$vio(x) = \min_i \{vio(x, g_i, b_i)\}.$$

Ultimately, a comparison between all candidate solutions based on fitness $obj(*)$ and violation level $vio(x)$ is conducted, thereby selecting the optimal solution to proceed to the subsequent cycle. For any two groups x_1, obj_1, vio_1 and x_2, obj_2, vio_2, the comparison function is defined as follows:

$$cmp(x_1, x_2) = x_1 \Leftrightarrow \begin{cases} obj_1 < obj_2 & \text{if } vio_1, vio_2 \ge \alpha_s \\ obj_1 < obj_2 & \text{if } vio_1 = vio_2 \\ vio_1 \ge vio_2 & \text{otherwise.} \end{cases} \tag{44}$$

4 Multiple Comparators Sizing Optimization with Incremental Thresholds

In this section, emphasis is placed on the design of gate sizing optimization for multiple comparators with incremental thresholds. Firstly, a model is proposed, utilizing an incremental thresholds vector to cater to scenarios involving multiple comparators. Expressions related to performance metrics, including propagation delay, power, and area, are extended to vector forms. Moreover, the previous design variable x undergoes an extension to a two-dimensional matrix representation. Additionally, non-linear error analysis and computation, such as differential non-linearity (DNL) and integral non-linearity (INL), are incorporated. Addressing scenarios involving multiple comparators involves the construction of a non-linear integer programming-based gate sizing model. Finally, an algorithm is presented, aiming to enhance runtime and memory usage efficiency. Detailed explanations of these steps are provided in the subsequent sections.

Table 2: Extension of the design variable and the metrics 1.15in

	Vector expression	Dimensions	Meaning
V_{gth}^{id}	$[V_{gth,i}^{id}]_{1\le i\le 2^w-1}$	$1\times(2^w-1)$	Thesholds (I)*
V_{gth}^{re}	$[V_{gth,i}^{re}]_{1\le i\le 2^w-1}$	$1\times(2^w-1)$	Thesholds (R)**
x_{id}	$[x_{id,i}]_{1\le i\le 2^w-1}$	$(2m+1)\times(2^w-1)$	Cell size (I)
x	$[x_i]_{1\le i\le 2^w-1}$	$(2m+1)\times(2^w-1)$	Cell size (R)
$DNL(x)$	$[DNL_i]_{1\le i\le 2^w-2}$	$1\times(2^w-2)$	DNL
$INL(x)$	$[INL_i]_{1\le i\le 2^w-2}$	$1\times(2^w-2)$	INL
$T_p(x)$	$[T_{P,i}]_{1\le i\le 2^w-1}$	$1\times(2^w-1)$	Delay
$P(x)$	$[P_i]_{1\le i\le 2^w-1}$	$1\times(2^w-1)$	Power
$A(x)$	$[A_i]_{1\le i\le 2^w-1}$	$1\times(2^w-1)$	Area
$EDP(x)$	$[EDP_i]_{1\le i\le 2^w-1}$	$1\times(2^w-1)$	EDP

[*] 'I' means the ideal value. [**] 'R' means the real value.

4.1 Incremental Comparator Thresholds based Reference Model (ICTRM)

In the context of applying threshold-based digital comparators to design a w bit flash ADC, it involves dealing with a maximum of 2^w-1 comparator thresholds. This encompasses defining the step threshold voltage between adjacent comparator thresholds:

$$V_{step} = \frac{V_{gth,msb}^{id} - V_{gth,lsb}^{id}}{2^w-2} \tag{45}$$
$$= V_{gth,i}^{id} - V_{gth,i-1}^{id} (i = 2, \ldots, 2^w-1),$$

where $V_{gth,msb}^{id}$ and $V_{gth,lsb}^{id}$ are associated with the maximum and minimum values among the 2^w-1 ideal comparator thresholds, respectively. Given that all the ideal $V_{gth,i}^{id}$ values, as the self-built reference, are determined prior to the design process, it follows that V_{step} should also be regarded as a known constant.

In response to the growing number of comparators, the dimensions of both the design variable and performance metrics presented in Table 2 are expanded. Furthermore, the ideal overall width extension factor is redefined in the form of a diagonal matrix m_f^{id}:

$$m_f^{id} = \begin{bmatrix} m_{f,1}^{id} & 0 & \cdots & 0 \\ 0 & m_{f,2}^{id} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & m_{f,2^w-1}^{id}, \end{bmatrix} \tag{46}$$

Building upon the preceding definitions, the incremental threshold vector-based model is proposed:

$$[(m_f^{id})^2 K_d^T - K_u^T] x_{id} = 0_{1\times(2^w-1)}, \tag{47}$$

where the matrix $m_f{}^{id}$ can be solved by the following equation:

$$(V_{gth} - V_{th}\cdot 1_{1\times(2^w-1)}) m_f^{id} = (V_{dd} - V_{th}) 1_{1\times(2^w-1)},$$

(47) generates multiple solutions for x_{id}, and the number of solutions increases exponentially with the number of bits in the flash ADC. Therefore, determining the optimal x necessitates additional crucial performance metrics.

4.2 Extended Propagation Delay, Total Power and Total Area

For the multiple comparators, the performance metrics, such as propagation delay, power, area, and EDP, are extended to vector forms with a length of 2^w-1, as displayed in Table 2. The values for $T_{p,i}$, P_i, A_i, and EDP_i can be separately computed using expressions (20), (29), (30), and (32), respectively.

4.3 Non-linear Error

In the design of multiple comparators, two primary types of non-linear errors are Differential non-linear Error (DNL) and Integral non-linear Error (INL) [9]. We represent these errors using vectors DNL and INL, each with a length of (2^w-2), as detailed in Table 2. The elements DNL_i and INL_i are computed as follows:

$$DNL_i = DNL(x_i, x_{i-1}) = \frac{V_{gth}^{re}(x_i) - V_{gth}^{re}(x_{i-1})}{V_{step}} - 1,$$

$$INL_i = INL(x_i) = \sum_{k=1}^{i} DNL_k, \quad (48)$$

where the expression for $V_{gth}^{re}(x_i)$ is derived from (7). In the algorithm design to follow [5], our primary focus is on optimizing the Differential non-linear Error (DNL), which is the source of Integral non-linear Error (INL).

4.4 Energy-Delay Product (EDP)

Optimizing the Energy-Delay Product (EDP) provides a superior balance between high speed and low energy consumption compared to single metric-based performance optimization like minimizing delay alone. To establish the relationship between energy cost and x, the following equation is derived:

$$E(x) = f_{clk}^{-1}P(x) = f_{clk}^{-1}P_{dp}^{T}x + f_{clk}^{-1}P_{ST}^{T}x \quad (49)$$
$$= K_{dp}{}'C_{eq}^{T}x + K_{st,p}{}'K_{u}^{T}x + K_{st,n}{}'K_{d}^{T}x,$$

where $K_{dp}{}' = f_{clk}^{-1}K_{dp}$, $K_{st,p}{}' = f_{clk}^{-1}K_{st,p}{}'$, and $K_{st,n}{}' = f_{clk}^{-1}K_{st,n}{}'$. Then, combined (49) with (20), we have the relationship between EDP and x as follows:

$$EDP(x) = E(x) \cdot T_p(x) \quad (50)$$
$$= K_{dp}{}''\frac{x^T(C_{eq}C_{eq}^T)x}{G_{pu}^T x} + K_{dp}{}''\frac{x^T(C_{eq}C_{eq}^T)x}{G_{nd}^T x}$$
$$+K_{st,p}{}''\frac{x^T(K_u C_{eq}^T)x}{G_{pu}^T x} + K_{st,n}{}''\frac{x^T(K_d C_{eq}^T)x}{G_{nd}^T x}.$$

where

$$K_{dp}'' = f^{-1}K_{dp} \cdot K_{tp} \quad (51)$$
$$K_{st,p}'' = f^{-1}K_{st,p} \cdot K_{tp} \quad (52)$$
$$K_{st,n}''{}_{st,p}'' = f^{-1}K_{st,n} \cdot K_{tp}. \quad (53)$$

We need to find the optimal vector x to minimize EDP, so that the delay and energy can be both decreased.

4.5 Problem Formulation

The indicators like power and delay between two comparators have no correlation with each other. However, as per the definition of DNL, each DNL_i $(i = 2,3,\dots,2^w - 2)$ is influenced by the preceding comparator $i - 1$. If **Algorithm 3.3** is directly applied to each comparator, achieving local optimal non-linear error is possible, but global optimization is not. To avoid this narrow focus, the multiple comparator sizing problem is decomposed into a two-stage optimization process, which involves optimizing EDP within the comparator and performing joint EDP-DNL optimization between comparators.

In the initial stage, for each comparator i, the top q best solutions are selected based on a modified single comparator sizing model. These selected solutions become candidates for the subsequent optimization between comparators. Form $S_i = \{EDP_1,\dots,EDP_q | EDP_1 < EDP_2,\dots,< EDP_q, q \in \mathbb{N}^+\}$ and the solution set $\Omega_i = \{x_{i,1}, x_{i,2},\dots,x_{i,q}\}$, the modified single comparator sizing model is as follows:

$$\Omega_i^* = \underset{\Omega_i}{\operatorname{argmin}} \quad S_i \quad (54)$$
$$s.t. \quad \eta_i(x) \le \eta_{max}, \quad (55)$$
$$T_{phl,i}(x) \le \tau_{max}, \quad (56)$$
$$T_{plh,i}(x) \le \tau_{max}, \quad (57)$$
$$P_i(x) \le P_{max}, \quad (58)$$
$$A_i(x) \le A_{max}, \quad (59)$$
$$m, x_j^{nand}, x_j^{nor}, x^{inv} \in N_p, j \in [1,m], \quad (60)$$
$$i = 1,2,\dots,2^w - 1.$$

For the second stage, we propose the following normalization model:

$$x^* = \underset{x}{\operatorname{argmin}} \sum_{i=1}^{2^w-1} \gamma EDP_i + (1-\gamma)DNL_i, \quad (61)$$
$$s.t. \quad x_i \in \Omega_i^*, i = 1,2,\dots,2^w - 1, 0 < \gamma \le 1.$$

When $\gamma = 1$, solving the multiple comparator sizing problem involves running the single comparator sizing algorithm $2^w - 1$ times, which doesn't globally improve DNL. To achieve better DNL, we solve the model in cases where $1 < \gamma \le 1$.

5 Two stage multiple Comparators sizing algorithms Design

In this section, the sizing problem of multiple comparators is addressed through a two-stage algorithm. In the first stage, a modified single comparator algorithm is employed across all comparators to derive a group of approximate solution sets. For the second stage, a dynamic programming (DP) approach is utilized to solve (61) and determine the optimal solution path. DP is chosen for its efficiency in avoiding redundant calculations and effectiveness in multi-layer optimization scenarios. To aid comprehension of the proposed algorithms, a weighted, fully connected, and directed network (FCN) denoted as G is developed, featuring a virtual source and a virtual sink. The network's structure is depicted in Fig. 4.

5.1 Network Model

Fig. 4 illustrates the directed network G , which contains:

• $(2^w-1)q$ ordinary nodes (not including virtual nodes), each of which represents a candidate solution,denoted by $x_{i,j}$ $(i=1,\dots,2^w-1, j=1,\dots,q)$.

• 2^w-1 layers, which is equal to the solution set $\Omega=\{\Omega_1,\Omega_2,\dots,\Omega_{2^w-1}\}$.

• $(2^w-2)q^2$ weighted directed connects, denoted by H. The weight of the connect between any ordering pair of nodes x_{i,j_1} and x_{i+1,j_2}, is denoted by $W(x_{i-1,j_1},x_{i,j_2})=\gamma EDP(x_{i-1,j_1})+(1-\gamma)DNL(x_{i,j_1},x_{i-1,j_2})$.

• Two virtual nodes: a souce node $s\in\Omega_0$ and a sink node $d\in\Omega_{2^w}$. we define the weight of virtual connects from the source or pointed to the sink as 0: $W(s,x_1)=W(x_{2^w-1},d)=0,\ x_1\in\Omega_1,\ x_{2^w-1}\in\Omega_{2^w-1}$. They make the system become a standard network flow model.

• A set of paths based on the source node, denoted by the node set $path(s,x_{i,j_i})=\{s,x_{1,j_1},x_{2,j_2},\dots\}$, $x_{i,j_i}\in\Omega_i$. The path length is represented as $L(s,x_{i,j_i})=\sum_{k=1}^{i-1}W(x_{k,j_k},x_{k+1,j_{k+1}})$. In addition, the shortest path and path length are separately marked as $path^*_{(s,x_{i,j_i})}$ and $L^*_{(s,x_{i,j_i})}$.

Therefore, the first problem, represented by (54)~(60), involves determining the optimal selection of nodes and constructing each layer of the network. Similarly, the second problem, characterized by (61), corresponds to finding the shortest path from the source to the sink within the network. Essentially, the first problem equates to deciding how to select the nodes and construct each layer of the network. Moreover, the second problem involves identifying the shortest path from the source to the sink. Network model can be seen in Figure 5.

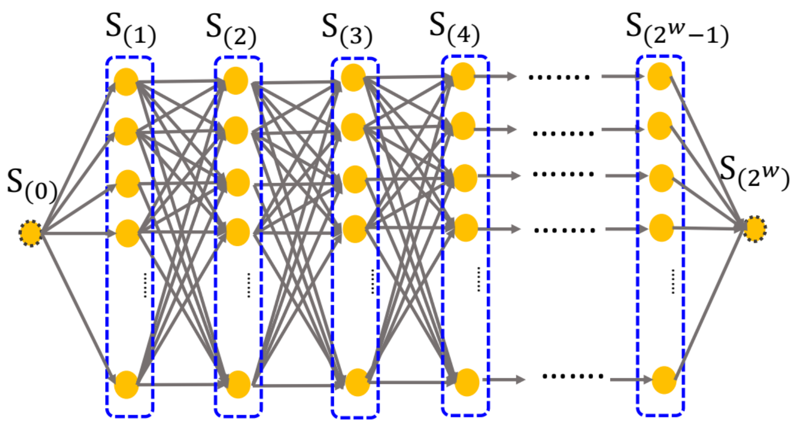

Figure 5: Network model

5.2 Layer Construction by Modified Single Comparator Sizing Algorithm

For the selection of the top q best nodes for each layer i, the following two modifications are made to **Algorithm 5.4** for each layer i:

• For DE, adjust the best individual selection function (43) to select and save top q_1 the best individuals.

• For DNM, the number of input points is expanded to a maximum of $(N+1)q_1$, which must meet the condition $(N+1)q_1\geq q$. Throughout the execution of DNM, the set Ω_i^* is updated, retaining the top q best solutions at each iteration. The algorithm stops when Ω_i^* no longer changes as the iterations progress.

5.3 Find the Shortest Path Using Forward Dynamic Programming (FDP) Method

In the second stage, the forward dynamic programming method is utilized to determine the connections between the layers in order to find the optimal path from the source to the sink. The aim is to minimize the total path length from the source to the sink. This process reveals a nesting property where the shortest path $L^*_{(s,x_{i+1,j_1})}$ $(i = 1,2,\ldots,2^w - 2)$ can be derived from $L^*_{(s,x_{i,j_2})}$ via FDP recursion:

$$L^*_{(s,x_{i+1,j_2})} = \min_{x_{i,j_1} \in B_i} \left\{ L^*_{(s,x_{i,j_1})} + W(x_{i+1,j_2}, x_{i,j_1}) \right\} \tag{62}$$

where $B_i = \left\{ x_{i,j1} \middle| x_{i,j1} \in \Omega_i, W(x_{i,j1}, x_{i+1,j2}) \neq 0 \right\}$ is the set of predecessor nodes of x_{i+1,j_2}. Due to the full connected structure, here $B_i = \Omega_i$. The initial condition of (62) is $L^*_{(s,x_{1,j_1})} = 0$.

5.4 Proposed Multiple Comparators Sizing Algorithm

Upon the earlier described process of layer construction and path finding, the EVSR has been devised for sizing multiple comparators.

Algorithm 5 OPTSCS-DP

for $i \leftarrow 1$ to $2^w - 1$ **do**
  for each layer i, execute modified SCS algorithm to generate the set $\Omega_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,q}\}$, containing top q the best solutions.
end for
$L^*_{(s,x_{1,j_1})} \leftarrow 0$
for $i \leftarrow 1$ to $2^w - 2$ **do**
  $L^*_{(s,x_{i+1,j_2})} = \min_{x_{i,j_1} \in B_i} \left\{ L^*_{(s,x_{i,j_1})} + W(x_{i+1,j_2}, x_{i,j_1}) \right\}$
end for

6 Experimental Results

In this section, simulations are conducted to validate the proposed model and algorithms using HSPICE simulator[39] and MATLAB with SMIC-55nm digital CMOS technology. The comparator, relying on standard cell (STC) and featuring a single-ended (SE) input, employs inner gates selected from the standard cells library. Utilizing six types of transistors with different thresholds, the simulation of comparator indicators necessitates defining the basic parameters. The Table 3 is obtained from the foundry's datasheet.

Table 3: Experiment Setup

Parameter	Value
Structure	SE+STC
V_{dd} (V)	0.4 ~ 1
β	1.4
Transistor type	P-type HVT, N-type HVT, P-type RVT, N-type RVT, P-type LVT, N-type LVT
V_{th} (V)	0.175,0.321,0.4477, 0.264,-0.348,-0.559
Fan-in	1~4
Gate type	NAND, NOR, INV

6.1 Case 1: Threshold based Single Comparator Design

6.1.1 Algorithm Comparision

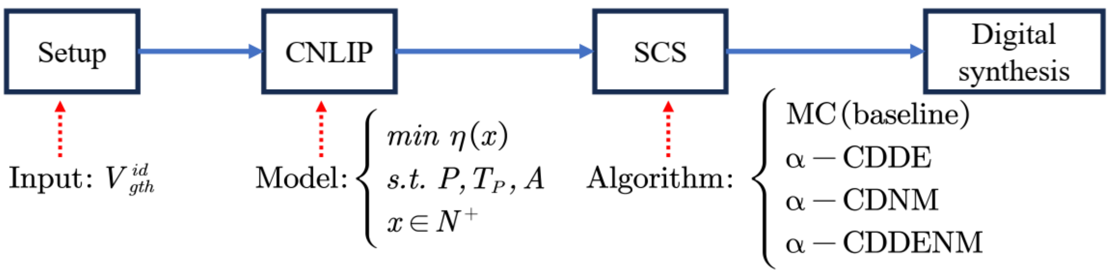

Figure 6: Design diagram of threshold based single digital comparator sizing optimization

The design diagram illustrating threshold-based single digital comparator sizing optimization is depicted in Fig. 6. The SCS algorithms are simulated using Matlab 2018b. Fig. 7 demonstrates the use of the built-in reference voltage value V_{gth}^{re} as an input parameter in the loss function. The voltage range spans from 0.2V to 0.6V with a step size of 0.1V, while four algorithms including MC (baseline), α-CDNM, α-CDDE, and α-CDDENM are employed to obtain the optimal loss value.

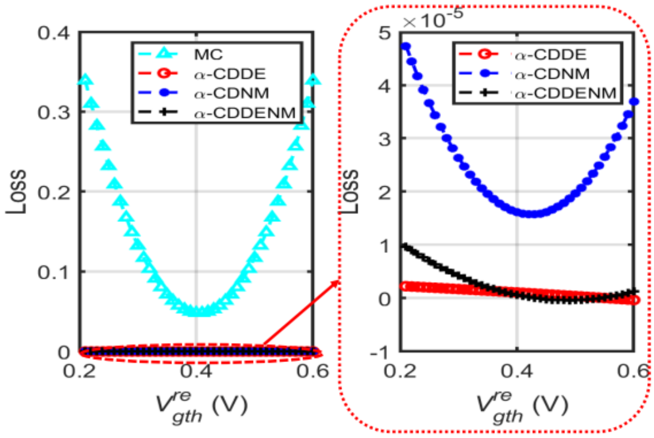

Figure 7: Loss function value versus V_{gth}^{re}

The loss value indicates the discrepancy between the desired reference voltage value required by the design, V_{gth}^{re}, and the approximate solution obtained by the algorithm, V_{gth}^{id}. The figure illustrates a quadratic relationship between the optimal loss value and the variation in V_{gth}^{re}. Specifically, at V_{gth}^{re} of 0.4V, the loss value reaches a minimum of 0.05. At this point, the absolute error between the theoretical and actual values of the built-in reference voltage reaches $\sqrt{0.05} = 0.22V$. This indicates that if the design target is 0.4V, the built-in reference voltage, V_{gth}^{id}, may be 0.18V or 0.62V, which is unacceptable in engineering applications. Conversely, the loss values obtained by the three proposed algorithms approach 0 as V_{gth}^{re} increases.

To further validate the accuracy of the loss values, we present Fig. 9, illustrating the distinct loss values for the three algorithms. In Fig. 8, the observed loss values for the three algorithms fall within the interval [0, 5e-5], corresponding to absolute errors in the range of [-7e-3, 7e-3]. Among these, α-CDDE records the lowest loss value, while α-CDDENM falls between α-CDDE and α-CDNM in terms of effectiveness.

Furthermore, Fig. 9 showcases the normalized values (ranging between 0 and 1) of the comparator propagation delay, $T_p(x)$, for the four algorithms, each reaching its respective optimal loss value as displayed in Fig. 9. When compared to the MC algorithm, the three proposed algorithms generally achieve lower delays, notably α-CDNM, which exhibits the lowest delay. Remarkably, the delay obtained by α-CDDENM closely approximates α-CDNM.

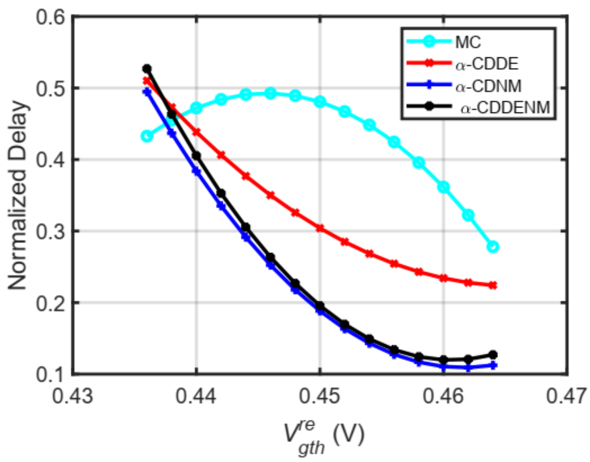

Figure 8: Normalized delay versus V_{gth}^{re}

Additionally, Fig. 9 illustrates the normalized total power consumption of the comparator for the four algorithms, each attaining its optimal loss value as shown in Fig. 10. As V_{gth}^{re} increases, the power consumption by the MC algorithm initially decreases and then increases, while for the three proposed algorithms, it initially increases and then decreases. This trend indicates clear power consumption advantages for the proposed algorithms when V_{gth}^{re} is below 0.45 or above 0.48.

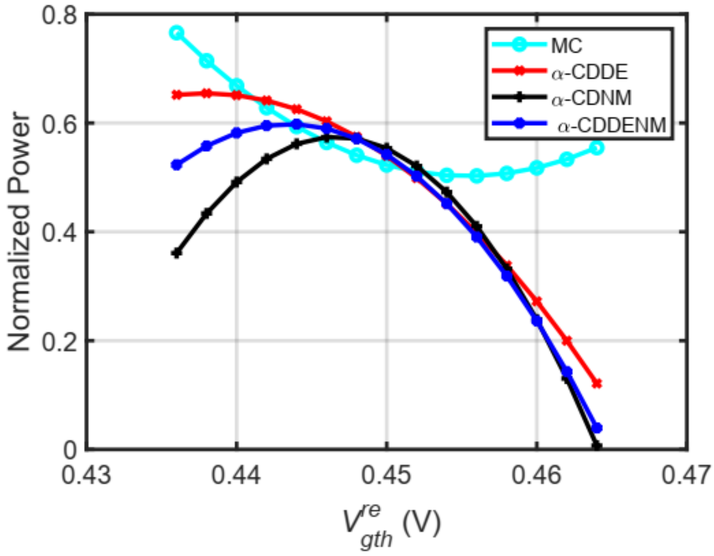

Figure 9: Normalized total power versus V_{gth}^{re}

Among the three algorithms, α-CDNM achieves the most optimal power consumption. When V_{gth}^{re} is below 0.45, the power consumption of α-CDDENM approximates the average value of α-CDNM and α-CDDE. Conversely, when V_{gth}^{re} exceeds 0.48, the power consumption of α-CDDENM closely aligns with that of α-CDNM.

6.1.2 Perfomance Comparision

Table 4 presents comparator designs referenced from literature [12, 40-43], and the design proposed in this paper. They are all based on digital standard cells. Notably, employing digital standard cells enables the full utilization of digital synthesis, enhancing design automation compared to traditional analog R2R structures, like in [40].

Unlike prior designs requiring an additional reference voltage due to the R2R structure [12, 40-43], and our design opt for single-ended methods and employ built-in reference voltages (comparator thresholds), negating the need for external reference voltage sources.

However, despite these advancements [12] largely relies on trial and error for sizing to ensure built-in reference voltage accuracy before digital synthesis due to comparator threshold constraints. In contrast, our approach achieves automation from setting internal reference voltage to digital synthesis via the design of single and multi-comparator sizing algorithms.

Normalized to 55nm and compared against existing optimal power consumption [41] (reduced by 72.25%) and area reduction in [40] (decreased by 41.18%), our design showcases the advantages: supporting digital synthesis, eliminating the need for external voltage, automating the entire process, and significantly reducing both area and power consumption.

Table 4: Comparision of different comparators

	TVLSI2014	TCASII2021	ISCAS2018	ISVLSI2017	ISCAS2022	Proposed
	[40]	[41]	[42]	[12]	[43]	
Synthesizable	No	Yes	Yes	Yes	Yes	Yes
Structure	R2R	R2R+STC	R2R+STC	SE+STC	R2R+STC	SE+STC
Eeference source	External	External	External	Built-in	External	Built-in

Built-in reference range(mV)	-	-	-	±400	-	0~600
Built-in reference accuracy(mV)	-	-	-	25	-	10
Transistor sizing	Manual	Semi-automatic	Semi-automatic	Semi-automatic	Semi-automatic	Fully automatic
Technology	180nm	180 nm	40nm	180 nm	28nm	55nm
$V_{DD,min}$ (V)	0.8	0.3~0.9	0.3~0.9	1.8	0.9	0.4~1.2
Max sampling frequency(MHz)	2400	1000		1.95	-	500
Max number of transistors	15	46	42	26	16	38~44
Input offset(mV)	7.8	28~49	28~60	0	8	0
Max Normalized Delay@55nm	0.33	9.29	269.5	-	0.53	1
Normalized Area @55nm	1.7	-	5.45	-	-	1
Max Normalized Power @55nm	219	3.67	538.83	54.34	38.2	1

6.2 Case2: Multiple Comparators Co-design with Incremental Thresholds

The diagram illustrating the optimization of multiple digital comparators is depicted in Fig. 10. Fig. 11 presents a comparative analysis between the MC-based multi-comparator sizing algorithm (MCMSC, baseline) and the proposed algorithm when applied to flash ADCs with varying bit sizes (N=2,3,4,5). All utilize the multi-comparator sizing algorithm framework detailed in this paper, integrating dynamic programming. For the joint optimization of DNL and EDP, the FoM (Figure of Merit) indicator is employed, calculated as $\mathrm{FoM} = \gamma\mathrm{DNL} + (1-\gamma)\mathrm{EDP}$ (where $\gamma = 0,0.2,0.4,0.6$). A smaller FoM value signifies superior performance. Comparatively, the multi-comparator algorithm introduced in this paper substantially diminishes the FoM value compared to MCMCS, indicating enhanced performance across different bit sizes.

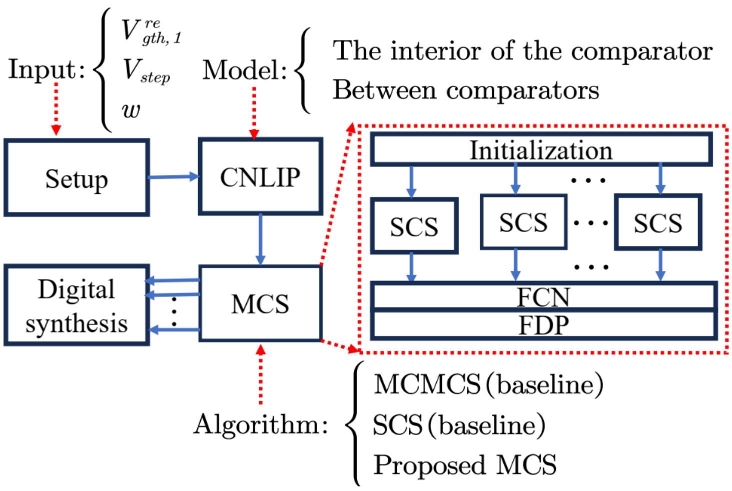

Figure 10: Design diagram of multiple digital comparators jointly sizing optimization

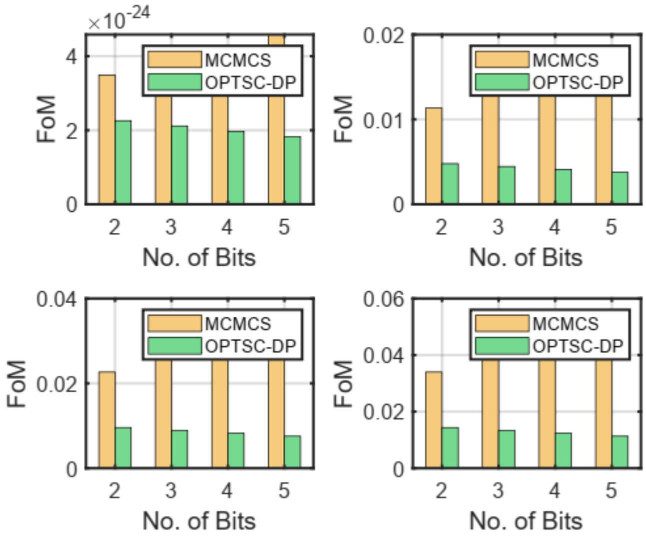

Figure 11: Compare the MC-based multi-comparator sizing algorithm MC-DP with the proposed in this paper Performance of the OPTSC-DP algorithm

Fig. 12 illustrates the FoM values across various γ values and bit numbers in two different multi-comparator design schemes:

• SCS Directly run the single comparator algorithm $2^w - 1$ times to get the design parameters of each comparator.

• OPTSC-DP. The proposed multi-comparator algorithm based on SCSdynamic programming is adopted.

As γ and the number of bits increase, the FoM value for the proposed OPTSC-DP algorithm consistently remains significantly smaller than that for SCS. This suggests that compared to individual design iterations for each comparator, the joint sizing of multiple comparators leads to lower DNL and EDP.

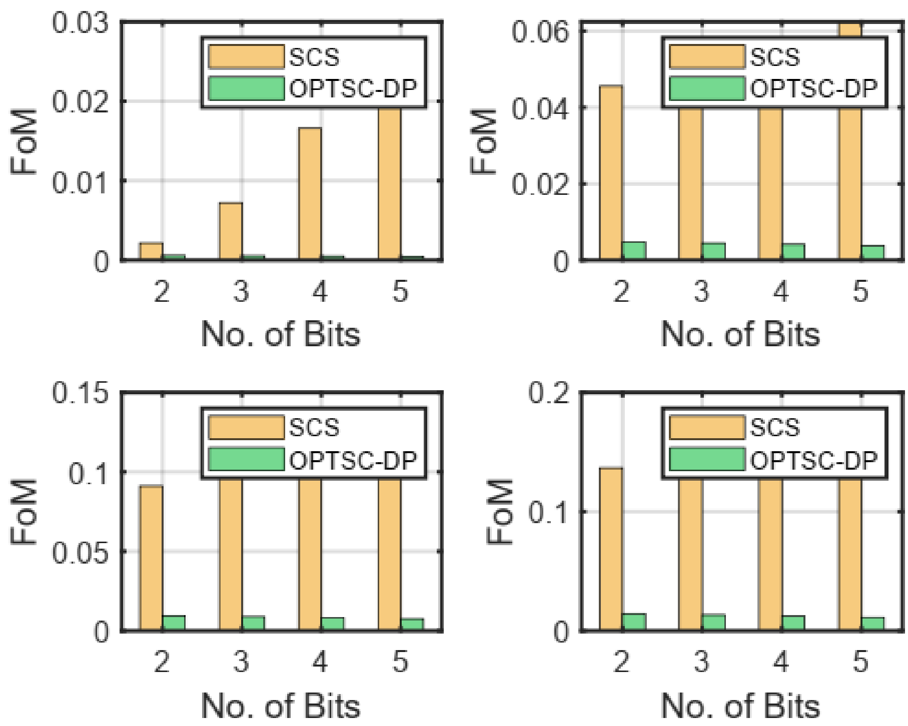

Figure 12: The FoM values versus bit Number of Comparators under two multi-comparator design schemes

7 Conclusion

In this paper, an innovative approach for designing digital comparators is presented, employing diverse standard cell sizes and manipulating thresholds to establish a built-in reference voltage model. With this model, digital comparators with various built-in reference voltages can be created by configuring logic gates with different sizes and types in parallel. The performance evaluation of the comparators includes considerations of built-in reference voltage accuracy, propagation delay, power consumption, and area.

To tackle this challenge, an optimization model for a single comparator is formulated, with constraints on propagation delay, power consumption, and area, with the ideal built-in reference voltage size as input. The objective is to maximize the accuracy of the actual built-in reference voltage. Three algorithms including α-CDNM, α-CDDE, and α-CDDENM are proposed to address this non-linear optimization model. Through simulations, these algorithms are compared with the Monte Carlo algorithm, revealing their superior performance, resulting in higher built-in reference voltage accuracy and lower power consumption, delay, and area.

In practical high-speed flash ADC circuits, different reference voltage inputs are managed by multiple comparators. To minimize the differential linear error of these comparators, a joint optimization model is developed. Dynamic programming is employed to achieve the lowest non-linear error. Experimental results demonstrate that the proposed joint design algorithm based on dynamic programming yields a lower figure of merit (FoM) compared to individually running the comparators. Moreover, the performance of the proposed multi-comparator algorithm surpasses that of the Monte Carlo algorithm combination.

DATA SHARING AGREEMENT

The datasets used and analyzed during the current study are available from the corresponding author on reasonable request.

DECLARATION OF CONFLICTING INTERESTS

The authors declared no potential conflicts of interest with respect to the research, author-ship, and publication of this article.

FUNDING

The authors received no financial support for the research.

ACKNOWLEDGMENT

The authors would like to thank the Institute of Microelectronics of the Chinese Academy of Sciences for the provision of computer resources that have significantly contributed to the research findings presented in this study (https://ime.ac.edu).

References

[1] Q. Dong, M. E. Sinangil, B. Erbagci, D. Sun, W.-S. Khwa, H.-J. Liao, Y. Wang, and J. Chang, "15.3 A 351 TOPS/W and 372.4 GOPS Compute-inMemory SRAM Macro in 7nm Fin-FET CMOS for Machine-Learning Applications," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 242–244.

[2] Z. Jiang, S. Yin, J.-S. Seo, and M. Seok, "C3SRAM: An In-Memory-Computing SRAM Macro Based on Robust Capacitive Coupling Computing Mechanism," *IEEE J. Solid-State Circuits*, vol. 55, no. 7, pp. 1888–1897, Jun, 2020.

[3] Y. Shu, H. Zhang, Q. Deng, H. Sun, and Y. Ha, "CIMC: A 603 TOPS/W In-Memory-Computing C3T Macro with Boolean/Convolutional Operation for Cryogenic Computing," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Apr. 2023, pp. 1–2.

[4] M. S. Njinowa, H. T. Bui, and F. Boyer, "Novel Threshold-Based Standard-Cell Flash ADC," *Circuits Syst.*, vol. 3, no. 1, pp. 29–34, 2012.

[5] A. Ozdemir, M. Alrizah, and K. Choi, "Optimization of Comparator Selection Algorithm for TIQ Flash ADC Using Dynamic Programming Approach," in *Proc. IEEE Comput. Society Annu. Symp. VLSI (ISVLSI)*, Jul. 2019, pp. 495–500.

[6] J. H. Park, S. Kwon, and K. Choi, "Designing Algorithm for the High Speed TIQ ADC, with Improved

Accuracy," in *Proc. Int. Syst.-on-Chip Conf. (SOCC)*, Sep. 2018, pp. 233–237.

[7] D. Lee, J. Yoo, and K. Choi, "Design Method and Automation of Comparator Generation for Flash A/D converter," in *Proc. Int. Symp. Quality Electronic Design (ISQED)*, 2002, pp. 138–142.

[8] E. Rahul, R. Siddharth, S. Vivek, M. Vasantha, and Y. Kumar, "Two-Step Flash ADC Using Standard Cell Based Flash ADCs," in *Proc. IEEE Int. Symp. Smart Electronic Syst. (iSES)*, Dec. 2019, pp. 292–295.

[9] S. Mayur, R. Siddharth, Y. Kumar, and M. Vasantha, "Design of Low Power 5-Bit Hybrid Flash ADC," in *Proc. Int. Symp. VLSI Circuits (ISVLSI)*, Jun. 016, pp. 343–348.

[10] R. Pradeep, R. Siddharth, Y. Kumar, and M. Vasantha, "Process Corner Calibration for Standard Cell Based Flash ADC," in *Proc. IEEE Int. Symp. Smart Electronic Syst. (iSES)*, Dec. 2019, pp. 195–200.

[11] M. S. Njinowa, H. T. Bui, and F.-R. Boyer, "Design of Low Power 4-Bit Flash ADC based on Standard Cells," in *Proc. IEEE 11th Int. New Circuits and Syst. Conf. (NEWCAS)*, Jun. 2013, pp. 1–4.

[12] K. Sumit, R. Siddharth, Y. Kumar, and M. Vasantha, "Design of 5-Bit Flash ADC Using Multiple Input Standard Cell Gates for Large Input Swing," in *IEEE Comput. Society Annu. Symp. VLSI (ISVLSI)*, Jul. 2017, pp. 585–588.

[13] W. Li, "Strongly NP-hard Discrete Gate Sizing Problems," in *Proc. IEEE Int. Conf. Comput. Design (ICCAD)*, 1993, pp. 468–471.

[14] S. R. Naidu, "Geometric Programming Formulation for Gate Sizing with Pipelining Constraints," in *Proc. 28th Int. Conf. on VLSI Design*, Jan. 2015, pp. 452–457.

[15] M. Rahman, H. Tennakoon, and C. Sechen, "Power Reduction via Near-Optimal Library-based Cell-Size Selection," in *Proc. 2011 Design, Autom. & Test in Europe (DATE)*, Mar. 2011, pp. 1–4.

[16] Y. Liu and J. Hu, "A New Algorithm for Simultaneous Gate Sizing and Threshold Voltage Assignment," *IEEE Trans. Comput.-Aided Design of Integr. Circuits Syst.*, vol. 29, no. 2, pp. 223–234, Feb. 2010.

[17] M. M. Ozdal, S. Burns, and J. Hu, "Gate Sizing and Device Technology Selection Algorithms for High-Performance Industrial Designs," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2011, pp. 724–731.

[18] S. Hu, M. Ketkar, and J. Hu, "Gate Sizing for Cell-Library-Based Designs," *IEEE Trans. Comput.-Aided Design of Integr. Circuits Syst.*, vol. 28, no. 6, pp. 818–825, Jun. 2009.

[19] R. Rashid and N. Nambath, "Hybrid Particle Swarm Optimization Algorithm for Area Minimization in 65 nm Technology," in *Proc. IEEE Int. Symp. Circuits and Syst. (ISCAS)*, May, 2021, pp. 1–5.

[20] H. Ren and S. Dutt, "A Network-Flow based Cell Sizing Algorithm," 2008.

[21] T. Reimann, G. Posser, G. Flach, M. Johann, and R. Reis, "Simultaneous Gate Sizing and Vt Assignment Using Fanin/Fanout Ratio and Simulated Annealing," in *Proc. IEEE Int. Symp. Circuits and Syst. (ISCAS)*, May. 2013, pp. 2549–2552.

[22] J. Xie and C. R. Chen, "Lookup Table based Discrete Gate Sizing for Delay Minimization with Modified Elmore Delay Model," in *Proc. 25th Great Lakes Symp. VLSI*, May, 2015, pp. 361–366.

[23] H. Cui, X. Luo, and Y. Wang, "Scheduling of Steelmaking-Continuous Casting Process Using Deflected Surrogate Lagrangian Relaxation Approach and DC algorithm," *Comput. & Ind. Engineering*, vol. 140, p. 106271, Feb. 2020.

[24] S. Daboul, N. Hahnle, S. Held, and U. Schorr, "Provably Fast and Near-Optimum Gate Sizing," ¨ *IEEE Trans. Comput.-Aided Design of Integr. Circuits Syst.*, vol. 37, no. 12, pp. 3163–3176, Dec. 2018.

[25] D. Mangiras and G. Dimitrakopoulos, "Incremental Lagrangian Relaxation based Discrete Gate Sizing and Threshold Voltage Assignment," in *Proc. 10th Int. Conf. Modern Circuits Syst. Technologies (MOCAST)*, Jul. 2021, pp. 1–5.

[26] H. Placido and R. Reis, "Tackling the Drawbacks of a Lagrangian Relaxation based Discrete Gate Sizing Algorithm," in *Proc. IEEE Comput. Society Annu. Symp. VLSI (ISVLSI)*, Jul. 2019, pp. 284–289.

[27] A. Sharma, D. Chinnery, T. Reimann, S. Bhardwaj, and C. Chu, "Fast Lagrangian Relaxation-based Multithreaded Gate Sizing Using Simple Timing Calibrations," *IEEE Trans. Comput.-Aided Design of Integr. Circuits Syst.*, vol. 39, no. 7, pp. 1456–1469, Jul. 2020.

[28] A. Stefanidis, D. Mangiras, C. Nicopoulos, D. Chinnery, and G. Dimitrakopoulos, "Autonomous Application of Netlist Transformations Inside Lagrangian Relaxation-Based Optimization," *IEEE Trans. Comput.-Aided Design of Integr. Circuits Syst.*, vol. 40, no. 8, pp. 1672–1686, Oct. 2021.

[29] M. M. Ozdal, S. Burns, and J. Hu, "Gate Sizing and Device Technology Selection Algorithms for High-

Performance Industrial Designs," in *Proc. IEEE/ACM Int. Conf. on Comput.-Aided Design (ICCAD)*, Nov. 2011, pp. 724–731.

[30] M. R. Jan, C. Anantha, N. Borivoje *et al.*, *Digital Integrated Circuits: a Design Perspective*. Reading, MA: Pearson, 2003.

[31] N. H. Weste and D. Harris, *CMOS VLSI Design: a Circuits System Perspective*. Pearson Education India, 2015.

[32] Y. Nasser, J. Lorandel, J. Prevotet, and M. Helard, "RTL to Transistor Level Power Modeling and Estimation Techniques for FPGA and ASIC: A Survey," *IEEE Trans. Comput.-Aided Design of Integr. Circuits Syst.*, vol. 40, no. 3, pp. 479–493, Mar. 2021.

[33] D. Li, X. Sun *et al.*, *Nonlinear integer programming*. Springer, 2006, vol. 84.

[34] B. V. Kumar, D. Oliva, and P. N. Suganthan, *Differential Evolution: From Theory to Practice*. Springer, 2022, vol. 1009.

[35] Li, Xiaodong, "Niching Without Niching Parameters: Particle Swarm Optimization Using a Ring Topology," *IEEE Trans. Evolutionary Computation*, vol. 14, no. 1, pp. 150–169, Feb. 2010.

[36] J. A. Nelder and R. Mead, "A Simplex Method for Function Minimization," *The Comput. Journal*, vol. 7, no. 4, pp. 308–313, Jan. 1965.

[37] F. Luchi and R. A. Krohling, "Differential Evolution and Nelder-mead for Constrained Non-linear Integer Optimization Problems," *Procedia Comput. Science*, vol. 55, pp. 668–677, 2015.

[38] E. Brea, "Una Extension del M ´ etodo de Nelder-Mead a Problemas de Optimizaci ´ on no Lineales Enteros Mixtos," ´ *Comput. & Ind. Engeering*, vol. 29, no. 3, pp. 163–174, Jul. 2013.

[39] *HSPICE ®User Guide: Simulation and Analysis*. Synopsys, 2010.

[40] S. Weaver, B. Hershberg, and U. Moon, "Digitally Synthesized Stochastic Flash ADC Using Only Standard Digital Cells," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 1, pp. 84–91, Jan. 2014.

[41] O. Aiello, P. Crovetti, P. Toledo, and M. Alioto, "Rail-to-Rail Dynamic Voltage Comparator Scalable Down to pW-Range Power and 0.15-V Supply," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 68, no. 7, pp. 2675–2679, Jul. 2021.

[42] O. Aiello, P. Crovetti, and M. Alioto, "Fully Synthesizable, Rail-to-Rail Dynamic Voltage Comparator for Operation down to 0.3V," *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, p. 5, 2018.

[43] M. Li, J. Wang, X. Cheng, and X. Zeng, "A Fully Synthesizable Dynamic Latched Comparator with Reduced Kickback Noise," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May. 2022, pp. 2876–2880.