applied sciences

MDPI

Article

Triple-Threshold Path-Based Static Power-Optimization Methodology (TPSPOM) for Designing SOC Applications Using 28 nm MTCMOS Technology

Peng Li [1], Shite Zhu [2], Wei Xi [1], Changbao Xu [3], Dandan Zheng [4,*] and Kai Huang [5]

[1] Digital Grid Research Institute, China Southern Power Grid, Guangzhou 510670, China
[2] College of Information Science & Electronic Engineering, Zhejiang University, Hangzhou 310027, China
[3] Electric Power Research Institute of Guizhou Power Grid Co., Ltd., Guiyang 550002, China
[4] College of Electrical Engineering, Zhejiang University, Hangzhou 310027, China
[5] School of Micro-Nano Electronics, Zhejiang University, Hangzhou 310027, China
* Correspondence: dandan_zheng@zju.edu.cn

Abstract: The threshold voltage distribution technique is an effective way to reduce the static power consumption of integrated circuits. Several gate-level-based distribution algorithms have been proposed, but the optimization effect and run time still need further optimization when applied to very large-scale integration (VLSI) designs. This paper presents a triple-threshold path-based static power optimization methodology (TPSPOM) for low-power system-on-chip. This method obtains the path weights and cell weights from paths' timing constraints and cells' delay-to-power ratios, then uses them as indexes to distribute each cell to low-threshold voltage (LVT), standard-threshold voltage (SVT), or high-threshold voltage (HVT). The experimental results based on a 28 nm circuit containing 385,781 cells show that the TPSPOM method reduces static power consumption by 15.16% more than the critical-path aware power consumption optimization methodology (CAPCOM). At the same time, run time is reduced by 96.85%.

Keywords: triple-threshold voltages; circuit optimization; design methodology; multi-threshold CMOS; static power consumption optimization

check for updates

Citation: Li, P.; Zhu, S.; Xi, W.; Xu, C.; Zheng, D.; Huang, K. Triple-Threshold Path-Based Static Power-Optimization Methodology (TPSPOM) for Designing SOC Applications Using 28 nm MTCMOS Technology. *Appl. Sci.* **2023**, *13*, 3471. https://doi.org/10.3390/app13063471

Academic Editor: Ernesto Limiti

Received: 10 December 2022
Revised: 6 March 2023
Accepted: 7 March 2023
Published: 8 March 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the development of complex semiconductors and communication technologies, the research and development of very large-scale integration (VLSI) designs have gradually emerged since the 1970s. With further process development, VLSI is mainly limited by area, speed, and power consumption [1]. Researchers have moved their focus from high-speed to low-power designs to improve portable system endurance, decrease heat dissipation issues, improve reliability, and decrease packaging complexity [2–5]. The purpose of low-power design is to reduce power consumption while maintaining performance [6].

The circuit power consumption is composed of static power consumption and dynamic power consumption. It is observed that in deep submicron technology, the leakage current contributes 50% of total power consumption [7]. Meanwhile, at the sub-threshold level, the leakage current influences static stability [8]. Therefore, reducing static power consumption is an essential part of low-power design. The sub-threshold leakage current is given by [9]

$$\begin{cases} I_{leakage} = I_o 10^{(V_{gs}-V_{th})/S} \\ S = ln(10) V_T (1 + \frac{C_d}{C_{ox}}) \end{cases} \quad (1)$$

where V_{gs} is the gate-to-source voltage, V_{th} is the threshold voltage, I_o is the drain current with $V_{th} = V_{gs}$, C_d is the depletion layer capacitance, C_{ox} is the gate-oxide capacitance, V_T is the thermal voltage, and S is the subthreshold slope.

Techniques such as multi-threshold CMOS (MTCMOS), transistor stacking [10], sleepy stack approach [11], sleepy keeper approach [12], and body bias technique [13] are used to minimize static power consumption. In these techniques, multi-threshold CMOS is very influential in reducing leakage currents [14,15]. Low-threshold voltage (LVT) devices have low latency but high static power consumption, while high-threshold voltage (HVT) devices do the opposite. According to Equation (1), the static power consumption will increase exponentially when the threshold voltage decreases linearly. Thus, when a cell is swapped from LVT to HVT, the proportion of change in static power consumption is much more significant than the proportion of change in delay [16]. Therefore, multi-threshold CMOS technology aims to reduce the proportion of LVT cells as much as possible. It means only using LVT cells on critical paths to meet the timing constraints and HVT cells on non-critical paths to reduce static power consumption. However, the complex circuit network makes it difficult to distribute the threshold voltage. In many circuit designs, the standard-threshold voltage (SVT) between LVT and HVT is also used, further increasing the threshold voltage distribution's complexity.

The threshold voltage distribution algorithms in [17–19] effectively reduce static power consumption. However, these methods require updating the timing in the static timing analysis (STA) tools after each cell replacement, which means that the number of timing updates depends directly on the circuit scale. When applied to a VLSI design, the algorithms are less time feasible. On the other hand, when considering the circuit timing, these algorithms use the number of critical paths passing through the cell to distinguish whether the cell is critical, which cannot accurately reflect the connection between the cell and the path timing.

This paper describes a triple-threshold path-based static power optimization methodology (TPSPOM). This method can effectively assign the three threshold voltages of the cell to reduce static power consumption. Its primary significance is as follows.

(1) We proposed a new way of distinguishing critical cells. We distinguish the criticality of a cell based on its delay-to-power ratio and the criticality of the paths through it. The most critical cell in the most critical path is then selected in the replacement process.
(2) Our algorithm flow avoids many operations of updating the timing and is suitable for VLSI designs.

2. Related Work

Various algorithms have been proposed for minimizing static power consumption using the dual-threshold technique at the transistor level [20–22]. However, the transistor-level analysis and assignment methodology is complex and unsuitable for current VLSI designs. Compared to the transistor stage, the gate-level circuit uses standard cells as the basic unit of the circuit, significantly reducing the complexity of the VLSI designs. As a result, the design approach for threshold voltage distribution has also evolved from the transistor level to the gate level [17–19,23,24]. The methods in [23,24] significantly reduce the number of operations by batch replacement, while the methods in [17–19] choose a node-by-node replacement for accuracy.

The gate-level dual-threshold static power-optimization methodology (GDSPOM) [17] provides a way to distribute threshold voltages at the gate level. The circuit's initial state in this method is all-HVT. This method analyzes the timing by STA [25] and then swaps the cell with the highest cost to LVT, where the cost of the cell refers to the number of critical paths through it. This method then repeats the above operation of analyzing the timing and replacing the cell until the timing meets the constraints. For example, as shown in Figure 1, there are six paths in the design, three of which are critical. Cells A and D will be swapped to LVT first since their costs are the highest.

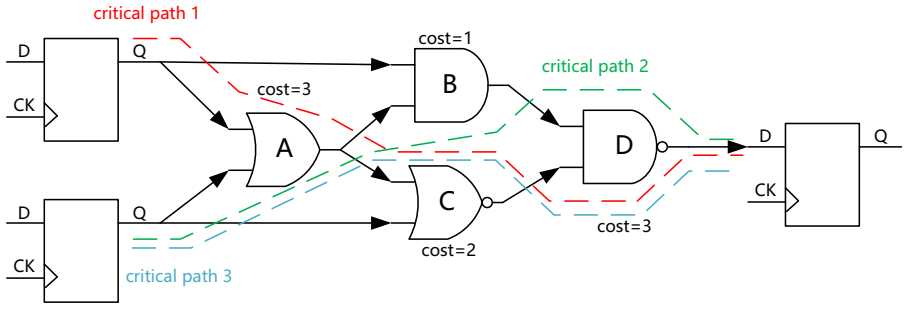


Figure 1. A gate-level dual-threshold static power-optimization methodology (GDSPOM) cell-swapping example.

The cell-based leakage power reduction priority (CBLPRP) [18] optimization methodology views the static power consumption difference Δp as a more effective indicator than the number of critical paths through the cell. The circuit's initial state in this method is all-LVT. This method calculates each cell's static power consumption difference Δp between LVT and HVT states, then swaps the cell with the highest Δp to HVT. If the timing does not meet the constraints after replacement, this method will swap the cell back to LVT. The above swapping operation continues until all cells are accessed. In Figure 2, for example, the cells are accessible in the order of $ACBD$ based on the value of Δp.

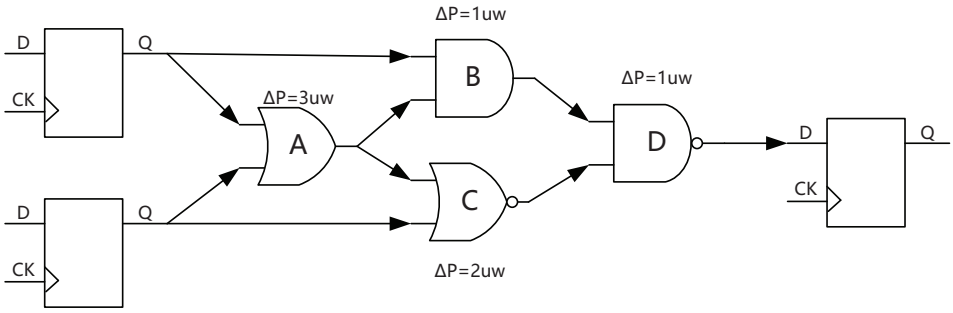


Figure 2. A cell-based leakage power reduction priority (CBLPRP) cell-swapping example.

Critical-path aware power-consumption optimization methodology (CAPCOM) [19] considers that cells' delay, static power consumption, and the number of critical paths through the cells need to be considered. This method divides cells into critical cells and non-critical cells. For non-critical cells, this method calculates the delay difference Δd and the static power difference Δp for each cell between LVT and HVT states. For critical cells, this method calculates the number of critical paths N through each cell additionally. The initial state of the circuit is all-LVT, and the non-critical cells are first accessed in descending order of the power-to-delay ratio $\Delta p / \Delta d$. Then, access each critical cell in descending order of $\Delta p / (\Delta d * N)$ values. As with CBLPRP, if the swapped cell does not meet the timing constraint, swap it back and loop until all cells are accessed. This method can be applied to triple-threshold voltage distribution: prioritize whether the cell can be swapped from LVT to HVT; after accessing all cells, consider whether the remaining LVT cells can be swapped to SVT.

However, these methods have the following drawbacks:

- These methods need to update the timing every time they swap a cell, which can lead to significant runtime in VLSI designs. The operation to update the timing is performed in STA tools, which calls the composite current source (CCS) timing library to recalculate the cell's delay and update the path timing. Calculating delay using the CCS library is time-consuming [26], and updating the paths requires traversing

many paths. For the same number of cells, replacing one cell at a time takes longer than replacing cells in a batch.
- The VLSI designs have a large order of magnitude of paths, so calculating the number of critical paths is also time-consuming. In GDSPOM, the critical path number must be updated after each cell replacement, while CAPCOM only calculates the critical path number once in the initial stage. Regarding time complexity, CAPCOM is much less than GDSPOM, while similar to CBLPRP.
- Targeted replacement is more effective if the timing constraints for each path are considered rather than just the number of critical paths.

3. Theoretical Analysis

This paper describes a triple-threshold path-based static power optimization methodology (TPSPOM). This method concentrates on the specific time constraints of each path and considers how each cell affects the overall circuit and the given path. At the same time, this method saves run time by reducing the number of update times. For designs containing hundreds of thousands of cells, the TPSPOM method can efficiently distribute triple-threshold voltages.

The main steps of the method are as follows:

(1) A simplified path collection is used instead of all paths when considering the circuit's timing.
(2) Obtain the timing constraint inequalities for each path based on the information of cells and paths.
(3) Quantify the criticality of paths and use them as path weights.
(4) Get cell weight based on the weights of all paths associated with the cell and the delay-to-power ratio of the cell.
(5) The initial state of the circuit is all-HVT. Path weights are used as the first order and cell weights as the second order to select the cells needing replacement. After each replacement, the path and cell weights are updated using approximate calculations. Repeat swapping cells until all the paths in the path collection meet the timing constraints. This stage saves time by avoiding updating the timing within the STA tools.
(6) Apply the assignment results obtained from step 5 in the STA tools. If there are still timing violations, expand the path collection and perform step 5 again until the STA results meet the constraints.

3.1. The Reason and Way of Simplifying Path Collection

The critical paths reflect the timing state of the circuit, and the information provided by the critical paths is an important reference for threshold voltage distribution. However, the order of magnitude of the paths in the mesh structure is so large that traversing all critical paths is time-consuming or infeasible. For example, as shown in Figure 3, suppose that the structure in the dotted box is repeated 15 times. Currently, the maximum logic depth (the number of cells on one path) is $(5 + 15 * 2) = 35$, which is common in VLSI designs. When the structure inside the dashed box is not considered, the paths through A and B are 3 respectively. For each repetition of the structure inside the dashed box, the number of paths is multiplied by 2. Thus the number of paths is $(2^{15} * (3 + 3)) = 196,608$, while the cell number is $(7 + 15 * 3) = 52$. It means that the order of magnitude of the paths may be much larger than that of cells.

The high similarity of these paths means they provide similar information, and access to all of them means redundant information at a high cost. In order to reduce the computational complexity, the proposed method obtains only the worst timing path through each pin. In the case of Figure 3, the maximum number of paths counted in this way is 68, which is the same order of magnitude as the pin. Using simplified path collection lacks completeness, and ways to expand the path collection to compensate for this drawback are described later in Section 3.5.

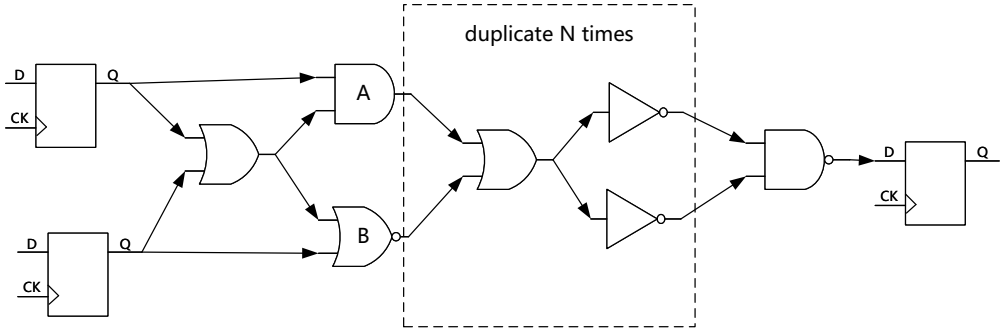


Figure 3. A path number example.

3.2. Obtaining Path-Based Timing Constraints

The CAPCOM method considers the cell delay without considering the path and uses the average delay of the cell as a parameter. However, the specific impact of the cell on a path needs to be considered. For example, in Figure 4, the delays of timing arc *A-to-O* and timing arc *B-to-O* are different. A significant difference exists between the cell delay on the path through pin *A* and the path through pin *B*.

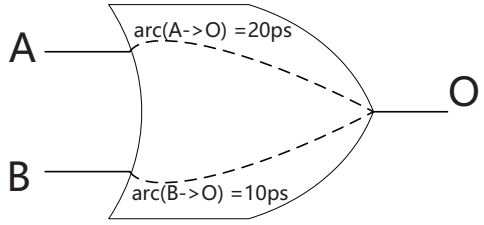


Figure 4. An example of or-gate timing arcs.

On the other hand, CAPCOM only distinguishes paths into critical paths and non-critical paths. The method in this paper can obtain more accurate assignment results from distributing threshold voltages against path constraints. The path constraint is given by

$$\sum_{j \in CIP_i} (\alpha_j * d_{C_{(i,j)}} + d_{N_{(i,j)}}) + t_{s_i} \leq t_{r_i} \tag{2}$$

where i refers to a particular path, CIP_i is the set of cells in path i, α_j is the timing derate of the cell j, $d_{C_{(i,j)}}$ is cell j's delay in the path i, $d_{N_{(i,j)}}$ is the net delay which connected to cell j's output pin in path i, t_s is the library setup time, t_r is the required time.

3.3. Path Weight

The timing slack of the path is given by

$$t_{slack_i} = t_{r_i} - \sum_{j \in CIP_i} (\alpha_j * d_{C_{(i,j)}} + d_{N_{(i,j)}}) - t_{s_i} \tag{3}$$

where t_{slack} is the timing slack of the path. When t_{slack} is not lower than 0, the timing of the path is met, and the opposite is not met.

Then we propose the concept of repairable timing, which means the total timing gain we can obtain when converting the path from all-HVT to all-LVT. The equation of the repairable timing is given by

$$t_{repairable_i} = \sum_{j \in CIP_i} (\alpha_{H_j} * d_{CH_{(i,j)}} + d_{NH_{(i,j)}} - \alpha_{L_j} * d_{CL_{(i,j)}} - d_{NL_{(i,j)}}) \tag{4}$$

where $t_{repairable}$ is the repairable timing of the path, α_{H_j}, $d_{CH_{(i,j)}}$, $d_{NH_{(i,j)}}$ is obtained when cell j is in HVT state, α_{L_j}, $d_{CL_{(i,j)}}$, $d_{NL_{(i,j)}}$ is obtained when cell j is in LVT state.

Before distributing the threshold voltage, the circuit meets the timing at a given frequency. It means that the timing of the path will meet the constraint if the path is all-LVT. Thus, the sum of $t_{repairable}$ and t_{slack} will not exceed 0. The path weight R_i is then given by

$$R_i = -\frac{t_{slack_i}}{t_{repairable_i}}, R_i \leq 1 \tag{5}$$

When R_i is less than 0, it indicates that path i meets the timing. The closer R_i is to 1, the higher the proportion of LVT required within the path.

When a cell is replaced, the path weight will be updated. The equation is given by

$$\begin{gathered} U_i = -\frac{t_{slack_i} + (\alpha_{H_n} * d_{CH_{(i,n)}} + d_{NH_{(i,n)}} - \alpha_{L_n} * d_{CL_{(i,n)}} - d_{NL_{(i,n)}})}{t_{repairable_i} - (\alpha_{H_n} * d_{CH_{(i,n)}} + d_{NH_{(i,n)}} - \alpha_{L_n} * d_{CL_{(i,n)}} - d_{NL_{(i,n)}})} \\ R_i^{'} = \begin{cases} 0 & \text{if } U_i < 0 \\ U_i & \text{if } U_i \geq 0 \end{cases} \end{gathered} \tag{6}$$

where n refers to the cell swapped from HVT to LVT. The value of $R_i^{'}$ becomes smaller after the update, which indicates that the probability of the remaining unreplaced cells being replaced with LVT decreases. When all paths' weight R does not exceed 0, all paths in the path collection have met the timing.

The larger the value of R, the more significant the proportion of LVT cells required by this path and the higher the probability of cells being replaced by LVT cells. If the low-probability cell is swapped first, the high-probability cell may still need to be swapped because the path timing constraint where it is located is tight. It can cause redundant replacements. Conversely, replacing high-probability cells will make many low-probability cells no longer need to be swapped. Therefore, it is necessary to prioritize the replacement of cells on paths with larger weights. Thus, in this paper, the algorithm prioritizes traversing the paths with larger R and then selects a cell on it for replacement and subsequent update t_{slack}, $t_{repairable}$, and R, after which the above operation is iterated until all paths meet timing constraints.

3.4. Cell Weight

When selecting a cell to be replaced within the path, we consider the cell's delay-to-power ratio and the timing constraints of the paths through it. The weight of the cell is obtained from the path collection and is given by

$$W_j = \sum_{i \in PTC_j} \frac{(\alpha_{H_j} * d_{CH_{(i,j)}} + d_{NH_{(i,j)}} - \alpha_{L_j} * d_{CL_{(i,j)}} - d_{NL_{(i,j)}}) * R_i}{p_{L_j} - p_{H_j}}, R_i > 0 \tag{7}$$

where j refers to a cell, PTC is the set of paths through the cell j, and p is the static power consumption of the cell j. The cell's weight is based on its delay and static power consumption and depends on the criticality of the paths through it.

When a particular path m through the cell j changes, cell j's weight will decrease accordingly. It is shown in Equation (8), where $R_m^{'}$ is obtained from Equation (6).

$$W_j^{'} = W_j + \frac{(\alpha_{H_j} * d_{CH_{(m,j)}} + d_{NH_{(m,j)}} - \alpha_{L_j} * d_{CL_{(m,j)}} - d_{NL_{(m,j)}})}{p_{L_j} - p_{H_j}} * (R_m^{'} - R_m) \tag{8}$$

By updating the weights, we can find the cell in the given path with the most significant impact on the current timing.

3.5. Reasons for Expanding Path Collection

In this paper, the algorithm finds the critical path by R and then finds the most critical cell on this path by W. After that, update t_{slack}, $t_{repairable}$, R, W, and repeat the above operation until the slack of all paths is greater than 0.

However, applying the obtained results to STA may not fully meet the timing constraints. There are two main reasons.

The first reason is that we use approximate calculations to get the delay of the cell. The timing arc delay of the HVT cells is obtained in the all-HVT case. The same goes for LVT cells. Therefore, there is a small gap between the delay in the calculation and the delay after applying STA, which may result in paths with minor violations remaining in the path collection.

Another reason is that the original path collection collects the most critical path of each pin, and some sub-critical paths still need to be collected. After meeting the constraints of the paths in the original path collection, some leftover sub-critical paths will still not meet the constraints.

In this case, the path collection needs to be supplemented. The legacy sub-critical paths are added to the original path collection, and the state of the sub-critical paths must be changed to all-HVT. The above algorithm is then re-run in the all-HVT state based on the new path collection. Repeat until the result of the threshold voltage distribution meets the timing constraints within the STA.

3.6. Advantages of the Method

3.6.1. Effectively Reduce Runtime

For single-cell replacement, updating the timing in the STA tools is replaced by updating the path weights and cell weights. The delay information of the cell is obtained at the initial stage, and the calculation method using CCS is replaced by an approximate calculation when updating the cell delay. The approximate calculation may bring two cases: (1) a small amount of redundant replacement slightly increase static power consumption; (2) a small amount of path timing is incompletely repaired, which can be solved by adding paths to the path collection. However, the approximate computation can substantially improve the algorithm's feasibility and reduce the operating cost.

Another way to improve the algorithm's feasibility is to use some of the most critical paths instead of all as constraints. The disadvantage is the existence of sub-critical paths that are not considered, which can also be solved by expanding the path collection.

3.6.2. More Accurate Swapping Strategy

The CAPCOM method uses the delay-to-power ratio as the cell weight, leading to the optimal solution if the optimization object has only one path. Considering that the object is the overall timing (all paths), the CAPCOM method multiplies the delay-to-power ratio by the number of critical paths to represent the impact of the cell on the overall timing.

The advantage of the algorithm in this paper is that, compared with previous algorithms, the impact of cells on timing is considered more precisely by using specific path constraints rather than simply dividing them into critical or non-critical paths. The traversal is divided into two stages. The first stage obtains the path with the largest weight. The larger the path weight, the more significant the proportion of LVTs needed within the path, and the greater the probability that cells within this path will be replaced with LVTs. Prioritizing the cells with higher replacement probability can effectively avoid redundant replacements. The second stage is to obtain the cell with the largest weight within the path. The cell weight depends on the path weight and the delay-to-power ratio, which reflects the cell's ability to influence the overall timing in more detail.

3.6.3. Effectively Compensate for the Impact of Simplification

The path collection is expanded to compensate for the little differences due to approximate calculations and the legacy timing problems due to simplifying paths. After bulk

replacement in STA tools, add the legacy path constraints to the original path collection. Loop until the result of STA shows that the timing constraints have been met. It effectively ensures the convergence of the running time and makes the operation of simplifying paths feasible.

4. Algorithm Flow

4.1. Basic Flow

As shown in Figure 5, the flow of TPSPOM can be divided into three parts.

- **Part 1**: Initial data.
  Convert the circuit to all-HVT/all-SVT/all-LVT states separately, perform STA, and obtain the timing arc, derate, and power consumption information. Then obtain the initial path collection with the circuit at all-HVT.
- **Part 2**: Sub algorithm.
  Obtain the result of the threshold voltage assignment based on the given path collection. The first path collection is provided in **Part 1**, and the subsequent one is in **Part 3**.
- **Part 3**: Top algorithm.
  Apply the threshold voltage assignment result provided by **Part 2**, and perform STA. If the STA timing results do not meet the timing constraints, add new paths to the original path collection. Then the new path collection is provided to **Part 2**, repeat until the results of STA meet timing constraints.

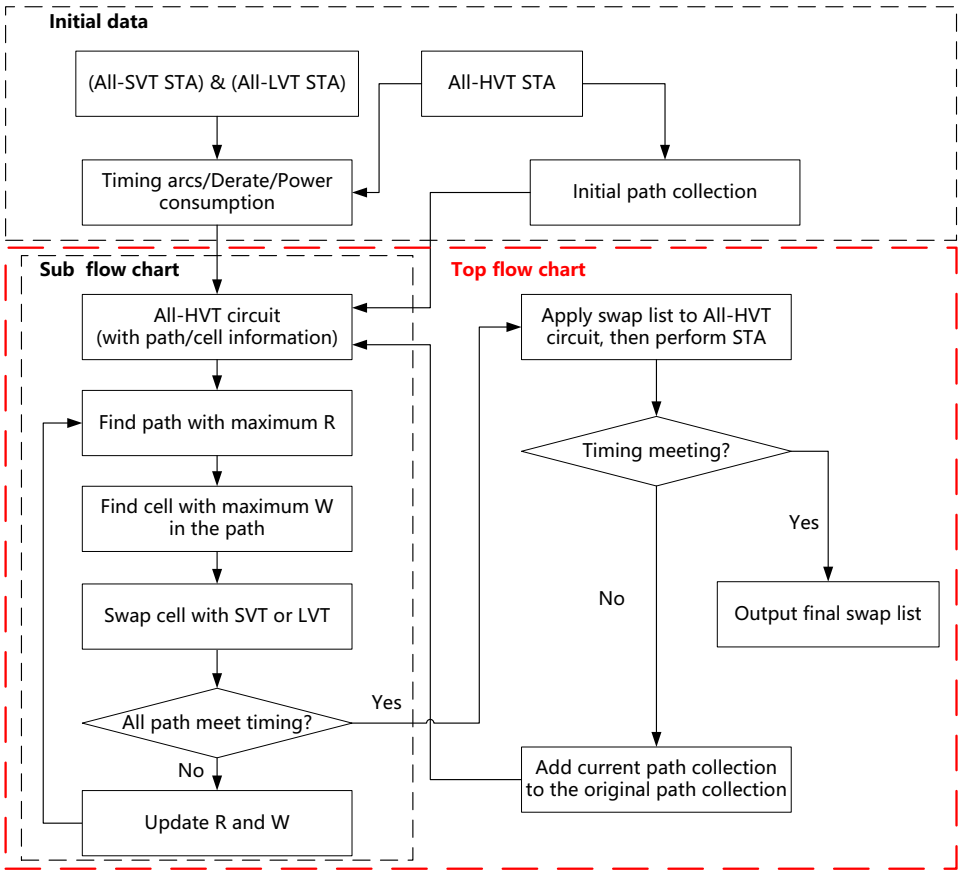


Figure 5. Flow diagram of triple-threshold path-based static power optimization methodology (TPSPOM).

The details of the steps of the above process are implemented in Algorithm 1 (sub algorithm) and Algorithm 2 (top algorithm).

When the path collection is given, the result of the assignment of cell thresholds is obtained according to the Algorithm 1 as follows.

- Step 1: Initialize data.
  According to Equations (2)–(5) and (7), get t_{slack_i}, $t_{repairable_i}$ and R_i of each path i and get WL_j, WS_j of each cell j. WS_j denotes the weight brought by the replacement of the cell from HVT to SVT, and WL_j denotes the weight brought by the replacement of the cell from SVT to LVT. For the same cell, WS_j is greater than WL_j, according to Equation (1). Swapping a cell to LVT is considered only after it has been swapped to SVT. W_j equal WS_j when cell is in at HVT, while equal WL_j when in SVT and equal 0 when in LVT.
- Step 2: Iterate to find replacement cells.
  Find the most critical path m with the largest R_m, then find the cell n with the largest weight W_n on the path. After that, swapping the cell n from HVT to SVT or SVT to LVT.
- Step 3: Update data during iteration.
  When a cell is replaced, update all paths through it, and update the cells' weight on those paths.

Algorithm 2 calls Algorithm 1 and keeps updating the path collection to find the optimal solution until the STA results meet the timing constraints.

Algorithm 1 Sub triple-threshold voltage distribution algorithm

Input: path collection C, static power consumption P, timing arcs, the OCV derate α
Output: sub cell swap list

Initialize each path i's t_{slack_i}, $t_{repairable_i}$ and R_i from collection C
Collect the replaceable cells on each path i as collection CIP_i
Collect the paths through each cell j as collection PTC_j
Initialize each cell j's weight WS_j (HVT-to-SVT) and WL_j (SVT-to-LVT)
Initialize the state of all cells to 0
When cell j's state is 0, $W_j = WS_j$; when state is 1, $W_j = WL_j$; when state is 2, $W_j = 0$
repeat
  Select the path m with the largest R_m value
  Select the cell n with the largest W_n value in CIP_m
  if cell's state = 0 **then**
    Swap cell n from HVT to SVT, remark cell's state as 1
  else if cell's state = 1 **then**
    Swap cell n from SVT to LVT, remark cell's state as 2
  end if
  for each path x in PTC_n **do**
    Update t_{slack_x}, $t_{repairable_x}$, R_x with Equations (3), (4) and (6)
    for each cell y in CIP_x **do**
      Update WS_y, WL_y with Equation (8)
    end for
  end for
until all paths' $t_{slack} \geq 0$
Return sub cell swap list

Algorithm 2 Top triple-threshold voltage distribution algorithm

Input: Static timing analysis results
Output: Final cell-swapping list
1: Initialize the circuit state to all-HVT
2: Initialize the path collection C
3: **repeat**
4: Provide path collection C to Algorithm 1
5: Obtain the sub cell-swapping list from Algorithm 1
6: Obtain the new path collection $C^{'}$ after applying the sub cell swapping list in STA tools
7: Add the new path collection $C^{'}$ to the original path collection C
8: Restore the circuit state to all-HVT
9: **until** STA results meet the timing constraints
10: **Return** Final cell-swapping list

4.2. Optimize the Sub Algorithm

We then will discuss the time feasibility of Algorithm 1. Calculate the time complexity of Algorithm 1. Assume that the number of swapped cells is A, the number of paths in path collection is B, and the maximum number of cells on a path is C.

The time complexity of the initialization phase is $O(B * C)$.

In the algorithm, we need to retrieve the most critical path before each cell replacement, i.e., find the path with the worst timing among all the paths that have not yet met the timing constraints. With the execution of the algorithm, the number of paths that do not meet the timing constraints gradually decreases. Therefore, the number of paths that need to be traversed per replacement is also decreasing. Figure 6 shows the relationship between the number of paths to be traversed and the current number of replaced cells. The experimental circuit in this figure is the same as that in Section 5. In this figure, the experiment was performed on a CPU circuit containing 385781 cells and triple-threshold voltage. The experiment was to execute Algorithm 1 once based on the initial path collection. The experimental results show that the number of paths to be traversed decreases approximately linearly as the number of replaced cells increases, so the time complexity of finding the worst path is $O(A * B)$.

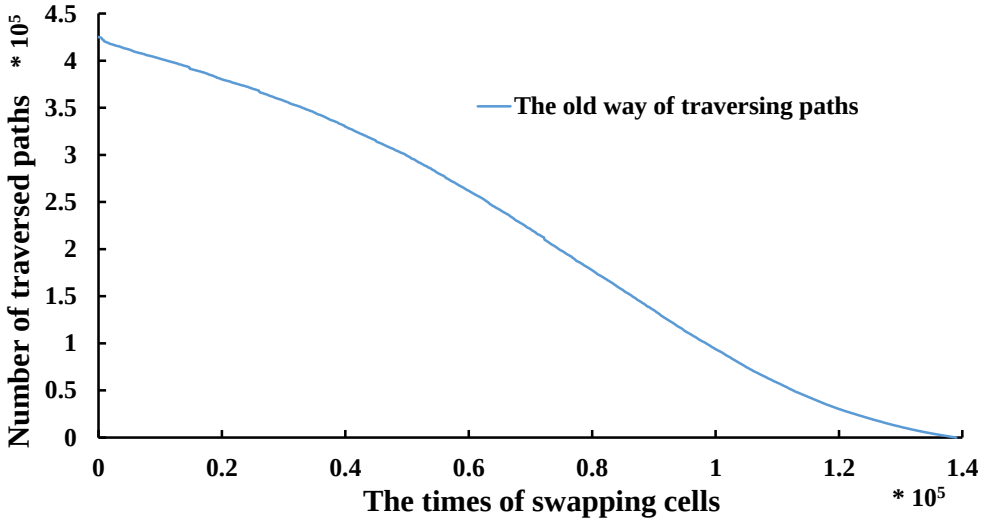


Figure 6. The relationship between the number of traversed paths and the times of swapping cells.

The time complexity of finding the cell with the highest weight in the worst path is $O(A * C)$, while that of swapping cells is $O(A)$. The worst case is that every cell of every path is replaced. Thus the time complexity of updating the path weights is $O(B * C)$.

A change in each path causes the weights of the C cells to be updated. Thus the time complexity of updating the cell weights is $O(B * C^2)$.

In the actual circuit design, the logical depth C of the path is much lesser in order of magnitude than the number of replaced cells A and the number of traversed paths B, which can be considered as a constant when calculating the time complexity. Moreover, A and B can be considered in the same order of magnitude. Thus, the worst time complexity obtained is $O(B^2)$ of finding the worst path.

Then we update the sub-algorithm, which improves the running time in two ways while guaranteeing the same execution results.

4.2.1. Optimize the Way to Update Cells' Weight

Weighting aims to find the most critical cell in a given path. The original approach was to replace a cell and update the weights of all associated cells. Therefore, we propose a new updating strategy that can effectively reduce the number of updating weights.

First, we do not update the cell weight immediately after the cell is replaced. At this point, the weights of all the cells in the given path are not updated. Moreover, there is a fact that the weight of the updated cell will not exceed what it was when it was not updated. Therefore, we only need to obtain a cell that satisfies the following conditions: (1) its weight has been updated; (2) its weight is greater than the weight of all other cells in the path (regardless of whether the weight of other cells is updated).

To this end, the following update operations are taken: (1) to obtain the cell with the highest weight in the path. (2) If the weight of this cell is not updated, update it and return to step (1); otherwise, output this cell.

Figure 7 shows an example. In step (a), we obtain cell A with the highest weight in the path. In step (b), we judge that the weight of cell A needs to be updated, so we update it and find a new cell F with the largest weight. In step (c), we judge that the weight of cell F still needs to be updated; after updating it, we find that cell F has met the abovementioned conditions, so we output it.

The above operations effectively reduce the number of operations to update the cell's weight and the algorithm's running time.

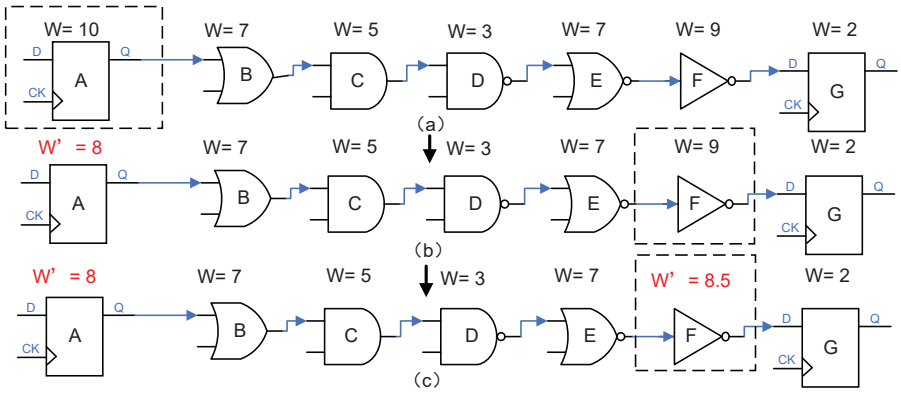


Figure 7. An example of the new way to update the cell weights.

4.2.2. Optimize the Way to Find the Path

Updating path weight is the operation with the highest time complexity in Algorithm 1. Therefore, the following method is proposed to reduce the time complexity.

We set a range according to path weight R, obtain the paths with the worst timing in the path collection, and set it as the subpath collection. There is a fact that for each path, the path weight only decreases as the cells are swapped to a lower threshold voltage. As long as the following operations are performed, the path with the worst timing must be in the subpath collection: (1) replace and update the path's weight in the subpath collection

every time and remove the path that no longer belongs to the range. (2) When the subpath collection is empty, give a new range to obtain a new subpath collection.

Therefore, we can only traverse the collection of subpaths when looking for the worst path, avoiding traversing all paths. We obtain the subpath collection in the following way: set a parameter D ($D > 1$) and set the initial range of the subpath set as $(1 - 1/D) < R < 1$. When a new range is needed, reduce the left and correct bounds by $1/D$ each.

We compared the experiment shown in Figure 6 based on the same circuit condition. Set $D = B^{0.5} = 655$. The relationship between the number of traversed paths and the times of swapping cells are shown in Figure 8. Observe that the number of traversed paths is always less than the value of B/D. According to the experiment results, the number of operations for extracting the sub-collection and traversing path is less than $(A * B/D + B * D)$. We take $D = B^{0.5}$ in the experiment. Thus the time complexity in the experimental results of the new approach is $O(B^{1.5})$.

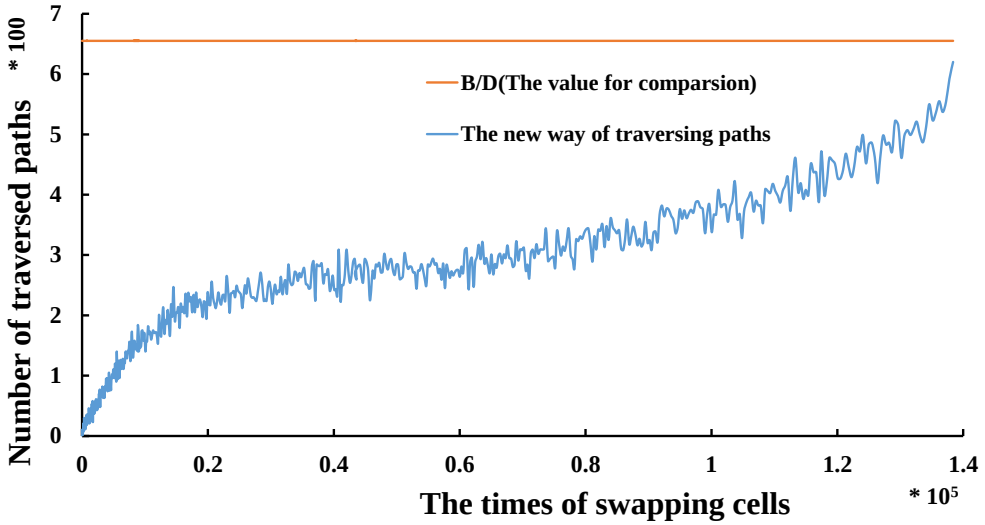


Figure 8. The relationship between the number of traversed paths and the times of swapping cells when using the new way.

4.2.3. The Update Algorithm

After applying the above two methods, the update algorithm for Algorithm 1 is shown in Algorithm 3. The main update steps are shown below.

- Updated step 1: Optimize the way to update cells' weight.
  The weights are not updated when replacing cells. Update the weights of some cells on the path after finding the most critical path. First, find the cell with the maximum weight on the path, update the weight of this cell, and then find the new cell with the maximum weight. Loop until the cell with the maximum weight has been updated.
- Updated step 2: Optimize the way to find the path.
  Extract the worst part from the path collection C as the sub-path collection SC, and the smallest R-value in the sub-collection is denoted as R_F. Find the most critical path m with the largest R_m in the sub-collection SC. When R_m is smaller than R_F, update the sub-collection SC.

Algorithm 3 The updated sub triple-threshold voltage distribution algorithm

Input: path collection C, static power consumption P, timing arcs, the OCV derate α
Output: sub cell swap list

Initialize each path i's t_{slack_i}, $t_{repairable_i}$ and R_i from collection C
Collect the replaceable cells on each path i as collection CIP_i
Collect the paths through each cell j as collection PTC_j
Initialize each cell j's weight WS_j (HVT-to-SVT) and WL_j (SVT-to-LVT)
Initialize the state of all cells to 0
When cell j's state is 0, $W_j = WS_j$; when state is 1, $W_j = WL_j$; when state is 2, $W_j = 0$
repeat
  Obtain sub-collection SC from collection C, and get R_F, R_m from SC
  Select the path m with the largest R_m value from SC
  while $R_m \geq R_F$ **do**
    repeat
      Select the cell n with the largest W_n in CIP_m
      Update cell n's WS_n (when state is 0) or WL_n (when state is 1)
    until W_n has been updated in this loop
    if cell state = 0 **then**
      Swap cell n from HVT to SVT, remark cell's state as 1
    else if cell state = 1 **then**
      Swap cell n from SVT to LVT, remark cell's state as 2
    end if
    for each path x in PTC_n **do**
      update t_{slack_x}, $t_{repairable_x}$, R_x with Equations (3), (4) and (6),
      update collection C, SC
    end for
    Select the path m with the largest R_m value from SC
  end while
until all paths' $t_{slack} \geq 0$
Return sub cell swap list

5. Experimental Results

The experiment is based on a CPU IP using 28 nm technology, while the number of cells is 385781, and the required frequency is 1 GHz. TPSPOM is applied on this circuit, while CBLPBR and CAPCOM have also been used for comparison.

5.1. Static Power Consumption Results

When the circuit is all-LVT, the value of static power consumption is 19.7909 mW.

First, we compare the methodologies' results in using only dual-threshold voltage, HVT, and LVT. Table 1 shows the experimental results of power consumption, the number of the LVT combinational cell, and the number of the LVT sequential cell under various delay constraints. Figure 9 shows the result of static power consumption, and Figure 10 shows the results of the percentage of the LVT cells.

The results show that for static power consumption optimization, CAPCOM is slightly better than CBLPRP, while TPSPOM is significantly better than the other two methods. Moreover, the results indicate that under the shortest 1 ns delay constraint, compared to the all-LVT circuit, CBLPRP's static power consumption is reduced to 35.12%, while CAPCOM's is reduced to 34.91% and TPSPOM's is reduced to 24.26%.

Table 1. Static power consumption, the number of low-threshold voltage (LVT) combinational cells and LVT sequential cells of the 28 nm circuit using CBLPRP, CAPCOM, and TPSPOM procedures with dual-threshold voltage.

Period (ns)	CBLPRP			CAPCOM			TPSPOM		
	Power (*mW*)	LVT *C*-Cells	LVT *S*-Cells	Power (*mW*)	LVT *C*-Cells	LVT *S*-Cells	Power (*mW*)	LVT *C*-Cells	LVT *S*-Cells
1.2	4.3813	108,354	2182	4.3818	65,112	10,284	2.6220	47,326	3804
1.1	5.5906	128749	4394	5.4998	85,966	12,923	3.5008	64558	5545
1.0	6.9498	146,037	8001	6.9088	109,385	16,035	4.8008	88315	7858

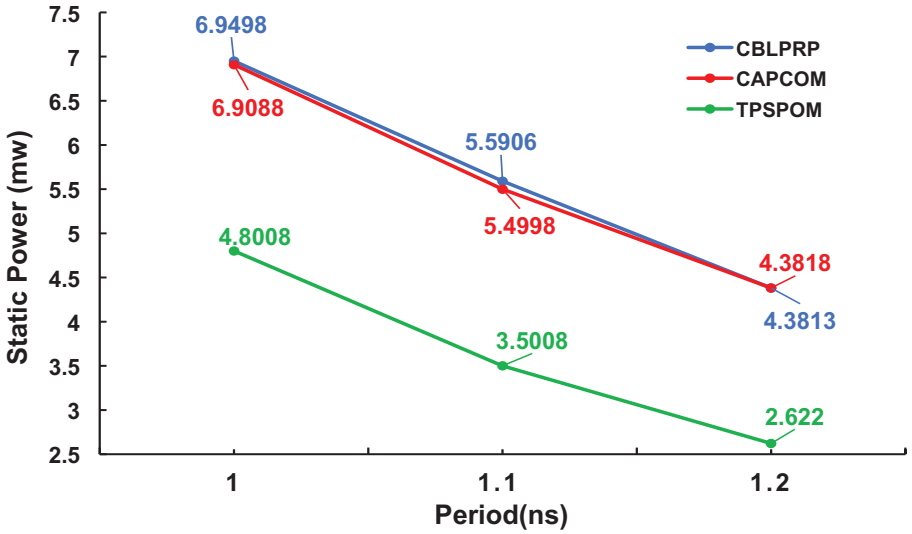


Figure 9. Static power consumption of the 28 nm circuit using CBLPRP, CAPCOM, and TPSPOM procedures with dual-threshold voltage.

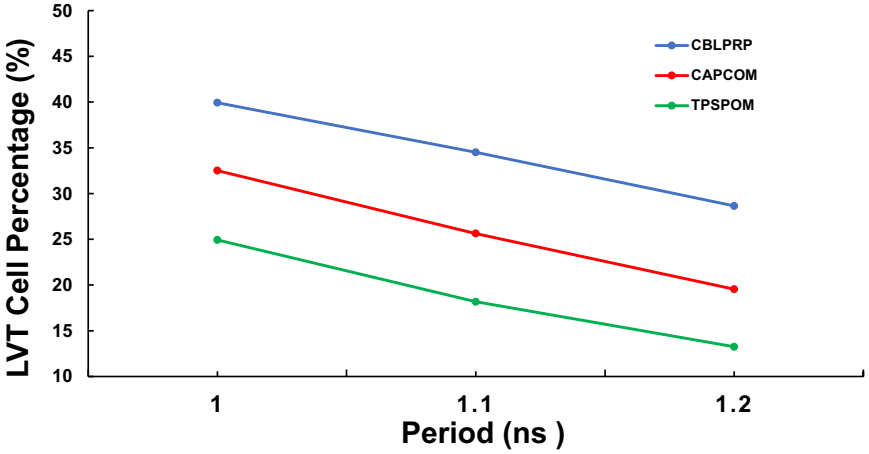


Figure 10. Percentage of LVT cells of the 28 nm circuit using CBLPRP, CAPCOM, and TPSPOM procedures with dual-threshold voltage.

For the percentage of LVT cells, CAPCOM is significantly lower than CBLPRP, while TPSPOM is significantly lower than the other two methods. It proves that using the parameter of delay-to-power ratio as a basis for distributing threshold voltage can effectively reduce the proportion of LVT cells.

Further, compare the results of the CAPCOM and TPSPOM methods when using triple-threshold voltage. The result is shown in Table 2, Figures 11 and 12.

Table 2. Static power consumption, the number of LVT/standard-threshold voltage (SVT) combinational cells and LVT/SVT sequential cells of the 28 nm circuit using CAPCOM, and TPSPOM procedures with triple-threshold voltage.

Period (ns)	CAPCOM					TPSPOM				
	Power (*mW*)	SVT *C*-Cells	SVT *S*-Cells	LVT *C*-Cells	LVT *S*-Cells	Power (*mW*)	SVT *C*-Cells	SVT *S*-Cells	LVT *C*-Cells	LVT *S*-Cells
1.2	3.074	23,650	5190	41,462	5094	1.3285	47,141	4027	9848	860
1.1	3.8933	30,022	6080	55,944	6843	1.5586	71,128	6335	12,588	1175
1.0	4.9298	36,291	7311	73,094	8724	1.9289	105,021	9216	17,459	1617

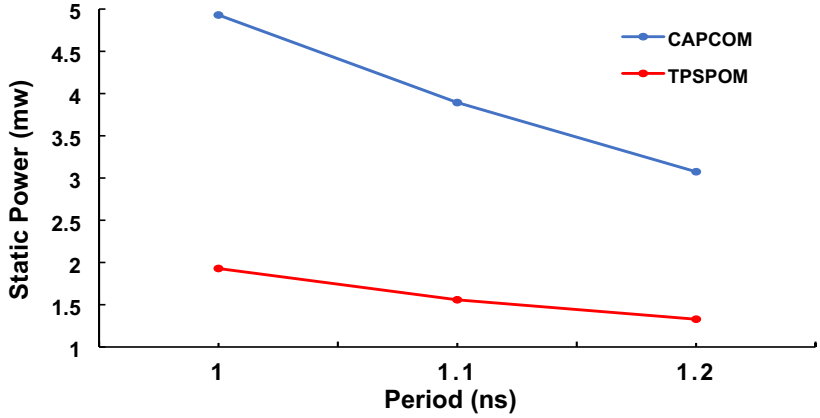


Figure 11. Static power consumption of the 28 nm circuit using CAPCOM, and TPSPOM procedures with triple-threshold voltage.

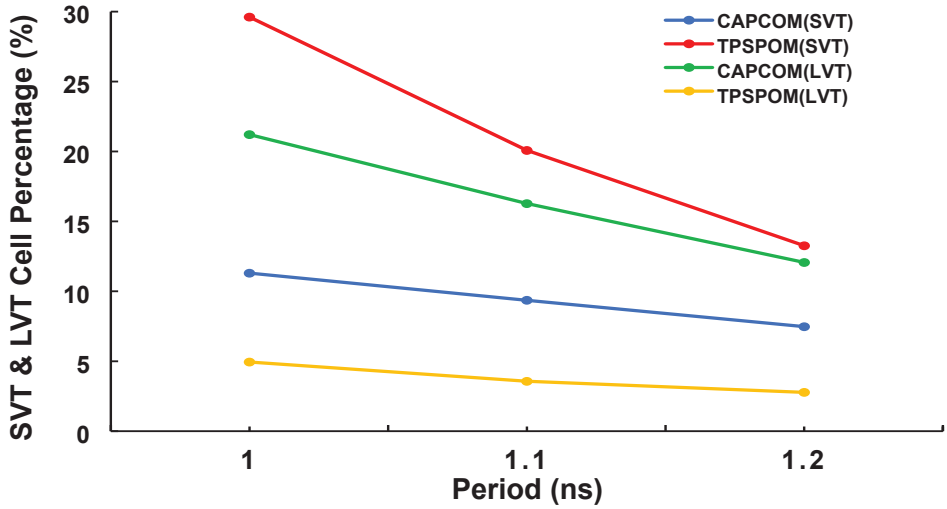


Figure 12. Percentage of SVT/LVT cell of the 28 nm circuit CAPCOM, and TPSPOM procedures with triple-threshold voltage.

The results show that the TPSPOM method is more effective when using triple-threshold voltage. The results indicate that under the shortest 1 ns delay constraint, compared to the all-LVT circuit, CAPCOM's static power consumption is reduced to 24.91%, while TPSPOM's is reduced to 9.75%. The percentage of the SVT cells for CAPCOM is 11.30%, while for TPSPOM is 29.61%. Meanwhile, the percentage of the LVT cells for CAPCOM is 21.2%, while for TPSPOM is 4.95%.

The CAPCOM method prioritizes swapping cells from LVT to HVT, then considers swapping cells from LVT to SVT. Moreover,the CAPCOM method will have a higher percentage of LVT compared to the priority replacement of SVT and then HVT.

However, the difference in power consumption between LVT and SVT is much more significant than that between SVT and HVT, and priority should be given to swapping cells from LVT to SVT. For example, in this circuit, the value of static power consumption for a particular inverter is $6.403 * 10^{-10}$ W for HVT, $6.561 * 10^{-9}$ W for SVT, and $9.663 * 10^{-8}$ W for LVT. The delay difference between HVT and SVT is 0.03235 ns, while the value is 0.01265 ns between SVT and LVT. Calculate the ratio of static power consumption difference to delay difference, and the result is that swapping from SVT to LVT is 38.9 times higher than that of swapping from HVT to SVT. It indicates that the static power consumption cost of using LVT cells is too large.

Therefore, when fixing timing violations, the design should prioritize converting the cell from HVT to SVT before considering converting the cell from SVT to LVT. The TPSPOM method implements a design philosophy that prioritizes HVT, followed by SVT, and finally LVT, which is more consistent with the current circuit design goals. Therefore, even if the percentage of SVT cells of TPSPOM is significantly larger than that of CAPCOM, it can achieve a lower static power consumption value because of the lower percentage of LVT cells.

5.2. Method Runtime

The experiments are run on a computer with 41.60 GFLOPS. With the lower LVT percentage of CAPCOM compared to CBLPRP, there will be fewer operations for CAPCOM to replace cells that do not meet the timing from HVT back to LVT, which makes CAPCOM run faster than CBLPRP. In the triple-threshold case, under the timing constraints of 1 ns, the CAPCOM runs for 139.6 h and the TPSPOM for 4.4 h. It shows that another advantage of the TPSPOM is that the time is acceptable when applied to a VLSI design.

In addition, with the initial path collection, the running time of Algorithm 1 is 15.6 h, while Algorithm 3 is 6 min.

6. Conclusions

This paper describes a triple-threshold path-based static power-optimization methodology (TPSPOM) using the MTCMOS technique. This method addresses the shortcomings of existing threshold voltage assignment algorithms that do not specifically consider the connection between cells and paths and frequently update the timing in STA tools. It improves both static power consumption results and algorithm running time:

(1) The algorithm proposed in this paper locates the cells to be replaced based on path weights and cell weights to optimize static power consumption. The path weight is obtained from the path constraint, and the cell weight is obtained from the path weight through the cell and its delay-to-power consumption ratio.
(2) The proposed algorithm uses a simplified collection of paths for running-time optimization to characterize the overall timing and avoids frequent timing updates in STA tools.

The experiments on a CPU IP containing 385,781 cells on a 28 nm process show that the TPSPOM method reduces static power consumption by 15.16% more than CAPCOM with a cycle time of 1 ns while reducing the running time by 96.85%. It proves that the algorithm flow provided in this paper can be effectively applied in the designs of VLSI and supports the distribution of triple-threshold voltage with a significant effect of optimizing static power consumption and low running time.

For future work, since the running speed of the algorithm is directly related to the scale of the circuit, we will further improve the running speed of the algorithm to be suitable for larger-scale circuits.

Author Contributions: Conceptualization and supervision, D.Z. and K.H.; methodology, P.L. and S.Z.; validation, P.L., S.Z., W.X. and C.X.; data curation, W.X. and C.X.; writing—original draft preparation, S.Z.; writing—review and editing, S.Z.; funding acquisition, D.Z. and K.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Key R&D Program of China (2020YFB0906000, 2020YFB0906001).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kumar, S. V.; Rao, P. V.; Sharath, H. A.; Sachin, B. M.; Ravi, U. S.; Monica, B. V. Review on VLSI design using optimization and self-adaptive particle swarm optimization. *J. King Saud Univ.-Comput. Inf. Sci.* **2020**, *32*, 1095–1107. [CrossRef]
2. Singh, V.; Arya, S. K.; Kumar, M. Different Perspectives of Low Power Design for CMOS VLSI Circuits. *Int. J. Electron. Eng.* **2018**, 604–613.
3. Lu, Y. H.; De Micheli, G. Comparing system level power management policies. *IEEE Des. Test Comput.* **2001**, *18*, 10–19.
4. Caldari, M.; Conti, M.; Crippa, P.; Orcioni, S.; Solazzi, M.; Turchetti, C. Dynamic power management in an AMBA-based battery-powered system. In Proceedings of the 9th International Conference on Electronics, Circuits and Systems, Dubrovnik, Croatia, 15–18 September 2002; Volume 2, pp. 525–528.
5. Zhang, J. F.; Gao, R.; Duan, M.; Ji, Z.; Zhang, W.; Marsland, J. Bias Temperature Instability of MOSFETs: Physical Processes, Models, and Prediction. *Electronics* **2022**, *11*, 1420. [CrossRef]
6. Moradinezhad Maryan, M.; Amini-Valashani, M.; Azhari, S. J. An input controlled leakage restrainer transistor-based technique for leakage and short-circuit power reduction of 1-bit hybrid full adders. *Int. J. Circuit Theory Appl.* **2021**, *49*, 2382–2395. [CrossRef]
7. Ghosh, P.; Saha, T.; Kumari, B. Aspects of Low-Power High-Speed CMOS VLSI Design: A Review. In *Industry Interactive Innovations in Science, Engineering and Technology;* Springer: Singapore, 2018; pp. 385–394.
8. Bai, N.; Hu, Z.; Wang, Y.; Xu, Y. Leakage Current Stability Analysis for Subthreshold SRAM. *Electronics* **2022**, *11*, 1196. [CrossRef]
9. Kao, J. T.; Chandrakasan, A. P. Dual-threshold voltage techniques for low-power digital circuits. *IEEE J. -Solid-State Circuits* **2000**, *35*, 1009–1018. [CrossRef]
10. Narendra, S.; De, V.; Antoniadis, D.; Chandrakasan, A.; Borkar, S. Scaling of stack effect and its application for leakage reduction. In Proceedings of the 2001 International Symposium on Low Power Electronics and Design, Huntington Beach, CA, USA, 6–7 August 2001; pp. 195–200.
11. Park, J. C.; Mooney, V. J.; Pfeiffenberger, P. Sleepy stack reduction of leakage power. In Proceedings of the International Workshop on Power and Timing Modeling, Optimization and Simulation, Santorini, Greece, 15–17 September 2004; pp. 148–158.
12. Hanchate, N.; Ranganathan, N. A new technique for leakage reduction in CMOS circuits using self-controlled stacked transistors. In Proceedings of the 17th International Conference on VLSI Design, Proceedings IEEE, Washington, DC, USA, 5–9 January 2004; pp. 228–233.
13. He, X.; Al-Kadry, S.; Abdollahi, A. Adaptive leakage control on body biasing for reducing power consumption in CMOS VLSI circuit. In Proceedings of the 2009 10th International Symposium on Quality Electronic Design, San Jose, CA, USA 16–18 March 2009; pp. 465–470.
14. Suguna, T.; Rani, M. J. Survey on power optimization techniques for low power VLSI circuit in deep submicron technology. *Int. J. Vlsi Des. Commun. Syst.* **2018**, *9*, 1–15.
15. Kuo, J. B.; Lin, S. C. *Low-Voltage SOI CMOS VLSI Devices and Circuits*; John Wiley & Sons: Hoboken, NJ, USA, 2004; pp. 241–244.
16. Dixit, A. Transistor Leakage Mechanisms and Power Reduction Techniques in CMOS VLSI Design. *Int. J. Adv. Res. Comput. Commun. Eng.* **2016**, *5*, 102–107.
17. Chung, B.; Kuo, J. B. Gate-level dual-threshold static power optimization methodology (GDSPOM) using path-based static timing analysis (STA) technique for SOC application. *Integration* **2008**, *41*, 9–16. [CrossRef]
18. Huang, H. X.; Shen, S. R.; Kuo, J. B. Cell-based leakage power reduction priority (CBLPRP) optimization methodology for designing SOC applications using MTCMOS technique. In Proceedings of the International Workshop on Power and Timing Modeling, Optimization and Simulation, Madrid, Spain, 26–29 September 2011; pp. 143–151.
19. Lin, G. J.; Hsu, C. B.; Kuo, J. B. Critical-path aware power consumption optimization methodology (CAPCOM) using mixed-V TH cells for low-power SOC designs. In Proceedings of the 2014 IEEE International Symposium on Circuits and Systems (ISCAS), IEEE, Melbourne, Australia, 1–5 June 2014; pp. 1740–1743.
20. Wei, L.; Chen, Z.; Johnson, M.; Roy, K.; De, V. Design and optimization of low voltage high performance dual threshold CMOS circuits. In Proceedings of the 35th Annual Design Automation Conference, San Francisco, CA, USA, 15–19 June 1998; pp. 489–494.

21. Samanta, D.; Pal, A. Optimal dual-V/sub T/assignment for low-voltage energy-constrained CMOS circuits. In Proceedings of the ASP-DAC/VLSI Design 2002, 7th Asia and South Pacific Design Automation Conference and 15h International Conference on VLSI Design, Bangalore, India, 11 January 2002; pp. 193–198.
22. Wang, Q.; Vrudhula, S. B. Algorithms for minimizing standby power in deep submicrometer, dual-V/sub t/CMOS circuits. *IEEE Trans. -Comput.-Aided Des. Integr. Circuits Syst.* **2002**, *21*, 306–318. [CrossRef]
23. Xianrui, L.; Haibo, G.; Xinquan, K.; Yushan, L. Dynamic Threshold Static Power Optimization Techniques for the chip. *J. Univ. Electron. Sci. Technol. China* **2009**, *38*, 443–446. (In Chinese)
24. Fan, R.; Dandan, Z.; Xiaolang, Y. An algorithm for reducing leakage power based on dual-threshold voltage technique. In Proceedings of the 2013 Fourth International Conference on Digital Manufacturing & Automation, Singapore, 13–15 December 2013; pp. 132–134.
25. Bhasker, J.; Chadha, R. *Static Timing Analysis for Nanometer Designs: A Practical Approach*; Springer Science & Business Media: Berlin, Germany, 2009.
26. El Motassadeq, T. CCS vs NLDM comparison based on a complete automated correlation flow between PrimeTime and HSPICE. In Proceedings of the 2011 Saudi International Electronics, Communications and Photonics Conference (SIECPC), Riyadh, Saudi Arabia, 24–26 April 2011; pp. 1–5.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.