Mode-wise Voltage-scalable Design with Activation-aware Slack Assignment for Energy Minimization

TaiYu Cheng
Osaka University
Japan
t-cheng@ist.osaka-u.ac.jp

Yukata Masuda
Nagoya University
Japan

Jun Nagayama
Socionext Inc.
Japan

Yoichi Momiyama
Socionext Inc.
Japan

Jun Chen
Osaka University
Japan

Masanori Hashimoto
Osaka University
Japan
hasimoto@ist.osaka-u.ac.jp

ABSTRACT

This paper proposes a design optimization methodology that can achieve a mode-wise voltage scalable (MWVS) design with applying the activation-aware slack assignment (ASA). Originally, ASA allocates the timing margin of critical paths with a stochastic treatment of timing errors, which limits its application. Instead, this work employs ASA with guaranteeing no timing errors. The MWVS design is formulated as an optimization problem that minimizes the overall power consumption considering each mode duration, achievable voltage reduction, and accompanied circuit overhead explicitly, and explores the solution space with the downhill simplex algorithm that does not require numerical derivation. For obtaining a solution, i.e., a design, in the optimization process, we exploit the multi-corner multi-mode design flow in a commercial tool for performing mode-wise ASA with sets of false paths dedicated to individual modes. Experimental results based on RISC-V design show that the proposed methodology saves 20% more power compared to the conventional voltage scaling approach and attains 15% gain from the single-mode ASA. Also, the cycle-by-cycle fine-grained false path identification reduced leakage power by 42%.

KEYWORDS

mode-wise voltage-scaling, activation-aware slack assignment, multi-corner multi-mode, downhill simplex method

ACM Reference Format:
TaiYu Cheng, Yukata Masuda, Jun Nagayama, Yoichi Momiyama, Jun Chen, and Masanori Hashimoto. 2021. Mode-wise Voltage-scalable Design with Activation-aware Slack Assignment for Energy Minimization. In *26th Asia and South Pacific Design Automation Conference (ASPDAC '21), January 18–21, 2021, Tokyo, Japan.* ACM, Tokyo, Japan, 7 pages. https://doi.org/10.1145/3394885.3431575

ASPDAC '21, January 18–21, 2021, Tokyo, Japan

ACM ISBN 978-1-4503-7999-1/21/01…$15.00
https://doi.org/10.1145/3394885.3431575

1 INTRODUCTION

Voltage scaling (VS) is one of the most common techniques for power reduction. Voltage reduction remarkably contributes to quadratic power-saving. However, voltage decrease involves an increase in the circuit delay and raises the possibility of timing error. The techniques to prevent timing errors can be categorized into two levels; operation level and design level. The former implements *in situ* monitoring devices in the circuits to adapt the supply voltage for maximizing the power efficiency while sustaining circuit functionalities, such as Razor [1], critical-path replicas [2], or canary flip-flops [3]. In the contrary, the latter intends to re-design the circuit by manipulating the timing margin to enhance VS efficiency. A representative technique in this field is activation-aware slack assignment (ASA), which associates the timing criticality of a path with its topological path delay and activity [4–6]. A research [5, 6] introduces ASA by allocating the timing margin with a stochastic mean-time-to-failure (MTTF) analysis. Timing errors are characterized by statistical static timing analysis and path activation analysis. This method accepts very few, yet a certain amount of timing errors, and hence voltage can be aggressively scaled down. However, such stochastic treatment of timing error limits its application, such as image processing and machine learning [7, 8].

Industrial designs, on the other hand, may not tolerate any timing error even in those domains since it induces inconsistency with the current production test policy. Undoubtedly, all control paths must be timing clear in any product since they could cause terrible problems in finite state machines. At this point, an interesting and curious question arises; how much ASA can obtain power gain with guaranteeing no timing error. In this context, we focus on operation modes that use different circuit blocks and functionalities since they have different active paths. The bias of active paths can be exploited for mode-wise voltage scaling (MWVS). It is attractive to investigate the potential of ASA for MWVS.

In this work, we propose a methodology to achieve MWVS design under the scheme of ASA, guaranteeing no timing error. We first define mode-wise voltages and corresponding design freedom as the design variables, and then formulate the problem for MWVS in consideration of the duration for each mode. Also, we present a fine-grained way to identify the false paths for each mode. The proposed methodology exploits multi-corner-multi-mode design

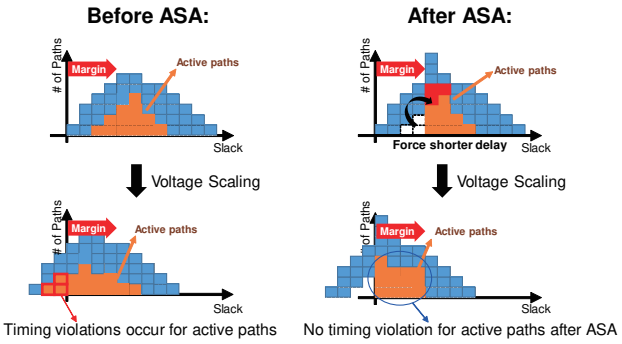


Figure 1: ASA concept to enhance VS efficiency.

flow in a commercial tool that considers sets of false paths in individual modes for performing mode-wise ASA. We explore the solution space with the downhill simplex algorithm (DSA), which is a direct search method for solving nonlinear optimization problems without derivative computation. We applied the proposed methodology to RISC-V CPU design. Experimental results show that our methodology achieves 13% to 20% overall power saving when treating conventional VS as the baseline. As for the comparison with single-mode ASA adopted in the conventional work [5, 6], the idea of MWVS attains additional power gain of 8% to 15%. Also, the false-path identification with the fine-grained method can reduce leakage power by 31% to 42%.

The rest of this paper is organized as follows. Section 2 reviews the related work about ASA and MWVS. Section 3 formulates the MWVS design optimization problem. Section 4 describes the proposed methodology for solving MWVS with DSA, which uses MCMM + ASA, and also presents fine-grained false path identification. Section 5 applies the proposed methodology to RISC-V and shows experimental results. Section 6 concludes this paper.

2 RELATED WORK

Fig. 1 illustrates the concept of ASA, which originates from an earlier technique so-called critical path isolation (CPI) [5, 9, 10]. ASA manipulates the active paths in a synthesized circuit by buffer insertion and cell swapping to allocate timing margin during an engineering change order (ECO) phase. After ASA, the active paths have more timing margin so that VS is applicable without timing error occurrence. Although the manipulation might increase the area due to inserted buffers and larger-size cells, the extra timing margin enables us to exploit VS.

Let us briefly explain the ASA method proposed by Masuda *et al.* [5, 6]. Given the pre-determined workloads, a set of active FFs is extracted. In ECO, such active FFs are given tighter setup timing constraints compared with the other FFs. As a result, the paths ending at the manipulated FFs increase their slacks after ECO. To determine the necessary timing margin, timing failure probability (P_{ERR}) is introduced and defined as a stochastic joint probability of the flop activation probability and its structural timing violation probability. Then, aggressive voltage scaling is applicable once the P_{ERR} of each selected flop at the scaled voltage is smaller than a given target value (but > 0). Hence, the circuit causes timing error at a certain probability, but it is very small.

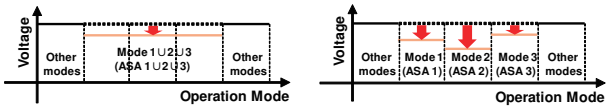


Figure 2: Expected voltage scaling operations for (a) single-mode VS design and (b) multi-mode VS design.

Nagayama *et al.* [4] apply ASA in an industry-friendly style. This work aims to lower the supply voltage for a power-hungry operation mode (workload) to reduce the peak power. It expands the timing margins of all the active paths in the mode of interest. It experimentally attains VS in the mode in an industrial design [4], where this mode-dependent VS is called mode-wise voltage scaling (MWVS). According to our objective of this work, no timing error should happen if the timing for those active paths is all clean. However, we reproduced their work and found that the timing error occurred at the scaled voltage of interest because of unexpected glitch events occurring in non-active paths. Thus, their ASA implementation is still risky for timing error issues. Overall, the previous works mentioned above for implementing ASA cannot achieve zero timing error.

Furthermore, previous works [4–6] consider only one specific operation mode. Therefore, even if there are three distinct operation modes from 1 to 3, for example, those modes are united to form a single mode such as Mode $1 \cup 2 \cup 3$, as shown in Fig. 2a. In this case, further VS opportunities existing in Mode 1 and Mode 2 are spoiled, as shown in Fig. 2b. Conversely, if multiple modes are individually considered in ASA-aware voltage-scalable circuit design, more benefits of overall energy saving can be expected because each mode consumes less power.

MWVS is a promising approach that can exploit the bias of active paths in different operation modes for voltage scaling. On the other hand, we need to establish a design methodology that can ensure no timing error occurrence, cope with multiple operation modes, and efficiently explore the design space. Here note that the managed VS is applied for each mode (workload), and the supplied voltage is spatially identical in the whole design. Therefore, the concept of MWVS is practical and reasonable for industrial designs. On the other hand, we need to mention that ASA for pursuing lower voltage operation involves circuit overhead due to timing margin expansion. Hence, this circuit overhead and achievable VS must be carefully taken into account in the design methodology.

3 PROBLEM FORMULATION FOR MWVS

In this section, we formulate the problem for optimizing a design under MWVS as follows.

- *Input* :
  (1) a gate-level pre-ASA circuit design
  (2) modes to be analyzed, M_i, where $i = 1, 2, ..., N$
  (3) duration DR_i for mode M_i, where $i = 1, 2, ..., N$
  (4) nominal voltage V_{NOM}
  (5) cycle time T
- *Output* :
  (1) a gate-level circuit after MWVS-aware ASA.
  (2) voltage V_i for mode M_i, where $i = 1, 2, ..., N$
- *Objective function*

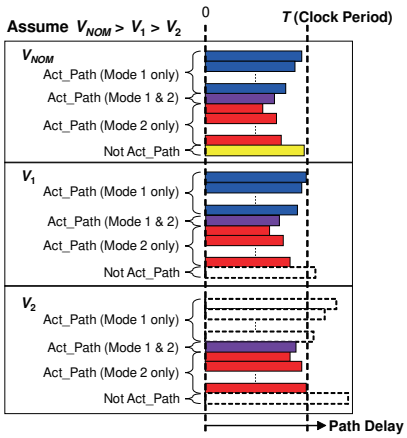


Figure 3: Mode-wise timing constraints (two-mode case).

(1) Minimize: $\sum_{i=1}^{N} \text{Power}(M_i, V_i) \times DR_i$

- *Constraints*

(1) Delay(V_i, Act_Paths$_i$) $\leq T$, where $i = 1, 2, ..., N$
(2) Delay(V_{NOM}, All_Paths) $\leq T$
(3) Area $\leq$ Area$_{\max}$

- *Variables*

(1) voltage V_i
(2) size and threshold voltage Vt type of individual cells

The input and output are the gate-level circuits before and after MWVS-aware ASA. M_i is the i-th mode in N modes, and DR_i is the portion of the i-th mode duration, and hence each DR$_i$ ≤ 1 and $\sum_{i=1}^{N} DR_i = 1$. The predetermined nominal voltage and clock cycle are V_{NOM} and T, respectively. The objective of this problem is to minimize the summation of the power multiplied by the duration from mode M_1 to M_N. The first two constraints are given for the timing closure. That is, for each mode M_i, all the delays of active paths in mode M_i, which are denoted as Act_Paths$_i$, should meet the setup time constraint for the given clock period of T at the operating voltage of V_i. In addition, at the nominal voltage V_{NOM}, the delays of all the paths, All_Paths, in the design should keep setup timing clean within the same criterion of T. Also, the area is constrained with Area$_{\max}$ since faster logic cells tend to be large. Hold timing constraints are considered, but they are omitted in the above since they are not our primary concern.

Here, we should mention that the power in each mode M_i is determined by its operation voltage V_i in two ways. The first way comes from the fact that the power is expressed as the product of voltage and current, i.e., Power $= V \times I$. The second way is that the cell sizing and and Vt assignment (swapping the same type of cells to different threshold voltage) result, which affects power dissipation, depends on V_i since the cell swapping is performed such that the path delay constraints are satisfied at V_i. Therefore, solving this problem needs to consider these two dependencies of power dissipation on the selection of V_i.

Fig. 3 shows an illustration regarding the timing constraints. There are four groups of paths in the figure; the paths activated in Mode 1 only (blue), Mode 2 only (red), both Mode 1 and 2 (purple), and non-activated paths (yellow). Three voltages having $V_{NOM} > V_1 > V_2$ relation are assumed. At V_{NOM}, all the path delays should pass the setup-time criterion with a given clock period of T. V_1 is the scaled voltage for Mode 1, and thus all the paths activated in Mode 1 (blue and purple) should meet the criterion of T under V_1. V_2 for Mode 2 (purple and red) follows accordingly. Note that the paths in blue are excluded from the criterion for Mode 2, and their delays can violate the setup-time constraint. The paths in purple activated in both Mode 1 and Mode 2 are required to meet the criterion for both the modes, while the paths in yellow not activated in Mode 1 or Mode 2 can violate the criterion in both the modes.

In summary, this optimization problem aims at the minimization of the total sum of Power × Duration. Since the duration represents the time period and Energy = Power × Time, the objective is equivalent to energy minimization.

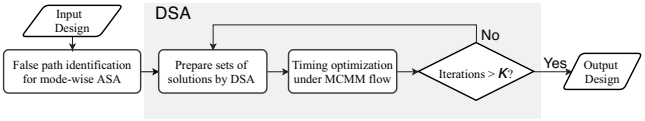


Figure 4: Overall flow for solving MWVS problem.

4 PROPOSED DESIGN METHODOLOGY

This section introduces the proposed MWVS design methodology that solves the optimization problem formulated in Section 3. Ultimately, we want to determine V_i, cell sizes, and cell types simultaneously. However, for each V_i value, we need to perform time-consuming circuit optimization, i.e., cell sizing and swapping for timing closure. Furthermore, this timing closure optimization needs to satisfy all the timing constraints separately specified for each mode. We, therefore, take a two-step approach that decouples V_i optimization and circuit optimization.

Fig. 4 shows the overall flow. The iteration loop aims to obtain the set of V_i that can minimize the objective function. In this work, the downhill simplex algorithm (DSA) is used for this iterative optimization. DSA is a classical algorithm that can solve optimization problems having multidimensional variables without derivatives, which helps save the computation cost [11, 12]. κ is the parameter to limit the number of iterations. This loop includes the circuit optimization with MCMM flow, where the timing closure optimization is executed for given sets of V_i. The detail of DSA in MWVS flow is described in Section 4.1. This circuit optimization with MCMM flow is explained in Section 4.2. Before this iterative optimization, we need to prepare sets of false paths separately for each mode, which is discussed in Section 4.3.

4.1 Downhill Simplex Algorithm (DSA)

Downhill simplex algorithm (DSA), which is also known as Nelder-Mead method, is a simple numerical method for finding the optimal solution in a multidimensional space [11, 12]. The advantage of DSA is that it does not rely on the gradients to search the next decision, and therefore, DSA can deal with the optimization problem in which computing the gradient of the objective function is impossible or too time-consuming. Thanks to this property, DSA saves computing effort for the problem in which the computation of the objective function is time-consuming and numerical calculation of the gradient is prohibitively expensive. Hence, the proposed methodology adopts DSA.

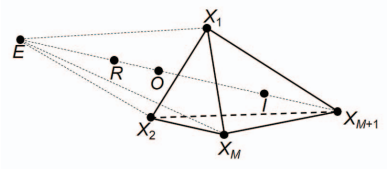


Figure 5: Illustration of three-dimensional simplex in DSA.

To solve an M-dimension problem, DSA prepares an initial geometrical simplex composed of $M + 1$ vertices in the solution space. Alternating the vertex iteratively through the comparison of their corresponding objective function values makes the simplex approach to the optimal solution. Fig. 5 illustrates a three-dimension simplex, or as known as tetrahedron, as an example. Let us assume there is an objective function, F(.), which needs to be minimized, and there are $M + 1$ vertices X_1, X_2, ..., X_M, X_{M+1}, which are initially selected and sorted as $F(X_1) \leq F(X_2) \ldots \leq F(X_M) \leq F(X_{M+1})$. Points R, E, O, I represent the next candidates of the vertices of the simplex with the name as reflection, expansion, outer contraction and inner contraction, respectively. The positions of R, E, O, I in the solution space are calculated as the linear combination of X_1, X_2, ..., X_M, X_{M+1}. DSA chooses the next set of vertices for the simplex in each iteration by conditionally comparing F(R), F(E), F(O), F(I) with F(X_1), F(X_M), and F(X_{M+1}). For the details, please see textbook, such as [11]. Note that the number of points newly evaluated in each iteration is at most four independent of M, which is an advantage of DSA.

We explain how to solve the MWVS problem with DSA. As mentioned at the beginning of Section 4, we search an optimal set of V_i using DSA, where a set of V_i corresponds to a vertex in DSA, and the total number of vertices are $N + 1$. For each vertex, we perform circuit optimization for timing closure with MCMM flow and evaluate the objective function. However, when the set of V_i includes too low voltage, the given timing constraints cannot be satisfied and non-zero negative slack is observed. In this case, the set of V_i cannot be considered as a solution candidate. On the other hand, DSA cannot consider such constraints directly. Therefore, we add a penalty function to the objective function to guide DSA taking into timing violation consideration. Here, there are many candidates of the penalty functions, but in this work we use the simple penalty function below since we empirically found the penalty shape does not affect the final solution much.

$$\mathrm{F}(.) = \begin{cases} \sum_{i=1}^{N} Power(M_i, V_i) \times DR_i & \mathrm{WNS} = 0, \\ EXT & \mathrm{WNS} > 0, \end{cases} \quad (1)$$

where EXT is an extreme large number, and WNS stands for worst negative slack that is reported by the EDA tool. In addition to WNS, EXT also applies to the condition that Area > $\mathrm{Area}_{\max}$ in our experiment, but it rarely happens since WNS is the primary constraint. Unless the run-time concern, the iteration number κ is recommended to be set with a sufficient large number to reach the convergence of DSA.

4.2 Integrating Mode-wise ASA into MCMM

We exploit MCMM flow in EDA tools [13, 14] to carry out mode-wise ASA. The EDA tool asks users to prepare scenarios, where each scenario is associated with the timing library under a particular PVT corner and corresponded design constraints files. Therefore, the false-path specification commands for each mode should be included in the design constraints files of the associated scenarios.

For each scenario corresponding to Mode 1 to Mode N, we assign the library at the specified voltage and associate the corresponding timing constraints files M_1.sdc to M_N.sdc as well. In addition to the clock period T, the false paths identified for individual modes are described in the design constraint. An additional scenario (for Mode NOM) is for the library of the nominal voltage of V_{NOM}, where no information on false paths is given. In other words, this scenario specifies that all the paths in design need to pass the timing at V_{NOM}. Note that if there are design-specific false paths, they should still be specified in M_{NOM}.sdc as well. Hence, as long as operating in nominal voltage, the design after MWVS flow can guarantee their functionality even for any modes that are not considered in MWVS design flow. The optimization in the MCMM flow aims to meet all constraints simultaneously. Therefore, as long as the timing is clean, i.e., WNS = 0, the design is guaranteed to operate correctly in every mode at the voltage of interest, and, for example, the logic simulation passes.

4.3 False Path Identification

Previous works on ASA [4–6] pay attention to active paths (or flops) obtained from logic simulation results and allocate additional slacks to those paths. However, due to glitch events, it cannot guarantee that there is no timing error at the scaled voltage of interest after ASA circuit optimization. Let us explain the reason. Even when there is no transition on a path, a glitch may arise and propagates through the path after ASA timing optimization since glitch occurrence strongly depends on transition timings. Such new glitches are hard to be predicted and then raise the risk for timing violation. Therefore, instead of specifying active paths, we decide to identify safe false paths that are non-active during the mode operation of interest. Thus there is no need to worry about accidental glitch transitions even after timing optimization.

Fig. 6 exemplifies false paths. The illustrated circuit is composed of flops (FF-0∼FF-7), AND2 gates (A1∼A5), and OR2 gates (O1∼O6). After performing a one-time logic simulation for a particular workload (mode), we can extract the states (0 or 1) for each cycle at the output pins of all the flops and define the flops whose output states are identical with the previous cycle as non-active flops. Then, for each end-point flop (e.g., FF-0), with assigning non-active flops (e.g., FF-1, FF-7) in static timing analysis, we can extract the false paths that include false sub-paths (e.g., from FF-1/Q to O1/A and FF-7/Q to O6/B). Even though the input patterns vary every cycle, some flops are non-active from the beginning to the end in a particular mode. Similar always non-active flops in a mode might be extracted by propagating mode-dependent constant values. We primarily extract the false paths based on those flops and regard this method as conventional false-path identification (FPI).

On the other hand, we have found that applying cycle-by-cycle analysis could potentially increase the number of false paths. Let us take the circuit in Fig. 6 again as an example. Topologically there are 13 paths in the circuit, which are listed on the right side with ID numbers. Conventionally, we consider these 13 paths during timing closure. On the other hand, supposing FF-1 and FF-7 are non-active

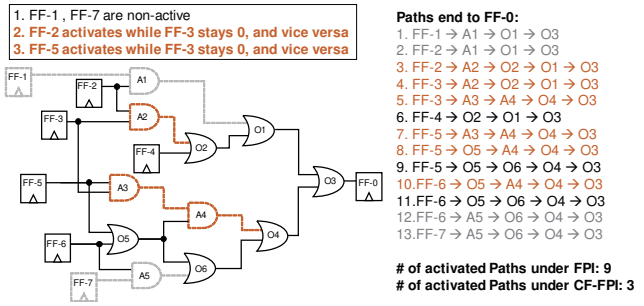


Figure 6: Examples of false path identification including cycle-by-cycle analysis.

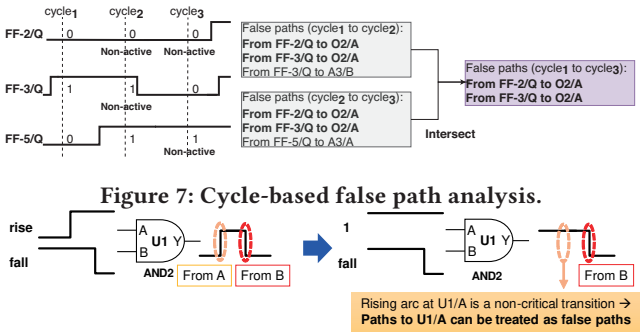


Figure 7: Cycle-based false path analysis.

Figure 8: Cycle-based non-critical transition and false path analysis in AND2 gate.

and 0 in a particular mode, we attain four false paths (paths 1,2,12,13 in gray) and reduce the number of paths to be considered to 9. This reduction is identical to FPI mentioned above. Next, let us consider the case that FF-2, FF-3, FF-5 are not always non-active but either of FF-2 and FF-3 transitions only when the other stays 0, and the same relation holds between FF-5 and FF-3. In this case, we reduce the number of paths to be considered in timing closure to 3 (paths 6,9,11 in black). Cycle-by-cycle analysis can automatically extract such FF-to-FF relations from the logic simulation results, which will be explained in the next paragraph. After the false path identification, we assign the false paths in EDA tools in an ordinary way. We can simply assign the false paths through the false stages. Taking an example from Fig. 6, we assign all the paths through A1,A2,A4,A5 as false paths. Gate A3 is also the false stage, but the paths through A3 can be specified by A4, and then A3 is skipped.

Let us explain how to automatically extract false paths mentioned above from the logic simulation result. The idea is very simple. We extract false paths for each clock edge. In this part, the executed analysis is the same as FPI. The set of false paths varies for each edge, but some false paths are included for all the clock edges. Finally, such false paths are given to the timing closure tool. In this way, the additional false paths discussed in Fig. 6 can be obtained. We call this false path identification as cycle-by-cycle fine-grained false path identification (CF-FPI). Fig. 7 illustrates CF-FPI from the example circuit in Fig. 6. The false paths are extracted for every cycle. From $cycle_1$ to $cycle_2$, we obtain the false paths from FF-2/Q to O2/A, FF-3/Q to O2/A, and FF-3/Q to A3/B for $cycle_1$ to $cycle_2$, and the false paths from FF-2/Q to O2/A, FF-3/Q to O2/A and FF-5/Q to A3/A for $clock_2$ to $clock_3$. Then, we take the AND of the sets of the false paths and obtain the final list, i.e., path from FF-2/Q to O2/A and FF-3/Q to O2/A. This example includes only two clock edges, but the same analysis can be applied to long simulation results.

Finally, we further squeeze false paths. Till now, we primarily extract false paths by using the information on non-active flops. On the other hand, we can identify additional paths from active flops as "false". This possibility arises in multi-input combinational gates, e.g., AND2 in Fig. 8. Suppose that U1/A has only one rise toggle, and U1/B has only one fall toggle in the same cycle. In this case, due to the AND2 boolean logic, the output transition U1/Y is dominated by the falling input i.e., U1/B. The other pin U1/A has a non-critical transition (NCT), and thus we can use the path through U1/A as a false path. The important property of NCT is that this dominance relation is independent of timing, and it holds even after any timing optimization. Therefore, we can exploit this false path safely in cell-swapping optimization. It should be noted that this false path identification based on the NCT is possible with the cycle-by-cycle analysis. Besides, the multi-input gates possessing similar characteristics include (N)AND , (N)OR, AOI, OAI series of gates. MUXs have NCT possibilities since in the case of S=0 or S=1, either A or B pin becomes don't care term, meaning that the transition at that pin becomes NCT. X(N)OR series, HA, and FA do not have this characteristic since every input transition affects the output value.

5 EXPERIMENTAL RESULTS AND ANALYSIS

5.1 Setup

We perform an experimental evaluation of the proposed methodology with RISC-V processor, a popular open-source CPU. We synthesized it with Nangate 45nm library for VTG (low-Vt) and VTH (high-Vt) cells at 0.5 GHz by Synopsys Design Compiler (DC) which enables MCMM and set V_{NOM} to 1.0V. The power for each mode is estimated by DC. After obtaining the value for mode-wise power, the operation duration is considered to calculate the overall power. Since 0.5 GHz is not the highest frequency for synthesizing RISC-V, many logic cells in the circuit are swapped to higher-Vt cells to save leakage power in the initial synthesis.

Three workloads are selected in our experiments; (1) dijkstra and (2) sha, which are included in MiBenchmark [15], and (3) mt-matmul-fp, which is a floating-point (FP) matrix-multiplication. The workloads of dijkstra and sha use similar parts of processor components since they use integer arithmetic and logic operations. On the other hand, mt-matmul-fp utilizes the FP units, especially for multiply-accumulate computation. Therefore, the modes dedicated for dijkstra and sha are quite different from the mode for mt-matmul-fp, which meets our assumption that the operating voltage and the power consumption might have a discrepancy between different modes.

For comparison, we prepare three methods for evaluation; (A) conventional VS, (B) single-mode ASA + VS, (C) mode-wise ASA + VS (proposed). Method (A) directly re-synthesizes the design at a lower voltage without any specification of false paths, which is a standard design flow and then our baseline. Method (B) applies ASA that is based on the false paths that are not activated in Mode

Table 1: Optimization results for (A) conventional VS (baseline), (B) single-mode ASA + VS, and (C) mode-wise ASA + VS (proposed).

Method	#Modes (used modes)	Duration	Voltage (V)	Power (W) (reduction)
A			0.82	0.358 (baseline)
B	1 (1∪3)		0.80	0.340 (-5.0%)
B	1 (1∪2∪3)		0.80	0.340 (-5.0%)
C	2 (1, 3)	50:50	(0.73, 0.80)	0.312 (**-12.8%**)
C	2 (1, 3)	70:30	(0.73, 0.80)	0.301 (**-15.9%**)
C	2 (1, 3)	95:5	(0.73, 0.80)	0.286 (**-20.1%**)
C	3 (1, 2, 3)	33:33:33	(0.76, 0.73, 0.80)	0.307 (**-14.3%**)

1 to Mode 3, namely Mode 1 to Mode 3 are merged into a single mode. This method has the similarity as the previous work [4–6] in terms of the number of modes while the stochastic treatment of timing error used in previous work [5, 6] is disabled. Method (C) is the proposed approach described in Section 4. In (C), the duration for each mode can be explicitly considered to minimize the overall power dissipation, while the duration information cannot be considered in (A) and (B).

5.2 Results

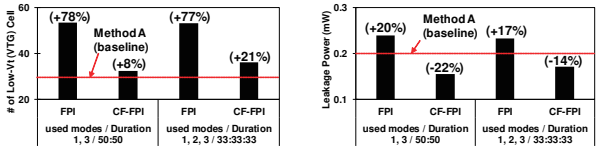
Table 1 lists the experimental setup and results. The first column represents the three methods (A), (B), (C). The second column lists how many modes are used and shows the combination of the used modes. Method (A) and (B) only apply one mode, but (B) considers the merged mode. The third column is the assumed duration that only (C) can explicitly consider when computing overall power. The fourth column for (A) and (B) is the minimum voltage at which the WNS sustains 0, while for (C) it is the optimization results after 30 DSA iterations. The final column for (A) and (B) represents the power dissipation, and for (C) it means the overall power considering the duration. We apply FPI here for (B) and (C). FPI and CF-FPI for (C) will be compared later.

From Method (A) to (B), the power is reduced by 5.0%. (B) unites different modes to one, and hence the VS efficiency of (B) is limited by the mode that needs the highest voltage. In this case, Method (B) is determined by Mode 3 (mt-matmul-fp) due to its smaller room for VS than the other two modes. However, the proposed methodology of Method (C) optimizes the design considering all the modes separately, and thus it enhances the VS efficiency. Even in the case of two modes (1, 3) having 50:50 duration, an additional 8% gain is obtained from Method (B) to (C). Additionally, the proposed methodology allows different sets of duration. When the duration ratio of Mode 1 to Mode 3 is changed from 50:50 to 70:30 and even 95:5, we can gain the power reductions of 16% and 20%, respectively. In this work, the area increase is limited to be smaller than 0.5%, and then the area increase was at most 0.4%. On the other hand, the number of low-Vt (VTG) cells increased. For one circuit optimization for MCMM timing closure takes 8~13 minutes for Method (A), (B) and (C) under FPI. Therefore, the entire MWVS design flow of (C) takes about six hours.

We next investigate the impact of false path identification methods on the optimization results. Unfortunately, the MCMM flow with the false-path set of CF-FPI is slow and circuit optimization with CF-FPI could take 6~7 hours for one run. Therefore, only the final iteration is executed with CF-FPI. Meanwhile, we think the run time varies with different designs and workloads, and CF-FPI flow is not always so slow. Table 2 compares the number of commands for false path specification for the cases with and without applying CF-FPI. Roughly speaking, the number of commands is similar. Fig. 9a compares the usage of low-Vt (VTG) cells. FPI specifies fewer false paths, and then the timing closure is more difficult, which results in 77 to 78% larger number of low-Vt cells. On the other hand, CF-FPI identified more false paths than FPI so that fewer lower-Vt cells are used, and the increase of VTG cells is suppressed to only 8% to 21%. Although we obtain minor improvement for the total power (0.03%) due to the small portion of leakage power to the overall power, Fig. 9b reveals that applying CF-FPI reduces leakage power by 42% in the two-mode case and 31% in the three-mode case. This result indicates that CF-FPI facilitates the timing optimization and reduces the number of lower-Vt cells.

Table 2: # of false-path specifications in design constraint file for each mode

	Mode		
	1	2	3
FPI	161K	162K	214K
CF-FPI	186K	192K	185K



(a) (b)

Figure 9: Comparison between FPI and CF-FPI for (a) # of low-Vt (VTG) cells (b) leakage powers.

5.3 Proposed methodology investigation

Fig. 10a and Fig. 10b show how the solution is becoming convergent, where Fig. 10a supposes two-mode (1, 3) case of 50:50 duration and Fig. 10b supposes three-mode (1, 2, 3) case of 33:33:33 duration. In each figure, we plot two curves starting from different initial values. Initialization 1 applies the voltage set of (1.0, 1.0), (0.9, 1.0), and (1.0, 0.9) as the start points in the two-mode optimization, while initialization 2 applies (1.0, 1.0), (0.85, 1.0), and (1.0, 0.85). Similarly, the start points are set in the three-mode case. Fig. 10a shows that both initial sets of the two-mode case can converge to the same power value after 17 iterations. However, in the three-mode case, the different initial sets would reach different power dissipation values. The reason is that, although DSA is not a completely greedy algorithm, it does not have an explicit hill-climbing capability, and then it can fall into local optimal points. Meanwhile, the objective function is supposed to be somewhat a smooth function since it is a linear sum of the product of the power and duration for each mode, and the power is roughly proportional to the voltage squared. However, the MCMM flow of EDA tool might generate non-continuous space for this objective function once the number of modes increases.

Fig. 11a and Fig. 11b show the scatter plots of the voltages in two different modes evaluated in the optimization process, where the red points mean that the timing closure fails in the MCMM flow and the blue points mean that the timing closure succeeds. Fig. 11a

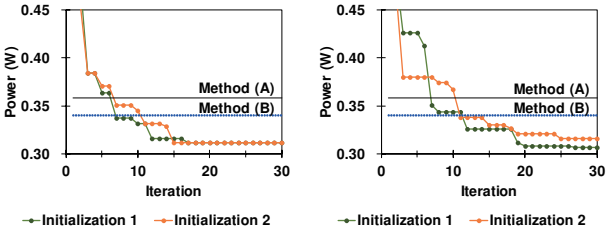


Figure 10: Convergence plots of the optimization for (a) two-mode case (1, 3), and (b) three-mode case (1, 2, 3) starting from two different initial points.

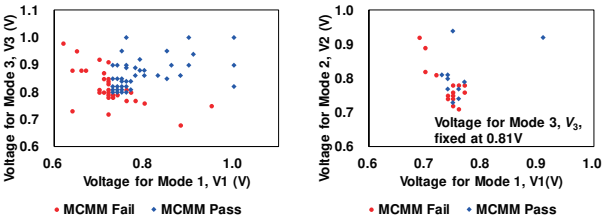


Figure 11: Timing closure results at various sets of voltages for individual modes. (a) two-mode case (1,3). (b) three-mode case (1, 2, 3) where the voltage for Mode 3 is fixed at 0.81 V.

is for two-mode case, and Fig. 11b is for three-mode case, where one dimension of voltage, i.e., voltage for Mode 3, V_3, is fixed at 0.81 V. Focusing on the two-mode case, we could find a clear bound; the timing closure passed in the MCMM flow when $V_1 \geq 0.73$ V and $V_3 \geq 0.80$ V. Although there are a few outlier points that fail, they are relatively rare, and the DSA works well. However, in the three-mode case, even when we fix one dimension of voltage V_3 at 0.81 V, the plot shows that the boundary is unclear and the red and blue dots are mixed, especially around the point of $(V_1, V_2) = (0.73, 0.73)$. This behavior indicates that if the solution approaches this area, the algorithm might be disturbed by the outlier points and converge at a local minimum point. Actually, since the MCMM flow itself is another complex optimization problem and its complexity also increases according to the number of modes, the stability of the overall solution may become sensitive to the set of initial points. On the other hand, comparing to Method (A) and (B), the proposed methodology provides higher power efficiency. Note that DSA is not the sole solver for this problem, and other methods can be used as long as their performance is better.

6 CONCLUSION

This work proposed a design methodology based on ASA to achieve a mode-wise voltage-scalable (MWVS) design with guaranteeing no timing errors. We formulated the MWVS design as an optimization problem toward the minimized energy operation by defining operation voltages for individual modes and cell sizes and Vt types as the variables. We integrated ASA with the MCMM flow in EDA tools, and then applied DSA to solve the problem numerically. We also introduced a fine-grained identification method of false paths that can be excluded in timing optimization without any risk of timing error. Our evaluation based on RISC-V design achieved 20% gain of power efficiency compared to the conventional VS method, where 15% gain comes from the mode-wise idea. We also confirmed that the fine-grained false path identification facilitated the timing closure and reduced leakage power by more than 30%.

REFERENCES

[1] S. Das, D. Roberts, S. P. S. Lee, D. Blaauw, T. Austin, T. Mudge, and K. Flautner, "A self-tuning dvs processor using delay-error detection and correction," *IEEE J. Solid-State Circuits*, vol. 41, no. 4, pp. 792–804, 2006.
[2] J. Park and J. A. Abraham, "A fast, accurate and simple critical path monitor for improving energy-delay product in dvs systems," *IEEE/ACM International Symposium on Low Power Electronics and Design*, pp. 391–396, 2011.
[3] Y. Kunitake, T. Sato, and H. Yasuura, "A replacement strategy for canary flip-flops," *IEEE Pacific Rim International Symposium on Dependable Computing*, pp. 227–228, 2010.
[4] J. Nagayama, Y. Masuda, M. Takeshige, Y. Ogawa, M. Hashimoto, and Y. Momiyama, "Activation-aware slack assignment (asa) for mode-wise power saving in high-end isp," *ACM/IEEE DAC Designer/IP track*, 2019.
[5] Y. Masuda, M. Hashimoto, and T. Onoye, "Critical path isolation for time-to-failure extension and lower voltage operation," *ACM/IEEE ICCAD*, pp. 1–8, 2016.
[6] Y. Masuda, T. Onoye, and M. Hashimoto, "Activation-aware slack assignment for time-to-failure extension and power saving," *IEEE TVLSI*, vol. 26, no. 11, pp. 2217–2229, 2018.
[7] K. Roy and A. Raghunathan, "Approximate computing: An energy-efficient computing technique for error resilient applications," *IEEE Computer Society Annual Symposium on VLSI*, pp. 473–475, 2015.
[8] Q. Xu, T. Mytkowicz, and N. S. Kim, "Approximate computing: A survey," *IEEE Design & Test*, vol. 33, no. 1, pp. 8–22, 2016.
[9] S. Ghosh, S. Bhunia, and K. Roy, "Crista: A new paradigm for low-power, variation-tolerant, and adaptive circuit synthesis using critical path isolation," *IEEE TCAD*, vol. 26, no. 11, pp. 1947–1956, 2007.
[10] A. B. Kahng, S. Kang, R. Kumar, and J. Sartori, "Slack redistribution for graceful degradation under voltage overscaling," *IEEE ASP-DAC*, pp. 825–831, 2010.
[11] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, *Numerical Recipes: The Art of Scientific Computing (3rd. ed.)*. Cambridge University Press, 2007.
[12] L. Baudzus and P. M. Krummrich, "Modified downhill simplex method for fast adaption of optical filters in optical communication systems," *Electronics Letters*, vol. 55, no. 8, pp. 471–473, 2019.
[13] A. B. Kahng, S. Kang, R. Kumar, and J. Sartori, "Enhancing the efficiency of energy-constrained dvfs designs," *IEEE TVLSI*, vol. 21, no. 10, pp. 1769–1782, 2013.
[14] D. Flynn, *Low power methodology manual: For system-on-chip design.* Springer Science & Business Media, 1st. ed., 2007.
[15] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "Mibench: A free, commercially representative embedded benchmark suite," *IEEE WWC*, pp. 3–14, 2001.