IEICE TRANSACTIONS

on Fundamentals of Electronics, Communications and Computer Sciences

VOL. E105-A NO. 3
MARCH 2022

A PUBLICATION OF THE ENGINEERING SCIENCES SOCIETY
The Institute of Electronics, Information and Communication Engineers
Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

PAPER *Special Section on VLSI Design and CAD Algorithms*

Activation-Aware Slack Assignment Based Mode-Wise Voltage Scaling for Energy Minimization

TaiYu CHENG[†a)], ***Nonmember***, **Yutaka MASUDA**[††], ***Member***, **Jun NAGAYAMA**[†††], ***Nonmember***, **Yoichi MOMIYAMA**[†††], ***Member***, **Jun CHEN**[†], ***Nonmember***, ***and*** **Masanori HASHIMOTO**[†], ***Member***

SUMMARY Reducing power consumption is a crucial factor making industrial designs, such as mobile SoCs, competitive. Voltage scaling (VS) is the classical yet most effective technique that contributes to quadratic power reduction. A recent design technique called activation-aware slack assignment (ASA) enhances the voltage-scaling by allocating the timing margin of critical paths with a stochastic mean-time-to-failure (MTTF) analysis. Meanwhile, such stochastic treatment of timing errors is accepted in limited application domains, such as image processing. This paper proposes a design optimization methodology that achieves a mode-wise voltage-scalable (MWVS) design guaranteeing no timing errors in each mode operation. This work formulates the MWVS design as an optimization problem that minimizes the overall power consumption considering each mode duration, achievable voltage lowering and accompanied circuit overhead explicitly, and explores the solution space with the downhill simplex algorithm that does not require numerical derivation and frequent objective function evaluations. 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. We applied the proposed design methodology to RISC-V design. Experimental results show that the proposed methodology saves 13% to 20% more power compared to the conventional VS approach and attains 8% to 15% gain from the conventional single-mode ASA. We also found that cycle-by-cycle fine-grained false path identification reduced leakage power by 31% to 42%.

key words: *mode-wise voltage-scaling, activation-aware slack assignment, multi-corner multi-mode, downhill simplex method*

1. Introduction

Low-power operations are eagerly demanded in various computing systems, such as IoTs [1], [2], wearable equipment [3], and the sensor networks [4], [5], in addition to mobile terminals. According to long standby time, tiny volume, and limited energy sources, those applications have strong demands for ultra-low power consumption. Also, power consumption becomes the most competitive factor in modern mobile SOC [6], or even high-performance chips [7]. Designers are dedicated to pursuing the maximization of power or energy efficiency.

Voltage scaling (VS) is one of the most common and powerful techniques for power reduction. Voltage reduction remarkably contributes to the quadratic gain of power-saving. However, voltage decrease involves an increase in the circuit delay and raises the possibility of timing error. Therefore, the techniques to prevent timing errors are studied, and they 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. These techniques are categorized as adaptive voltage scaling (AVS) techniques. The key idea is to insert the monitoring sensors behind the main circuits for detecting or predicting the logic error. If the main circuit keeps functional, then the voltage controlling unit keeps lowering down the supply voltage. On the other hand, once the sensors detect or predict the logic error, they would inform the controlling units to increase the voltage to maintain the functionality of the main circuit. This area includes Razor [8], [9], critical-path replicas [10], [11], or canary flip-flops [12]–[14]. On the other hand, the latter level intends to re-design the circuit such that the timing margin is manipulated to enhance voltage scaling efficiency. A state-of-the-art technique in this field is an activation-aware slack assignment (ASA), which associates the timing criticality of a path with its topological path delay and activity [6], [14]–[17].

Figures 1(a) and 1(b) illustrate the concept of ASA, which originates from an earlier technique so-called critical path isolation (CPI) [15], [17]–[20]. In a logic design, there are several flop-to-flop timing paths. If the path delays are too long compared with the given clock period and the paths have a transitive (rising or falling) signal propagation during the operation, those paths violate the "setup" timing constraint and would cause timing errors. This timing violation may easily occur during VS when the circuit is designed for nominal voltage, since VS would increase the paths delay. The concept of ASA focuses on a fact that not all the paths would be activated during the operation of interest, which means not all the paths would have the transitive signal propagation. From now on, we give a definition: the paths in which a transitive signal propagates during the operation are denoted as active paths, whereas the paths without any transitive signal propagation are denoted as false paths. Note that the active paths and the false paths would vary depending on the circuit workloads and operation modes.

Manuscript received March 9, 2021.
Manuscript revised July 12, 2021.
Manuscript publicized August 31, 2021.

[†]The authors are with the Department of Information Systems Engineering, Graduate School of Information Science and Technology, Osaka University, Suita-shi, 565-0871 Japan.

[††]The author is with the Center for Embedded Computing Systems, Graduate School of Informatics, Nagoya University, Nagoya-shi, 464-0814 Japan.

[†††]The authors are with Socionext Inc., Yokohama-shi, 222-0033 Japan.

a) E-mail: t-cheng@ist.osaka-u.ac.jp

DOI: 10.1587/transfun.2021VLP0006

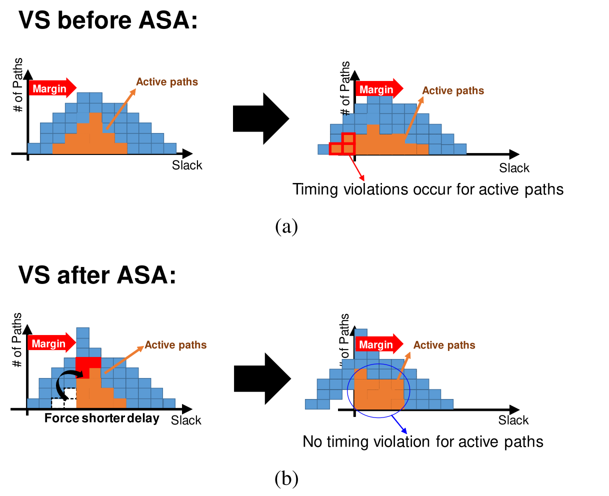


Fig. 1 Concept schematic for applying ASA (a) VS before ASA and (b) VS after ASA.

If the paths do not have any transitive signal propagation, those paths do not experience logic errors even with setup timing violation. Therefore, we need to consider the timing of only the active paths during the operation of interest. ASA manipulates a synthesized circuit for the active paths 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 gates, the extra timing margin enables us to apply voltage scaling. Moreover, AVS and ASA are highly feasible to achieve low power/energy regardless of the type of design. Besides, the two techniques can even integrate together to perform a synergy effect. Reference [14] claims that thanks to ASA, the number of timing critical paths becomes less, and hence the paths need to be inserted with monitoring sensors can be effectively reduced, which can achieve significant area overhead reduction while sustaining the circuit functionality.

However, current state-of-the-art ASA techniques remain space for further discussion. To enable larger VS, Masuda et al. [14]–[16] proposed an ASA method that allocates the timing margin with a stochastic mean-time-to-failure (MTTF) analysis. The timing errors are characterized by statistical static timing analysis and path activation analysis. This method accepts very few yet certain timing errors, and hence voltage can be aggressively scaled down. However, such stochastic treatment of timing error limits application domains to those that tolerate timing error, such as image processing and machine learning [21], [22]. In addition, MTTF-based ASA is very hard to be quantified since there is no effective way to validate its functional correctness in a reasonable time especially when things come to an aging issue. Besides, industrial designs may not tolerate any timing error even in those domains since MTTF-based ASA induces inconsistency with the current production test policy. On the other hand, without MTTF treatment, pure ASA may attain a very limited margin for VS since the paths whose delays are squeezable by gates or structural manipulation through ECO may not be many. Therefore, how to enlarge the power/energy gain through ASA with guaranteeing no timing error becomes an interesting and attractive problem.

This paper proposes a methodology to achieve a design applying to mode-wise VS (MWVS) under the scheme of ASA with guaranteeing no timing error. First, a mode-wise voltage and corresponding design freedom are defined as the design variables, and then a problem for MWVS is formulated in consideration of the duration of each mode. Then, this work presents a fine-grained way to identify the false paths for each mode. The proposed methodology exploits multi-corner multi-mode (MCMM) design flow in a commercial tool that considers sets of false paths in individual modes for performing mode-wise ASA. The solution space is explored through downhill simplex algorithm (DSA), which is a direct search method for solving nonlinear optimization problems without derivative computation. Note that the proposed method for MWVS is a general solution for application requiring low-power operation. A preliminary version of this work was presented in [23].

The contributions of this paper are:

- Formulate an optimization problem and provide its solving procedure to achieve mode-wise voltage-scalable design. Experiment results based on RISC-V design show that 13% to 20% overall power saving when treating design with conventional VS as the baseline. In addition, 8% to 13% additional power gain originates from the mode-wise idea.
- Introduce a cycle-by-cycle fine-grained method to identify false paths which are used during ASA implementations. Compared with the conventional way of false-path identification, the introduced method can save 31% to 42% leakage power.

The rest of this paper is organized as follows. Section 2 reviews the related work about ASA and MWVS, and also MCMM is briefly mentioned. 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 chapter.

2. Previous Work

2.1 Mode-Wise ASA

ASA is the technique to manipulate the timing margin of the active paths by engineering change order (ECO) phase. The active paths after ASA have more timing margin with the cost of the area from additional buffer insertion or fast-but-large gates swapping, while the extra timing margin offers further down-scaling of voltage and benefits the

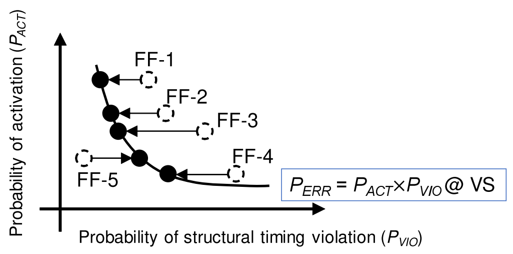


Fig. 2 Timing margin for each FF is determined such that P_{ERR} is constraint while individual P_{ACT} values are different.

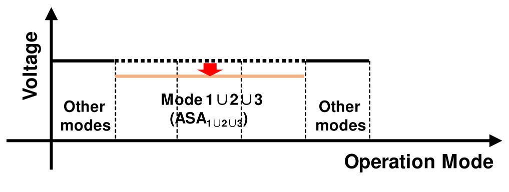
power/energy reduction. The benefit from ASA has been promoted in the state-of-the-art works [6], [14]–[16].

Let us briefly explain the ASA method proposed by Masuda et al. [14]–[16]. Given the pre-determined workloads or mode-control input pattern, a set of active FFs is extracted and then FF-based ASA is performed. That can be done in ECO phase by the re-synthesis with tighter delay constraint on the paths whose endpoints are those active FFs. Therefore, as long as the design can be re-synthesized successfully following the criterion, those paths should have more timing margin for VS. For determining the necessary timing margin, timing failure probability is introduced and defined as a stochastic joint probability (P_{ERR}) by a flop activation (P_{ACT}) and its structural timing violation probabilities (P_{VIO}). Then, aggressive voltage scaling, which reaches 25% voltage reduction in their experiments, is applied as long as the P_{ERR} defined at the scaled voltage for each selected flop is within a given target value, as shown in Fig. 2. Here, the target value is a non-zero positive value, and hence the circuit causes timing error at a certain probability while it is very small.

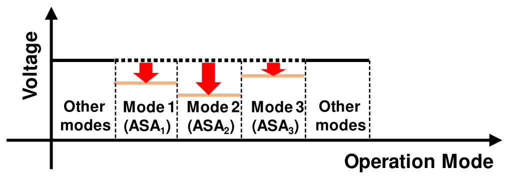
On the other hand, in order to enhance VS efficiency without tolerating timing errors, it is necessary to explore other margin sources that can be exploited. Reference [6] attempts to introduce mode-wise ASA to enjoy the benefit from ASA targeting on a certain power-hungry mode in the industrial design. That is, with ASA, it allows to apply reduced voltage to that power-hungry mode, while for other modes (or workload), regular voltage level is supplied to make sure the normal function with no modes exception.

However, even though [6] has introduced the term "mode-wise ASA," their implementation approach of ASA, which is the same as the previous work of MTTF-aware ASA [14]–[16], considers only one specific operation mode and focuses on enhancing the VS efficiency only for that single mode. Therefore, 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. 3(a). In this case, further VS opportunities existing in Mode 1 and Mode 2 are spoiled, as shown in Fig. 3(b). 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



(a)



(b)

Fig. 3 Expected voltage scaling operations for (a) single-mode VS design and (b) multi-mode (mode-wise) VS design.

bias of active paths in different operation modes for voltage scaling. On the other hand, it is necessary 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 scaled voltage is applied to each mode (workload) even under MWVS scheme, and the supplied voltage is spatially identical in the design of interest. Therefore, the concept of MWVS is practical and reasonable for industrial designs. On the other hand, it is noted 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.

2.2 MCMM

Multi-corner multi-mode (MCMM) design flow has been recently developed as one of the built-in features in many EDA vendors such as Synopsys and Cadence [24], [25]. The MCMM flow allows the timing analysis and the optimization of a hardware design according to multiple modes, where each mode can be associated with different clock periods, PVT corners (process, voltage, temperature), and design constraints for mode-based timing. The MCMM design flow works to generate a circuit design that satisfies all the required specifications in individual modes simultaneously. The MCMM flow can significantly save the design turn-around times for MCMM designs [25].

With the MCMM flow, the ASA technique can be explicitly extended to MWVS design methodology. The mode-wise ASA aims to obtain a design where all the FFs at nominal voltage and the active FFs at scaled voltage meet the clock frequency constraint. Thus, it is expected that the MCMM flow is a promising choice to enhance the capability of MWVS design flow.

3. Problem Formulation for MWVS

This section formulates the problem for optimizing a design under MWVS as follows.

- *Input*:
  1. a gate-level pre-ASA circuit design, D_0
  2. Mode M_i and associated duration DR_i, where $i = 1, 2, ..., N$
  3. nominal voltage V_{NOM}
  4. cycle time T
- *Output*:
  1. a gate-level circuit after MWVS-aware ASA, D.
  2. voltage V_i for mode M_i, where $i = 1, 2, ..., N$
- *Objectfunction*
  1. Minimize: $\sum_{i=1}^{N} \text{Power}(M_i, V_i) \times DR_i$
- *Constraints*
  1. Slack(T, V_i, Act_Paths$_i$) ≥ 0, where $i = 1, 2, ..., N$
  2. Slack(T, V_{NOM}, All_Paths) ≥ 0
  3. Area $\leq$ Area$_{\text{max}}$
- *Variables*
  1. voltage V_i
  2. size and V_t type of individual cells

The input and output are the gate-level circuits before and after MWVS-aware ASA, which are denoted as D_0 and D, respectively. 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 $DR_i \leq 1$ 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 active paths in mode M_i, which are denoted as Act_Paths$_i$, should meet the setup time constraint (slack ≥ 0) for the given cycle time T at the operating voltage of V_i, where the slack definition will be detailed later. In addition, at the nominal voltage V_{NOM}, the delays of all the paths, All_Paths, in the design should keep the setup timing clean as well for the given T. The area is constrained with Area$_{\text{max}}$ since faster logic cells tend to be large. Hold timing constraints are considered, but they are omitted in the above since they are not the primary concern in this work.

Here, it should be mentioned 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 power depends on the design after the re-synthesis, namely, the cell sizing and V_t assignment (swapping the same type of cells to different threshold voltage) performed for timing satisfaction at V_i affects power dissipation. Therefore, solving this problem needs to consider these two dependencies of power dissipation on the selection of V_i.

Figure 4 illustrates Constraints 1 and 2 with path delays and the cycle time T. Here, the path delay includes the

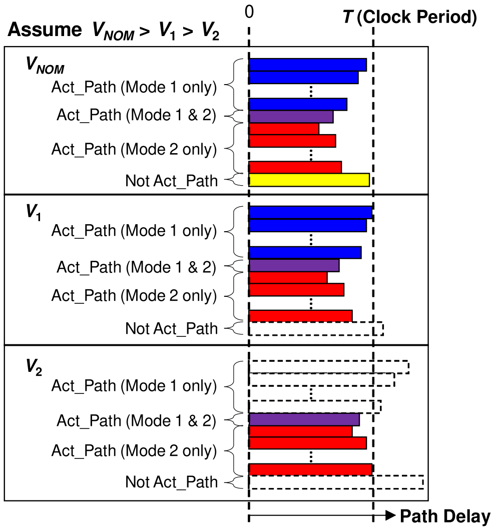


Fig. 4 Mode-wise timing constraints (two-mode case).

clock-to-q delay at the launch FF, the maximum path delay of the combinational circuit from the launch FF to the capture FF, and the setup time of the capture FF. Therefore, the slack is expressed as slack = T - (path delay). In this case, Constraint 1 can be considered as Delay(ACT_Path$_i$) $\leq T$ at V_i for $i = 1, 2, ..., N$, and Constraint 2 means Delay(All_Paths) $\leq T$ at V_{NOM}. In Fig. 4, 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-active 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 $\times$ Duration. Since the duration represents the time period and Energy = Power $\times$ Time, the objective is equivalent to energy minimization.

4. Proposed Design Methodology for MWVS

This section introduces the proposed MWVS design methodology that solves the optimization problem formulated in Sect. 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

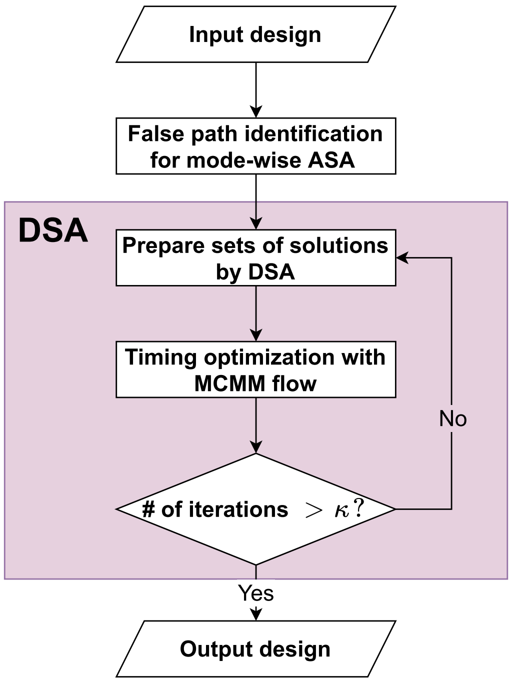

Input design
False path identification for mode-wise ASA
DSA
Prepare sets of solutions by DSA
Timing optimization with MCMM flow
# of iterations $> \kappa$?
No
Yes
Output design

Fig. 5 Overall flow for solving MWVS problem.

mode. This work, therefore, takes a two-step approach that decouples V_i optimization and circuit optimization. Figure 5 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 [26]–[28], 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 Sect. 4.1. This circuit optimization with MCMM flow is explained in Sect. 4.2. Before this iterative optimization, sets of false paths should be prepared separately for each mode, which is described in Sect. 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 [26]–[28]. The advantage of DSA is that it does not rely on the gradients to search the next decision. 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.

To solve an M-dimension problem, DSA prepares an initial geometrical simplex composed of $M+1$ vertices in the solution space, where the dimension of each vertex is M. Alternating the vertex iteratively through the comparison of their corresponding objective function values makes the simplex approach to the optimal solution. Figure 6 illustrates a three-dimension simplex for the number of mode $M = 3$, or known as tetrahedron, as an example. If $M = 2$, the simplex becomes a triangle. 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 determined by user and sorted as F(X_1) $\leq$ F(X_2) ... $\leq$ F(X_M) $\leq$ F(X_{M+1}). When referring to Fig. 6, there are $3+1 = 4$ vertices, X_1, X_2, X_3 and X_4, where each X_i ($i = 1 \sim 4$) consists of three values (x_{i1}, x_{i2}, x_{i3}). When $M = 2$, each X_i ($i = 1 \sim 3$) becomes a 2-value set (x_{i1}, x_{i2}). 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 potions 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}), where F(X_2) is excluded in comparison according to DSA rule. For the details, please see textbook, such as [26]. No matter what the number of M is, the number of points newly evaluated in each iteration is at most four (R, E, O, I), and the function values comparisons are at most within the seven points (F(R), F(E), F(O), F(I), F(X_1), F(X_M), and F(X_{M+1})). Then, each iteration replaces X_{M+1} by either of R, E, O, and I while the other points X_1, X_2, ..., and X_M with known objective function values are kept for the next iteration. This operation presents the advantage of DSA. The number of objective function evaluations in each iteration is constant even when the number of M is large. On the other hand, the methods that compute gradients need to perform $M+1$ evaluations for numerical gradient computation.

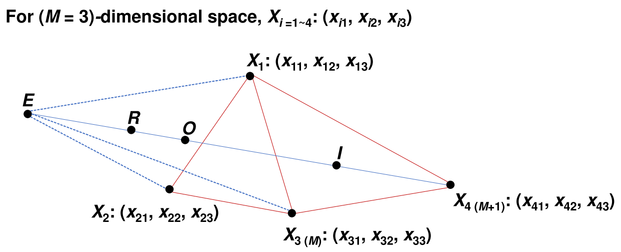

For (M = 3)-dimensional space, $X_{i=1\sim4}$: (x_{i1}, x_{i2}, x_{i3})
X_1: (x_{11}, x_{12}, x_{13})
E
R
O
I
X_2: (x_{21}, x_{22}, x_{23})
$X_{3\,(M)}$: (x_{31}, x_{32}, x_{33})
$X_{4\,(M+1)}$: (x_{41}, x_{42}, x_{43})

Fig. 6 Illustration of three-dimensional simplex in DSA.

This paragraph explains how to solve the MWVS problem with DSA. As mentioned at the beginning of Sect. 4, the solution aims to 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, this flow performs circuit optimization for timing closure with MCMM flow and evaluates the objective function. However, when the set of V_i includes too low voltage, the given Constraint 1 & 2 in Sect. 3 cannot be satisfied and negative slack values are 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, this flow adds a penalty function to the objective function to guide DSA taking into consideration timing violation. Here, there are many candidates of the penalty functions, but this work uses the simple penalty function below since the experiment results empirically found the penalty shape does not affect the final solution much.

$$\mathrm{F}(.) = \begin{cases} \sum_{i=1}^{N} Power(D_x, M_i, V_i) \times DR_i & \text{Constraints are met,} \\ EXT & \text{Otherwise,} \end{cases} \tag{1}$$

where D_x means the design re-synthesized at the point of interest (.) in the solution space. It should be noted that F(.) is the implicit function of D_x since the designs re-synthsized for individual points in the solution space, i.e. sets of V_i, are different. In each iteration of DSA, we only change the assigned voltages for all the modes and perform the re-synthesis, where the input design is always the original D_0. EXT is an empirical large number that increases F(.), for example 100 times, when either of constraints 1 to 3 is violated. 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. Our experiments, which will be presented in Sect. 5, show that DSA converges after 25 iterations for the MWVS design problem.

4.2 Integrating Mode-Wise ASA into MCMM Flow

This work exploits MCMM flow in EDA tools [24], [25] 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.

Figure 7 illustrates the preparation of the scenarios for mode-wise ASA. Several timing libraries characterized at different voltages are prepared beforehand. Then, for each scenario corresponding to Mode 1 to Mode N, this flow assigns the library at the specified voltage and associates 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 all the paths in design that 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., slack $\geq$ 0, the design is guaranteed to operate correctly in every mode at the voltage of interest, and, for example, the logic simulation passes.

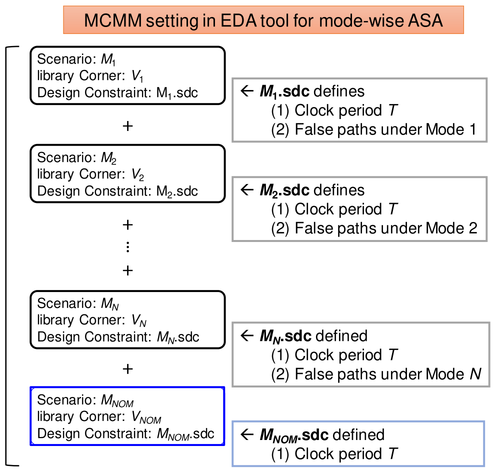


Fig. 7 MCMM setting in EDA tool for MWVS design.

4.3 False Path Identification

Previous works on ASA [6], [14]–[16] 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 propagate 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 they raise the risk for timing violation. Therefore, instead of specifying active paths, this work decides 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.

Figure 8 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), the states (0 or 1) for each cycle can be extracted at the output pins of all the flops, and the flops whose output states are identical with the previous cycle are defined 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, the false paths that include false sub-paths (e.g., from FF-1/Q to O1/A and FF-7/Q to O6/B) can be extracted. Even though the input patterns vary every cycle, some flops are non-active from the beginning to the end in a particular mode. Such always-non-active flops in a mode might be extracted by propagating mode-dependent constant values. The false paths are primarily extracted based on those flops. We regard this method as conventional false-path identification (FPI).

On the other hand, it is found that applying cycle-by-cycle analysis could potentially increase the number of false

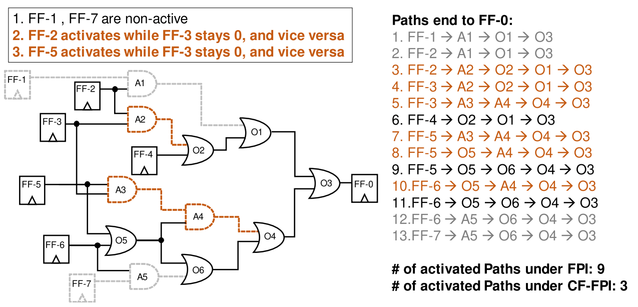

1. FF-1 , FF-7 are non-active
2. FF-2 activates while FF-3 stays 0, and vice versa
3. FF-5 activates while FF-3 stays 0, and vice versa
FF-1
FF-2
FF-3
FF-4
FF-5
FF-6
FF-7
FF-0
A1
A2
A3
A4
A5
O1
O2
O3
O4
O5
O6
Paths end to FF-0:
1. FF-1 → A1 → O1 → O3
2. FF-2 → A1 → O1 → O3
3. FF-2 → A2 → O2 → O1 → O3
4. FF-3 → A2 → O2 → O1 → O3
5. FF-3 → A3 → A4 → O4 → O3
6. FF-4 → O2 → O1 → O3
7. FF-5 → A3 → A4 → O4 → O3
8. FF-5 → O5 → A4 → O4 → O3
9. FF-5 → O5 → O6 → O4 → O3
10.FF-6 → O5 → A4 → O4 → O3
11.FF-6 → O5 → O6 → O4 → O3
12.FF-6 → A5 → O6 → O4 → O3
13.FF-7 → A5 → O6 → O4 → O3
# of activated Paths under FPI: 9
# of activated Paths under CF-FPI: 3

Fig. 8 Examples of false path identification including cycle-by-cycle analysis.

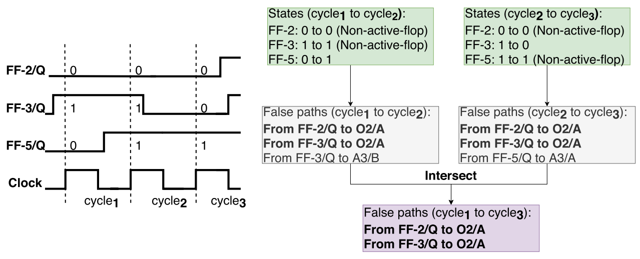

FF-2/Q
FF-3/Q
FF-5/Q
Clock
0
0
0
1
1
0
0
1
1
cycle$_1$
cycle$_2$
cycle$_3$
States (cycle$_1$ to cycle$_2$):
FF-2: 0 to 0 (Non-active-flop)
FF-3: 1 to 1 (Non-active-flop)
FF-5: 0 to 1
States (cycle$_2$ to cycle$_3$):
FF-2: 0 to 0 (Non-active-flop)
FF-3: 1 to 0
FF-5: 1 to 1 (Non-active-flop)
False paths (cycle$_1$ to cycle$_2$):
From FF-2/Q to O2/A
From FF-3/Q to O2/A
From FF-3/Q to A3/B
False paths (cycle$_2$ to cycle$_3$):
From FF-2/Q to O2/A
From FF-3/Q to O2/A
From FF-5/Q to A3/A
Intersect
False paths (cycle$_1$ to cycle$_3$):
From FF-2/Q to O2/A
From FF-3/Q to O2/A

Fig. 9 Cycle-based false path analysis.

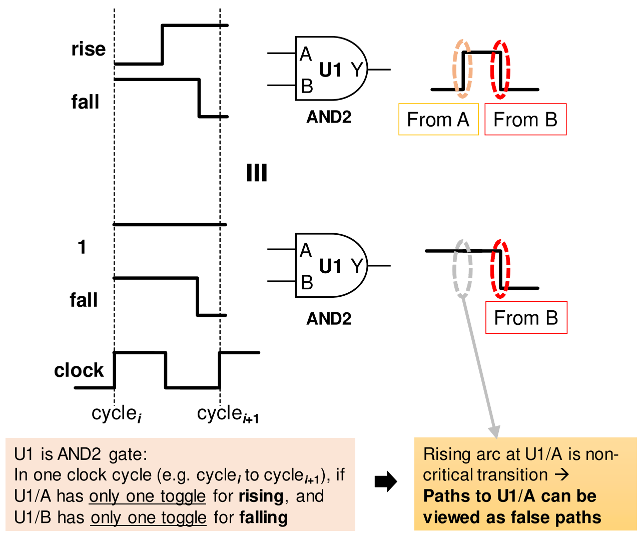

rise
fall
A
B
U1
Y
AND2
From A
From B
III
1
fall
A
B
U1
Y
AND2
From B
clock
cycle$_i$
cycle$_{i+1}$
U1 is AND2 gate:
In one clock cycle (e.g. cycle$_i$ to cycle$_{i+1}$), if U1/A has only one toggle for rising, and U1/B has only one toggle for falling
Rising arc at U1/A is non-critical transition →
Paths to U1/A can be viewed as false paths

Fig. 10 Cycle-based non-critical transition and false path analysis in AND2 gate.

paths. Let us take the circuit in Fig. 8 again as an example. Topologically there are 13 paths in the circuit, which are listed on the right side with ID numbers. Conventionally, in this schematic, these 13 paths are considered during timing closure. On the other hand, supposing FF-1 and FF-7 are non-active and 0 in a particular mode, four false paths (paths 1,2,12,13 in gray) arise, and the number of paths to be considered is reduced to 9. This reduction is identical to FPI mentioned above. Next, 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. This case reduces 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, the false paths are assigned in EDA tools in an ordinary way. EDA tool can simply assign the false paths through the false stages. Taking an example from Fig. 8, all the paths are assigned 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.

This paragraph explains how to automatically extract false paths mentioned above from the logic simulation results. The idea is very simple. The false paths are extracted 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. 8 can be obtained. This way of false path identification is called as cycle-by-cycle fine-grained false path identification (CF-FPI). Figure 9 illustrates CF-FPI from the example circuit in Fig. 8. The false paths are extracted for every cycle. 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 intersection 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, this work further squeezes false paths. Till now, the false paths are extracted primarily by using the information on non-active flops. On the other hand, additional paths from active flops can be identified as “false”. This possibility arises in multi-input combinational gates, e.g., AND2 in Fig. 10. 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 the path through U1/A can be used 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, this false path can be exploited 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 (AND-OR-INVERTER), OAI (OR-AND-INVERTER) 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 (half adder), and FA (full adder) do not have this characteristic since every input transition affects the output state.

In order to give a more concrete picture for explaining how to extract FPI and CF-FPI, Algorithms 1 and 2 show the pseudo codes for the processes utilized in this flow for

Algorithm 1: FPI for m-th mode

```
1: Run logic simulation to get the results with $m$-th mode.
2: Collect the flops with 0 toggle-rate through whole clock cycles
   $FF_{(j)}^{(m)}$, where $j = 1, 2, \ldots, F$.
3: FP_list = [ ]
4: for ($j = 1; j \le F; j++$)
5:     FP_list ← FP_list ∪ Paths-from-$FF_{(j)}^{(m)}$
6: Output: Design constraint file includes FP_list.
```

Algorithm 2: CF-FPI for m-th mode

```
1: Run logic simulation to get the results with $m$-th mode.
2: Based on the results, for $k$-th cycle ($k = 1, 2, \ldots, C$), collect
3: (1) 0 toggle-rate flops $FF_{(j)}^{(m,k)}$, where $j = 1, 2, \ldots, F^k$, and
4: (2) the nets with non-critical transition $N_{(x)}^{(m,k)}$, where
   $x = 1, 2, \ldots, M^k$.
5: FP_list = [ ]
6: for ($k = 1; k \le C; k++$)
7:     FP_list$^{(k)}$ = [ ]
8:     for ($j = 1; j \le F^k; j++$)
9:         FP_list$^{(k)}$ ← FP_list$^{(k)}$ ∪ Paths-from-$FF_{(j)}^{(m,k)}$
10:    for ($x = 1; x \le M^k; x++$)
11:        FP_list$^{(k)}$ ← FP_list$^{(k)}$ ∪ Paths-through-$N_{(x)}^{(m,k)}$
12:    if $k == 1$
13:        FP_list ← FP_list$^{(k)}$
14:    else
15:        FP_list ← FP_list ∩ FP_list$^{(k)}$
16: Output: Design constraint file includes FP_list.
```

extracting the false-path list in a certain mode. The inputs are all based on one-time simulation results based on that mode, and the output is the final false-path list (denoted as FP_list) that can be loaded by the EDA tool accompanied with MCMM feature for implementing mode-wise ASA. In Algorithm 1, the false paths are extracted from the non-active flops, where these flops have 0 toggle rate through the whole clock cycles in the m-th mode and the total number of these flops is denoted as F. **Paths-starting-from-$FF_{(j)}^{(m)}$** means that the set of paths which start from the j-th non-acitve FF in the m-th mode. With the help of EDA tool such as Synopsys PrimeTime or Design Compiler, it is available to trace the paths starting from the $FF_{(j)}^{(m)}$. CF-FPI shown in Algorithm 2 has a major difference compared with FPI in Algorithm 1 that it needs to extract the false paths cycle-by-cycle, and then performs an aggregation to find the intersect for all the cycle-based false-path lists. F^k in Algorithm 2 means the total number of the flops with 0 toggle-rate in the k-th cycle, and M^k means the total number of the nets with non-critical transition in the k-th cycle. **Paths-from-$FF_{(j)}^{(m,k)}$** means the set of paths starting from the flop $FF_{(j)}^{(m,k)}$, and **Paths-through-$N_{(x)}^{(m,k)}$** means the set of paths through the net $N_{(x)}^{(m,k)}$, where the set of paths can be extracted through the EDA tools.

5. Experimental Results and Analysis

5.1 Setup

This section performs an experimental evaluation of the proposed methodology with RISC-V processor, a popular open-source CPU. The processor is synthesized with Nangate 45nm library [29] for VTG (low-Vt) and VTH (high-Vt) cells at 0.5 GHz by Synopsys Design Compiler (DC) which enables MCMM, and V_{NOM} is set 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.

Our experiments select a processor as the evaluation platform, and then the programs of different workloads are compiled into different machine codes (0 and 1 sequences), which are used as the input patterns during logic simulation. Three workloads are selected in the experiments; (1) dijkstra and (2) sha, which are included in MiBenchmark [30], 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 mt-matmul-fp, which meets this work's assumption that the operating voltage and the power consumption might have a discrepancy between different modes. In this work, the area overhead is limited to be smaller than 0.5%. To be aware of MCMM fail cases during DSA, we simply set EXT to a constant 100 (W) in our experiments, where the estimated power of the processor is lower than 1 W.

For comparison, this experiment prepares 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 the baseline in this work. Method (B) applies ASA based on the false paths that are not activated in Mode 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 [6], [14]–[16] in terms of the number of modes while the stochastic treatment of timing error used in previous work [14]–[16] is disabled. Method (C) is the proposed approach described in Sect. 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

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%**)

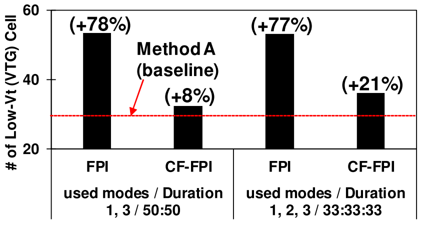
column represents the three methods (A), (B), (C). The second column lists how many modes are used and shows the combination of the used modes. Method (A) and (B) only apply one mode, but (B) considers the merged mode. The third column is the assumed duration that only (C) can explicitly consider when 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. The experiment applies 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 having the highest voltage. In this case, Method (B) is restricted 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, the proposed method can gain the power reductions of 16% and 20%, respectively while the area increases was merely 0.4%. On the other hand, the number of low-Vt (VTG) cells increased. 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.

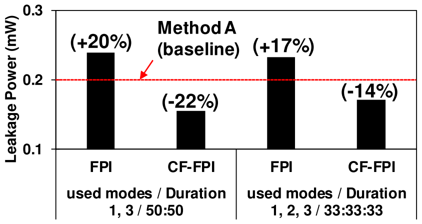
This work next investigates the impact of false path identification methods on the optimization results. Unfortunately, the MCMM flow with the false-path set of CF-FPI is slow and circuit optimization with CF-FPI could take 6∼7 hours for one run. Therefore, only the final iteration is executed with CF-FPI. Meanwhile, the run time varies with different designs and workloads, and CF-FPI flow may not be always so slow. Table 2 compares the number of commands for false path specification for the cases with and without applying CF-FPI. Looking at Mode 1 and Mode 2, obviously adopting CF-FPI increases the number of settings for assigning false paths by 25K and 30K, respectively. On the other hand, the number of false path settings for Mode 3 decreases by 29K. That is because many false stages have

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)



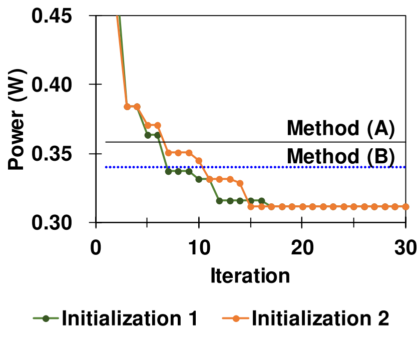
(b)

Fig. 11 Comparison between FPI and CF-FPI for (a) # of low-Vt (VTG) cells (b) leakage powers.

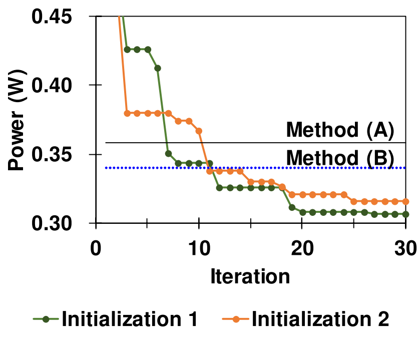
topologically upstream and downstream relationships, and thus we just neglect the setting for assigning the false paths through those upstream false stages. Figure 11(a) compares the usage of low-Vt (VTG) cells. FPI specifies fewer false paths, and then the timing closure is more strict, which results 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-V_t cells are used, and the increase of VTG cells is suppressed to only 8% to 21%. Since CF-FPI successfully suppress the usage of low-V_t cells, Fig. 11(b) reveals that applying CF-FPI reduces leakage power by 42% in the two-mode case and 31% in the three-mode case. However, the improvement of the dynamic (switching + internal) power from FPI to CF-FPI is minor since there is no significant difference in the obtained voltages, and thus the improvement of the total power (dynamic + leakage) is minor (0.03%) due to the small portion of leakage power to the overall power. On the other hand, when we apply CF-FPI to the design in which leakage power is dominant, the total power reduction might be significant. In overall, this result indicates that CF-FPI facilitates the timing optimization and reduces the number of lower-Vt cells.

5.3 Proposed Methodology Investigation

Figure 12(a) and Fig. 12(b) show how the solution is becoming convergent during DSA, where Fig. 12(a) supposes two-mode (1, 3) case of 50:50 duration and Fig. 12(b) supposes three-mode (1, 2, 3) case of 33:33:33 duration. In each fig-

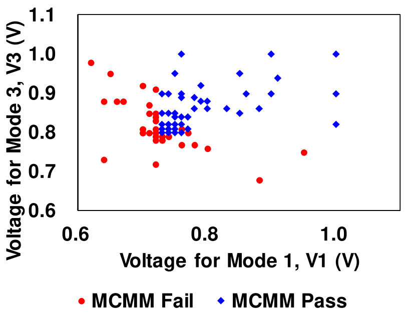


(a)

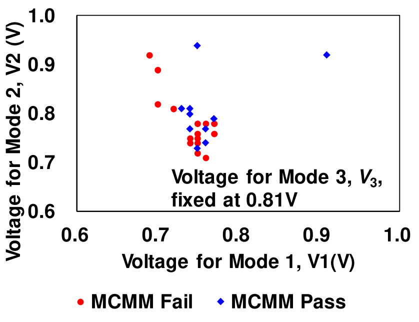


(b)

Fig. 12 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.



(a)



(b)

Fig. 13 Timing closure results at various sets of voltages for individual modes. (a) two-mode case (1, 3). and (b) three-mode case (1, 2, 3) where the voltage for Mode 3 is fixed at 0.81 V.

ure, two curves are plotted starting from different initial values. Initialization 1 applies the voltage set (V_1, V_2) of (1.0, 1.0), (0.9, 1.0), and (1.0, 0.9) as the start points in the two-mode optimization (associated with (x_{i1}, x_{i2}) for $i = 1 \sim 3$ mentioned in Sect. 4.1), 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 cases (associated with (x_{i1}, x_{i2}, x_{i3}) for $i = 1 \sim 4$ mentioned in Sect. 4.1) , i.e. voltage set (V_1, V_2, V_3) = (1.0, 1.0, 1.0), (1.0, 1.0, 0.9), (1.0, 0.9, 1.0), (0.9, 1.0, 1.0) for initialization 1 and (1.0, 1.0, 1.0), (1.0, 1.0, 0.85), (1.0, 0.85, 1.0), (0.85, 1.0, 1.0) for initialization 2. Figure 12(a) 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.

Figure 13(a) and Fig. 13(b) show the scatter plots of the voltages in two different modes evaluated in the optimization process, where the blackpoints mean that the timing closure fails in the MCMM flow and the blue points mean that the timing closure succeeds. Figure 13(a) is for two-mode case, and Fig. 13(b) 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, a clear bound could be found; 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 fixing one dimension of voltage V_3 at 0.81 V, the plot shows that the boundary is unclear and the blackand 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 paper proposed a design methodology based on ASA to achieve a design applying to mode-wise voltage-scaling (MWVS) with guaranteeing no timing errors. This work 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. The proposed method integrated ASA with the MCMM flow in EDA tools, and then applied DSA to solve the problem numerically. This work also introduced a fine-grained identification method of false paths that can be excluded in timing optimization without any risk of timing error. The evaluation based on RISC-V design achieved 20% gain of power efficiency compared with the conventional VS method, where 15% gain comes from the mode-wise idea. It is also shown that the fine-grained false path identification facilitated the timing closure and reduced leakage power by more than 30%. We believe that MWVS flow could offer more benefit on VS once combining the multi-mode voltage scaling with the stochastic MTTF analysis and consider it as our near future work.

References

[1] K. Shafique, B.A. Khawaja, F. Sabir, S. Qazi, and M. Mustaqim, "Internet of things (IoT) for next-generation smart systems: A review of current challenges, future trends and prospects for emerging 5G-IoT scenarios," IEEE Access, vol.8, pp.23022–23040, 2020.

[2] A. Dunkels, J. Eriksson, N. Finne, F. Österlind, N. Tsiftes, J. Abeillé, and M. Durvy, "Low-power IPv6 for the internet of things," 2012 Ninth International Conference on Networked Sensing (INSS), pp.1–6, 2012.

[3] M. Magno, L. Benini, C. Spagnol, and E. Popovici, "Wearable low power dry surface wireless sensor node for healthcare monitoring application," 2013 IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp.189–195, 2013.

[4] R. Yu and T. Watteyne, "Reliable, low power wireless sensor networks for the internet of things: Making wireless sensors as accessible as web servers," Linear Technology, Dec. 2013.

[5] P. Pawar, R. Nielsen, N. Prasad, S. Ohmori, and R. Prasad, "Hybrid mechanisms: Towards an efficient wireless sensor network medium access control," 2011 The 14th International Symposium on Wireless Personal Multimedia Communications (WPMC), pp.1–5, 2011.

[6] 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.

[7] K. Lee, S.-J. Lee, and H.-J. Yoo, "Low-power network-on-chip for high-performance SoC design," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.14, no.2, pp.148–160, 2006.

[8] S. Das, D. Roberts, S. Lee, S. Pant, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "A self-tuning dvs processor using delay-error detection and correction," IEEE J. Solid-State Circuits, vol.41, no.4, pp.792–804, 2006.

[9] U. Jinbo, J. Yamada, R. Shioya, and M. Goshima, "Applying razor flip-flops to sram read circuits," IEICE Trans. Electron., vol.E100-C, no.3, pp.245–258, 2017.

[10] 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.

[11] A. Drake, R. Senger, H. Deogun, G. Carpenter, S. Ghiasi, T. Nguyen, N. James, M. Floyd, and V. Pokala, "A distributed critical-path timing monitor for a 65 nm high-performance microprocessor," 2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers, pp.398–399, 2007.

[12] Y. Kunitake, T. Sato, and H. Yasuura, "A replacement strategy for canary flip-flops," 2010 IEEE 16th Pacific Rim International Symposium on Dependable Computing, pp.227–228, 2010.

[13] Y. Kunitake, T. Sato, H. Yasuura, and T. Hayashida, "Possibilities to miss predicting timing errors in canary flip-flops," 2011 IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS), pp.1–4, 2011.

[14] Y. Masuda and M. Hashimoto, "MTTF-aware design methodology of adaptively voltage scaled circuit with timing error predictive flip-flop," IEICE Trans. Fundamentals, vol.E102-A, no.7, pp.867–877, July 2019.

[15] Y. Masuda, M. Hashimoto, and T. Onoye, "Critical path isolation for time-to-failure extension and lower voltage operation," 2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp.1–8, 2016.

[16] Y. Masuda, T. Onoye, and M. Hashimoto, "Activation-aware slack assignment for time-to-failure extension and power saving," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.26, no.11, pp.2217–2229, 2018.

[17] Y. Masuda, J. Nagayama, T. Cheng, T. Ishihara, Y. Momiyama, and M. Hashimoto, "Critical path isolation and bit-width scaling are highly compatible for voltage over-scalable design," 2021 Design, Automation and Test in Europe Conference (DATE), pp.1260–1265, 2021.

[18] 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 Trans. Comput.-Aided Design Integr. Circuits Syst., vol.26, no.11, pp.1947–1956, 2007.

[19] A.B. Kahng, S. Kang, R. Kumar, and J. Sartori, "Slack redistribution for graceful degradation under voltage overscaling," 2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC), pp.825–831, 2010.

[20] A.B. Ahmed, D. Fujiki, H. Matsutani, M. Koibuchi, and H. Amano, "AxNoC: Low-power approximate network-on-chips using critical-path isolation," 2018 Twelfth IEEE/ACM International Symposium on Networks-on-Chip (NOCS), pp.1–8, 2018.

[21] K. Roy and A. Raghunathan, "Approximate computing: An energy-efficient computing technique for error resilient applications," 2015 IEEE Computer Society Annual Symposium on VLSI, pp.473–475, 2015.

[22] Q. Xu, T. Mytkowicz, and N.S. Kim, "Approximate computing: A survey," IEEE Des. Test, vol.33, no.1, pp.8–22, 2016.

[23] T. Cheng, Y. Masuda, J. Nagayama, Y. Momiyama, J. Chen, and M. Hashimoto, "Mode-wise voltage-scalable design with activation-aware slack assignment for energy minimization," Proc. 26th Asia and South Pacific Design Automation Conference, ASPDAC'21, New York, NY, USA, pp.284–290, Association for Computing Machinery, 2021.

[24] D. Flynn, Low Power Methodology Manual: For system-on-chip design, 1st. ed., Springer Science & Business Media, 2007.

[25] A.B. Kahng, S. Kang, R. Kumar, and J. Sartori, "Enhancing the efficiency of energy-constrained dvfs designs," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.21, no.10, pp.1769–1782, 2013.

[26] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, Numerical Recipes: The Art of Scientific Computing, 3rd ed., Cambridge University Press, USA, 2007.

[27] L. Baudzus and P.M. Krummrich, "Modified downhill simplex method for fast adaption of optical filters in optical communication systems," Electron. Lett., vol.55, no.8, pp.471–473, 2019.

[28] Á. Bűrmen, J. Puhan, and T. Tuma, "Grid restrained nelder-mead algorithm," Comput. Optim. Applic., vol.34, pp.359–375, 2006.

[29] http://www.nangate.com/index.php, 2009.

[30] 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.

TaiYu Cheng received the B.E. and M.E. degrees in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2010 and 2012, respectively. From 2012 to 2018, he was with Taiwan Semiconductor Manufacturing Company, Hsinchu, Taiwan, where he has been engaged in design flow of timing closure. He received the Ph.D. degree in the Department of Information Systems Engineering, Osaka University, Osaka, Japan in 2021. His research interests include low-power circuit design and circuits, architectures, and systems for machine learning.

Yutaka Masuda received the B.E., M.E., and Ph.D. degrees in Information Systems Engineering from the Osaka University, Osaka,Japan, in 2014, 2016, and 2019, respectively. He is currently an Assistant Professor in Center for Embedded Computing Systems, Graduate School of Informatics, Nagoya University. His research interests include low-power circuit design. He serves on the Technical Program Committee of international conferences including ASP-DAC. He is a member of IEEE, IEICE, and IPSJ.

Jun Nagayama received the B.S. and M.S. degrees in physics from Sophia University, Tokyo, Japan, in 2002 and 2004, respectively. In 2004, he joined the Fujitsu Ltd., Kawasaki, Japan. In 2015, he moved the Socionext Inc., Yokohama, Japan. He has been engaged in research and development of low-power CMOS design.

Youichi Momiyama received the B.S. and M.S. degree in electronics engineering from Niigata University, Niigata, Japan, in 1990 and 1992, respectively. In 1992, he joined Fujitsu Laboratories Ltd., Atsugi, Japan, where he has been engaged in research and development of low-power and high-speed CMOS devices. He moved to SOCIONEXT Inc. at 2015, where he has been investigating CMOS low-power design. His research interests include CMOS low-power enablement. He received the Best Paper Award at the 1st conference of IEEE IWJT. He has been serving as editor of IEEE Transaction on Electron Devices since 2011. Mr. Momiyama is a member of the IEEE electron devices society, solid-state circuits society and IEICE.

Jun Chen received the B.E. and M.E. degrees in control theory and engineering from Tongji University, Shanghai, China, in 2004 and 2007, respectively. From 2008 to 2016, he was with Synopsys Inc., Shanghai, China, where he has been engaged in research and development of routing congestion, power and placement optimization flow. He received the Ph.D. degree in the Department of Information Systems Engineering, Osaka University, Osaka, Japan in 2020. His research interests include computer-aided-design for digital integrated circuits, power and signal integrity analysis.

Masanori Hashimoto received the B.E., M.E. and Ph.D. degrees in communications and computer engineering from Kyoto University, Kyoto, Japan, in 1997, 1999, and 2001, respectively. He is currently a Professor in Graduate School of Informatics, Kyoto University, Kyoto, Japan. His current research interests include the design for manufacturability and reliability, timing and power integrity analysis, reconfigurable computing, soft error characterization, and low-power circuit design. Dr. Hashimoto was a recipient of the Best Paper Awards from ASP-DAC in 2004 and RADECS in 2017, and the Best Paper Award of the IEICE Transactions in 2016. He was on the Technical Program Committee of international conferences, including DAC, ICCAD, ITC, Symposium on VLSI Circuits, ASP-DAC, and DATE. He serves/served as an Associate Editor for the IEEE Transactions on VLSI Systems, IEEE Transactions on Circuits and Systems I, ACM Transactions on Design Automation of Electronic Systems, and Elsevier Microelectronics Reliability.