ADAPTIVE INTEGRATED CIRCUIT DESIGN FOR VARIATION RESILIENCE AND SECURITY

A Dissertation

by

JIAFAN WANG

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2017

Major Subject: Computer Engineering

ABSTRACT

The past few decades witness the burgeoning development of integrated circuit in terms of process technology scaling. Along with the tremendous benefits coming from the scaling, challenges are also presented in various stages. During the design time, the complexity of developing a circuit with millions to billions of smaller size transistors is extended after the variations are taken into account. The difficulty of analyzing these non-deterministic properties makes the allocation scheme of redundant resource hardly work in a cost-efficient way. Besides fabrication variations, analog circuits are suffered from severe performance degradations owing to their physical attributes which are vulnerable to aging effects. As such, the post-silicon calibration approach gains increasing attentions to compensate the performance mismatch. For the user-end applications, additional system failures result from the pirated and counterfeited devices provided by the untrusted semiconductor supply chain. Again analog circuits show their weakness to this threat due to the shortage of piracy avoidance techniques.

In this dissertation, we propose three adaptive integrated circuit designs to overcome these challenges respectively. The first one investigates the variability-aware gate implementation with the consideration of the overhead control of adaptivity assignment. This design improves the variation resilience typically for digital circuits while optimizing the power consumption and timing yield. The second design is implemented as a self-validation system for the calibration of diverse analog circuits. The system is completely integrated on chip to enhance the convenience without external assistance. In the last design, a classic analog component is further studied to establish the configurable locking mechanism for analog circuits. The use of Satisfiability Modulo Theories addresses the difficulty of searching the unique unlocking pattern of non-Boolean variables.

ACKNOWLEDGMENTS

Looking back on my five years' pursuit of the doctoral degree, I experienced the confusion at the first year in Texas A&M University, the frustration due to the bottle neck of researches, and the doubt of personal ability after the failure. It is hardly for me to struggle to the end without the help from different people.

First and foremost, I would like to thank my advisor, Professor Jiang Hu, who helps me tremendously both in research and in life. Knowledge is not the only thing he conveys to me but also the rigorous ethic of the work, strict demands on oneself, and tolerant attitude to others. It's my honor to be his student.

Moreover, I wish to express the sincere gratitude to my co-advisor, Professor Edgar Sánchez-Sinencio. He provides me the golden opportunity to prove my idea by chip fabrication and forgives my mistake during the first access to the interdisciplinary project. His trust guarantees our final success.

I also need to give grateful thanks to my lab mates, Hao He, Congyin Shi, Sanghoon Lee, and Adriana Sanabria-Borbon. Working with them in different challenging projects are the best memories of my academic life. They make me believe that team work could move around any obstacles.

I appreciate my parents. They have had no idea of my research since I left them fifteen years ago. However, they always stood by my side whenever I needed their suggestions, financial aid, or just a telephone greeting. I know I shall feel for ever in their debt.

Last but not the least, I want to thank my wife Xuefen Cheng, who gave up her own promising career just to support my academic dream. Her companion and encouragement are the impetuses to make me through the most difficult time during the international study period.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a dissertation committee consisting of Professor Jiang Hu [advisor], Edgar Sánchez-Sinencio [co-advisor], and Peng Li of the Department of Electrical & Computer Engineering and Professor Duncan M. (Hank) Walker of the Department of Computer Science and Engineering.

All work for the dissertation was completed by the student, in collaboration with Hao He, Congyin Shi and Adriana Sanabria-Borbon of the Department of Electrical & Computer Engineering.

Funding Sources

The dissertation research is partially supported by National Science Foundation (CCF-1255193, CNS-1618824) and Semiconductor Research Corporation (2013-TJ-2421, 2016-TS-2688).

NOMENCLATURE

VLSI	Very Large Scale Integration
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
LR	Lagrangian Relaxation
ABB	Adaptive Body Bias
FBB	Forward Body Bias
SSTA	Statistical Static Timing Analysis
CUT	Circuit Under Test
ADC	Analog-to-Digital Converter
BPF	Band Pass Filter
SA	Simulated Annealing
SS	Sensitivity Search
CORDIC	COordinate Rotation DIgital Computer
OTA	Operational Transconductance Amplifier
CCM	Configurable Current Mirror
SAT	Boolean Satisfiability Problem
SMT	Satisfiability Modulo Theories
PUF	Physically Unclonable Function

TABLE OF CONTENTS

Page

ABSTRACT ii

ACKNOWLEDGMENTS iii

CONTRIBUTORS AND FUNDING SOURCES iv

NOMENCLATURE v

TABLE OF CONTENTS vi

LIST OF FIGURES ix

LIST OF TABLES xiii

1. INTRODUCTION OF CIRCUIT ADAPTIVITY IN VERY LARGE SCALE INTEGRATION 1

1.1 Significance of Circuit Adaptivity 1
1.2 Difficulties and Algorithms 2

2. OPTIMIZATION WITH ADAPTIVE CIRCUIT DESIGN 4

2.1 Introduction 4
2.2 Background 6
2.2.1 Timing Constraint and Gate Implementation 6
2.2.2 Adaptive Body Biasing 8
2.2.3 Adaptivity Assignment Scheme 10
2.2.4 Lagrangian Relaxation 11
2.2.5 Application in Gate Implementation Selection 12
2.2.6 Process Variation Modeling 14
2.3 Problem Formulation 15
2.3.1 Overview of Algorithm Flow 15
2.3.2 Variation-aware Gate Implementation 17
2.3.3 Variation-aware Lagrangian Relaxation 19
2.4 Subproblem Solution 20
2.4.1 Backward Search with Inferior Pruning 20
2.4.2 Solution Inconsistence and Forward Restoration 22

2.4.3 Iterative Refinement 23
2.4.4 Solution for Redundant Counting 24
2.5 Dual Problem Solution 26
2.6 Experiment Results 28

3. BUILT-IN SELF OPTIMIZATION FOR VARIATION RESILIENCE OF ANALOG FILTERS 34

3.1 Introduction 34
3.2 Background 36
3.2.1 System Architecture 36
3.2.2 Programmable Bandpass Filter 37
3.2.3 Non-ideal Effects 39
3.3 Cost Function Design and Implementation 40
3.3.1 Optimization Cost Function 40
3.3.2 Circuit Implementation 42
3.3.3 CORDIC Algorithm Based Calculation 45
3.4 Optimization Engine and Implementation 46
3.4.1 Hybrid Optimization Algorithm 46
3.4.2 State Machine Diagram 49
3.5 Reconfigurable Circuit Design and Implementation 51
3.5.1 Post-silicon Configuration 51
3.5.2 General Microprocessor 52
3.6 Experiment Results for Non-Reconfigurable Circuit 54
3.6.1 Test Chip Measurement Results 54
3.6.2 Evaluation of Variation Resilience 55
3.6.3 Verification of Reconfigurable Circuit 57
3.6.4 Comparison of Algorithm Performance 60
3.6.5 Die Photographs for Tapeout Chips 63

4. THWARTING ANALOG IC PIRACY VIA COMBINATIONAL LOCKING. . . 66

4.1 Introduction 66
4.1.1 Previous Works 68
4.2 Overview and Scope of This Work 69
4.3 Attack Analysis 71
4.4 Current Mirror and Its Role in Analog ICs 73
4.4.1 Basic MOSFET Current Mirror 73
4.4.2 Importance of Bias Current 75
4.4.3 Application in Gm-C Band Pass Filter 76
4.5 Configurable Current Mirror for Locking 78
4.5.1 Difficulties of Naïve Configurable Design 78
4.5.2 A General Locking Architecture 80

4.5.3 Locking Design by Satisfiability Modulo Theories 82
4.5.4 Formulation of SMT Constraints 83
4.6 Experiment Results and Discussion 85
4.6.1 Experiment Result of BPF (Band-Pass Filter) 85
4.6.2 Experiment Result of Quadrature Oscillator 89
4.6.3 Experiment Result of LC Oscillator 90
4.6.4 Experiment of Class-D Amplifier 92
4.6.5 Security Protection Level and Attack Analysis 93
4.6.6 Area and Design Overhead 96

5. SUMMARY AND CONCLUSIONS 98

REFERENCES 99

LIST OF FIGURES

FIGURE Page

1.1 Circuit adaptivity in different scenarios. 3

2.1 Static timing analysis. (a) Arrival time analysis by forward traversal from primary inputs; (b) Required time analysis by backward traversal from primary outputs. 7

2.2 Adaptive circuit with the control of body biasing. 9

2.3 Different adaptivity assignment schemes. (a) Over-design in gate implementation selection (gate sizing and V_{th} assignment); (b) Under-design and coarse-grained adaptivity; (c) Under design and fine-grained adaptivity; (d) Collaborative gate implementation selection and fine-grained adaptivity. Reprinted from [1]. 10

2.4 Chip die with 9 grids and its corresponding 9×9 covariance matrix. . . . 15

2.5 Overview of the adaptivity-aware gate implementation selection algorithm flow. Reprinted from [1]. 16

2.6 Backward topological traverse with inferior pruning. 20

2.7 Forward topological traverse with solution back trace. 21

2.8 Forward topological traverse with solution inconsistence. 22

2.9 Forward topological traverse with solution restoration. 23

2.10 Backward topological traverse followed by iterative refinement. 24

2.11 Backward topological order solution propagation on DAG. (a) Cost Ld is double-counted at node a. (b) Conventional approach to avoid double counting. (c) Propagation only on spanning tree (in solid edges). Reprinted from [1]. 25

2.12 Power/area-timing tradeoff for circuit c7552. Reprinted from [1]. 33

2.13 Power/area vs. granularity for circuit fft. Reprinted from [1]. 33

3.1 Overview of the proposed built-in self optimization system architecture. Reprinted from [2]. 37

3.2 Second-order band-pass filter and the configurable resistance/capacitor array structure. Reprinted from [2]. 38

3.3 Cost function circuit diagram. Each gray rectangle is an arithmetic block, and each white rectangle is a register. "Reg" implies the resistor, while "Mux21" and "Mux41" represent the 2-to-1 and 4-to-1 multiplexers, respectively. 43

3.4 Specific processes during the multi-start meta-heuristic. (a) random update to SA neighbor (x_1', x_2') based on the old solution and temperature T in SA. (b) curve of SA acceptance probability based on $\Delta\phi$ and the temperature T. (c) all four neighbors around the $\vec{x}_{best}$ during the SS. (d) LFSR based pseudo random process. 49

3.5 The high-level state machine of multi-start meta-heuristic, in which the upper multi-start SA and the lower sensitivity-based searches are divided by the horizontal dashed line. Reprinted from [2]. 50

3.6 Components of the designed microprocessor and its working flow. 52

3.7 High-level state machine of the microprocessor and 16-bit instructions. . . 53

3.8 Cost function $\Phi(\vec{x})$ for decision variables $\vec{x} = (x_1, x_2)$ corresponding to R_{x_1} and R_{x_2} in the BPF with $f_c = 31MHz$ and $BW = 8MHz$. The measurements are based on the test chip. Reprinted from [2]. 55

3.9 Frequency responses under different decision variables. The design target is the BPF with $f_c = 31MHz$ and $BW = 8MHz$. Reprinted from [2]. . . 56

3.10 Probability density functions of frequency response mean squared error (MSE) before and after the self optimization from the $5,000$-run Monte Carlo simulation on BPF with $f_c = 25MHz$ and $BW = 15MHz$. Reprinted from [2]. 56

3.11 Cost function $\Phi(\vec{x})$ for decision variables $\vec{x} = (x_1, x_2)$ corresponding to R_{x_1} and R_{x_2} in the BPF with $f_c = 74MHz$ and $BW = 13MHz$. The measurements are based on the test chip. 57

3.12 Cost function curve calculated by the microprocessor with the same configuration in Figure 3.11. 58

3.13 Search trace and optimal point found by the microprocessor. 59

3.14 Cost function changes over iterations for our multi-start meta-heuristic, multi-start sensitivity and standalone SA on BPF with $f_c = 23MHz$ and $BW = 6MHz$. Reprinted from [2]. 60

3.15 Cumulative distribution functions of solution errors vs. the optimal solution from the simulation performed on 4,000 BPFs. Reprinted from [2]. . 61

3.16 Chip die photograph of the proposed built-in self-optimization system. CUT includes the active-RC BPF, and the digital circuits are fully integrated in one block. 63

3.17 Chip die photograph of the reconfigurable optimization system including the microprocessor and on-chip SRAM 1024x8 bit. 64

4.1 Locking system overview. Reprinted from [3]. 70

4.2 Basic MOSFET current mirror with five finger branches. Reprinted from [3]. 74

4.3 Gm-C implementation of BPF with differential amplifiers. Reprinted from [3]. 76

4.4 A naïve design of configurable current mirror. Reprinted from [3]. 78

4.5 An example of the proposed locking architecture. Reprinted from [3]. . . 80

4.6 Bias currents from different keys for three current mirrors in the 4th order BPF. Reprinted from [3]. 86

4.7 Normalized frequency responses of the 4th-order BPF with $f_c = 250kHz$ and $BW = 150kHz$, for different keys. Reprinted from [3]. 87

4.8 Only one key makes the BPF satisfying $f_c = 250kHz$, $BW = 150kHz$ and $Amplitude = 0dB$. Reprinted from [3]. 88

4.9 Normalized oscillation frequencies of the quadrature oscillator with $f_{osc} = 2.34MHz$, affected by different keys. Reprinted from [3]. 89

4.10 Oscillation frequency and amplitude histogram of the quadrature oscillator with $f_{osc} = 2.34MHz$, for different keys. Reprinted from [3]. 90

4.11 Oscillation amplitudes of the LC oscillator with $f_{osc} = 2GHz$ for all combinations of 6-bit key for CCM7. Reprinted from [3]. 91

4.12 Triangular waveforms of the generator for different keys for two current mirrors, CCM9 and CCM10. Reprinted from [3]. 92

4.13 Amplitude error histogram of the triangle generator with $\Delta = 30\%$ and $\Theta = \infty$, for different keys. Reprinted from [3]. 93

4.14 MSE curve in the second stage of the 4th order BPF satisfying $f_c = 250kHz$, $BW = 150kHz$ and $Amplitude = 0dB$. Reprinted from [3]. . . 95

LIST OF TABLES

TABLE Page

2.1 Experimental results of Naïve method with only forward body bias (FBB). Total area overhead and power overhead are denoted by ΔA and ΔW, respectively. Reprinted from [1]. 29

2.2 Experimental results of our method with only forward body bias (FBB). Gate area, total area and power overhead are denoted by ΔA_g, ΔA, and ΔW, respectively. Reprinted from [1]. 30

2.3 Experimental results of Naïve method with forward body bias and reverse body bias (ABB). Total area and power overhead are denoted by ΔA and ΔW, respectively. Reprinted from [1]. 31

2.4 Experimental results of our method with forward body bias and reverse body bias (ABB). Gate area, total area and power overhead are denoted by ΔA_g, ΔA and ΔW, respectively. Reprinted from [1]. 32

3.1 Usage of registers within the summation circuit. 44

3.2 Gate count, power and processing time comparison. Reprinted from [2]. . 62

3.3 SRAM memory space consumption for microprocessor. 62

4.1 Area overhead and key bitwidth. Reprinted from [3]. 96

4.2 Runtime for solving the SMT for different bitwidths. Reprinted from [3]. . 97

1. INTRODUCTION OF CIRCUIT ADAPTIVITY IN VERY LARGE SCALE INTEGRATION

1.1 Significance of Circuit Adaptivity

In adaptive integrated circuits (ICs), the word “adaptive” implies the feature of circuit modifications after the hardware fabrication. As the increasingly scaling of IC process technology, the necessity of this feature is enhanced by the exacerbation of parametric variations in transistors’ attributes, such as geometry size, oxide thickness, doping density, etc. These interior mismatches further cause the measurable loss to the output performance.

For example, when process technology scales down from 350 nm to 90 nm, chip yields reported by the foundries have reduced from nearly 90% to 50% [4]. This situation is even worse at Taiwan Semiconductor Manufacturing Company’s (TSMC’s) 10 nm process, such that a unexpected low yield rate for Apple’s A10X chips is likely to disrupt the production schedule of iPad in March 2017 [5]. Besides process variations, aging effects manifest themselves as another major source of characteristic changes of devices, particularly analog circuits. For a Class A amplifier designed in a 65 nm process, the simulation based on the Channel Hot Carrier degradation (CHC) model indicates a significant gain loss of 15% over ten years’ lifetime [6]. Moreover, the early in-use system failures caused by the pirated and counterfeited components could be regarded as a severe form of performance degradation, which has now turned into an emerging threat to the global IC security. Again, a study by IHS technology [7] presents that analog ICs rank the first out of five most counterfeited semiconductors due to their relative large size compared to digital circuits and the shortage of piracy avoidance techniques.

To overcome the above challenges, circuit adaptivity shows its significance in different stages of the very large scale integrated (VLSI) circuit design. For example, during the design time, the adaptivity circuits, with the ability of body bias control or voltage adaptation, are technically introduced to the original system. They will be invoked later by the timing violation detectors at runtime and compensate the performance degradation by modifying the body bias or supply voltage. Another scenario to apply the adaptivity is the post-fabrication tuning. Essentially relying on the in-situ configurable structure, circuit performance could be calibrated towards the designed specifications whenever it's needed before the advent of final wear-out stage. Furthermore, the circuit adaptivity may contribute to the field of hardware security. Similar to the digital lock, the adaptive circuit won't enable the whole design until the correct keys are provided. Otherwise it locks the system by malfunction or even entire breakdown.

1.2 Difficulties and Algorithms

Despite the significance of circuit adaptivity, to figure out the appropriate ways of considering the adaptivity is not an easy task under different situations. For instance, to compensate the variations during the design time, the conventional optimization has to be re-formulated with the introduction of the variation property. The run-time calibration method will be limited to the area consumption of the tuning strategy if it is implemented on chip. And the protection level of the key-lock scheme will be weakened in case the configurable circuits are not well designed.

In this dissertation, we proposed systematic ways to assign the circuit adaptivity. In Chapter 2, a Lagrangian Relaxation based algorithm is further improved to optimize the gate implementation under the process variations. Body bias is applied as an additional dimension to minimize the system power assumption, while the adaptivity overhead is also handled in the optimization function. The stochastic evaluation is executed to analyzed the circuit timing to maintain the performance with a high probability. In Chapter 3, a

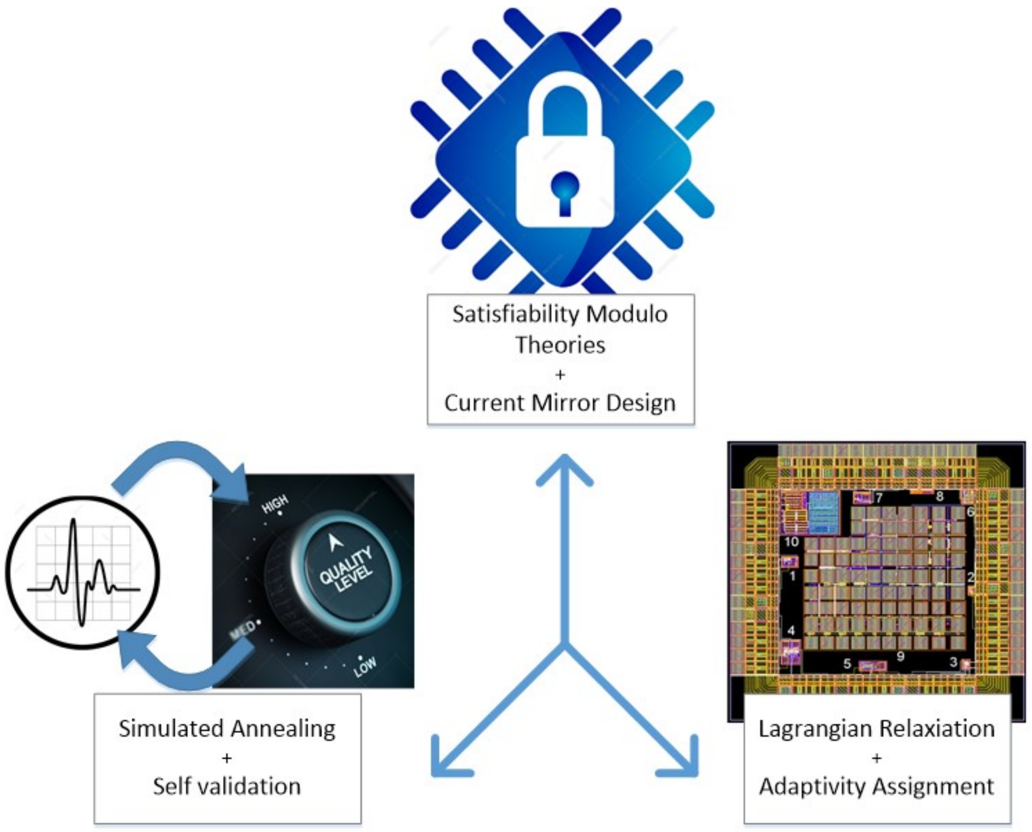


Figure 1.1: Circuit adaptivity in different scenarios.

self calibration design is implemented on-chip to adjust the performance of analog circuits towards the designed specifications. The calibration circuit is based on meta-heuristic with the balance between computation complexity and searching quality. The effectiveness of the circuit is demonstrated by both the chip measurement and simulation. In Chapter 4, inspired from the variation-sensitive analog circuits, we upgrade the common used current mirror as a locking structure to supplement the lack of protection to the analog system. Depending on the satisfiability module theories, the locking structure is so well designed that it could only be opened by a unique key out of numerous candidates, while all other keys will lead to the system malfunction or totally breakdown.

2. OPTIMIZATION WITH ADAPTIVE CIRCUIT DESIGN*

2.1 Introduction

As a power-efficient approach to variation resilience, adaptive circuit design has been demonstrated the effectiveness by results from test chips using body biasing [8], voltage interpolation [9], circuit reconfiguration [10] or a combination of them. The purpose of these changes is to compensate the performance variability due to manufacturing process variations [8], device aging effect [11], and thermal fluctuations [12] while to bring power savings to the whole circuit.

Although numerous adaptive design techniques have been reported, the adaptivity overhead is only mentioned in several previous works [10, 13, 14], and has rarely been a main emphasis. Actually, the overhead of adaptive design highly depends on its granularity. Coarse-grained adaptivity, such as uniform adaptivity for an entire processor core, has relatively small amortized overhead. For example, if a processor core has only one critical path replica based sensor, the area overhead is only 0.2% [15]. When intra-die variations are more pronounced [16], fine-grained adaptivity [9, 14] (in blocks of hundreds or thousands of gates) allows the compensation to be applied in a pinpoint manner. Evidently, fine-grained adaptivity tends to entail large overhead of sensors, voltage regulators and control circuits. This can cause as much as 50% area overhead for voltage interpolation [13] and 20% in [10, 14].

In addition to the area overhead, the effectiveness of power saving also largely depends on the granularity. For example, many variation effects are intrinsically fine-grained. Therefore, two adjacent transistors may have different doping densities due to process fluctuations. If adaptivity is coarse-grained, e.g., an entire processor core adapts uniformly, the

*Reprinted from "Collaborative Gate Implementation Selection and Adaptivity Assignment for Robust Combinational Circuits", by Hao He, Jiafan Wang, Jiang Hu, ©2015 IEEE.

adaptation must aim at transistors with the most critical timing requirement on the core. Evidently, this is hardly an efficient use of power. By contrast, fine-grained adaptivity, in blocks of hundreds or thousands of gates, speeds up the critical components with the cost of increasing additional power within a small region while keeps other non-critical components working in a large and low power state, thus enables power saving to the whole circuit in spite of the large area overhead.

Obviously, one prefers the power savings from fine-grained adaptivity but not its large overhead. Nevertheless, there are very few works on the optimization of adaptive circuit, especially the control of its overhead. One attempt is to perform gate clustering. In [17], an algorithm is introduced for joint design-time and post-silicon tuning optimization, but with little attention on the overhead issue. Moreover, it assumes that gate size can be continuously changed while most of modern designs are based on highly discrete cell libraries. The objective of [18, 19] is to minimize the overhead of adaptive body biasing (ABB). These works attempt to cluster gates with similar timing criticality and variability and are very useful in deciding adaptivity granularity, but they assume ABB is applied to all clusters. Another work [20] restricts variation sensors only at timing critical paths so that the overhead is not excessively large. However, it does not consider control or voltage generation overhead. Variability-aware discrete gate sizing is discussed in [21, 22]. These works are focused on how to propagate statistical timing information during sizing without much emphasis on the optimization aspect.

In this chapter, we develop a general algorithm framework for the optimization of adaptive circuit design with the consideration of overhead control [1]. Since adaptivity optimization is closely related with gate implementation selection (gate sizing and threshold voltage assignment), we perform them jointly such that their efforts are concerted with each other. Evidently, variations must be accounted and make the optimization problem rather difficult. We make use of Lagrangian relaxation that solves a multi-objective problem in two layers – subproblem and dual problem. The subproblem is focused on solution search while the dual problem can employ variability-aware models to guide the tradeoff among multiple objectives. As such, accurate models are used in a lightweight manner without causing too long runtime. The main advantages of our approach includes the following:

1. Compared to [17], which is restricted to linear and continuous models, our work is a discrete approach and compatible with realistic models in industry.
2. Our work provides a relatively complete adaptivity assignment solution for general adaptive circuit designs while the works of [18, 19] focus on clustering and ABB.
3. A new technique to solve the over-counting problem with less cost estimate error is proposed in our work, without increasing the overall algorithm complexity.
4. Area overhead is explicitly handled in our work but neglected in [17]. Experimental results on benchmark circuits confirm the effectiveness of our method.

2.2 Background

2.2.1 Timing Constraint and Gate Implementation

One target of the physical design of digital circuits is to derive the timing satisfied circuit, which defines that all signals arriving at the primary inputs at time a should reach the primary outputs no later than q, given the gate delay d of each cell. Since the timing information is only given at the primary inputs and primary outputs, the satisfaction of the whole circuit is guaranteed by the sufficient and necessary condition that all the arrival times derived from a in Figure 2.1 (a) shouldn't be later than the required arrival times derived from q in Figure 2.1 (b) at every net within the circuit. Since the delay is fixed when the circuit design is done, this analysis method is also named as the Static Timing Analysis (STA).

Need to mention that, for the multi-fanin gate, signals might arrive at the input pins of

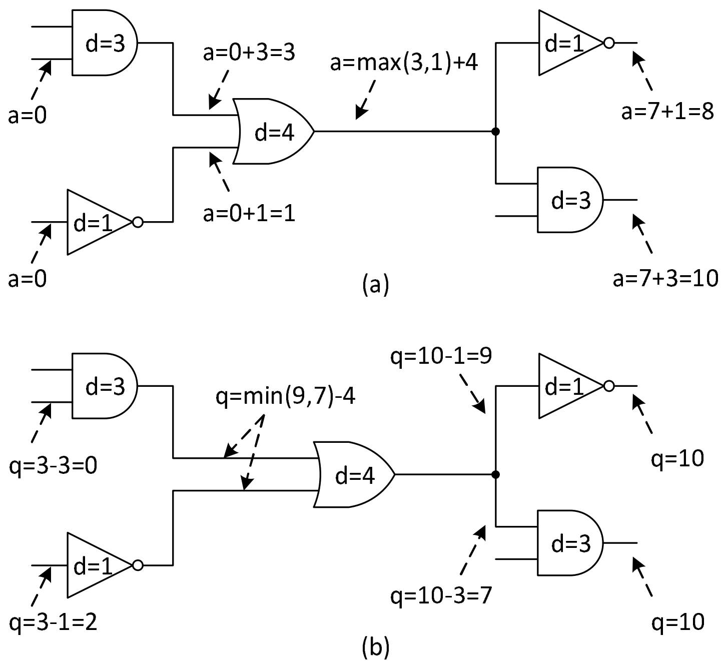


Figure 2.1: Static timing analysis. (a) Arrival time analysis by forward traversal from primary inputs; (b) Required time analysis by backward traversal from primary outputs.

v_m at different times in Figure 2.1 (a). Here the latest arrival time at the output of this gate is considered by $a_m = max_{i \in fanin(m)}(a_i + d_m)$, which could make sure the timing analysis work under the worst case signal propagation. Similar situation happens in Figure 2.1 (b) for the gate v_m due to its multiple fanouts. Still followed by the worst case guarantee of signal propagation, the required arrival time at the input of this gate is obtained by $q_m = min_{j \in fanout(m)}(q_j - d_m)$. Therefore, the time slack defined by $(q - a)$ at each point in the circuit could be used as a variable to evaluate the gate implementation.

Gate implementation is a classic technique during the physical design of digital circuits. It determines the gate size and threshold voltage of each gate so as to minimize the total power consumption of the circuit subject to the timing constraints. Under different implementations of each gate size and threshold voltage, the delay information d of an individual gate in Figure 2.1 will vary, and then affect the global timing status of the circuit.

2.2.2 Adaptive Body Biasing

ABB technique is based on the Metal Oxide Semiconductor Field Effect Transistor (MOSFET) body effect, which refers to the change in the transistor threshold voltage V_{th}. An analytical equation (2.1) shows the dependence of V_{th} on the source-to-body bias V_{SB} as follow.

$$V_{th} = V_{th0} + \gamma(\sqrt{|V_{SB} - 2\phi_F|} - \sqrt{|2\phi_F|}), \;\; V_{SB} = V_S - V_B \tag{2.1}$$

where V_{th0} is the threshold voltage with zero body bias ($V_{SB} = 0$) and is achieved by tying the substrate on the die to the source, e.g. power supply V_{DD} (in case of PMOS) or ground (in case of NMOS). ϕ_F and γ are the Fermi potential and the body effect parameter, respectively.

The effect of V_{th} to the subthreshold leakage current I_{subn} and the gate delay d is further revealed in [18]

$$\begin{aligned} I_{subn} &= K_1 e^{\frac{V_{GS}-V_{th}}{n\nu_T}} \\ d &= \frac{K_2}{(V_{DD} - V_{th})^{\alpha}} \end{aligned} \tag{2.2}$$

where α is the velocity saturation index, ν_T is the thermal voltage and n is the subthreshold swing coefficient. K_1 and K_2 are the factors related to the original subthreshold leakage current and gate delay, respectively.

As for the total power model, we apply $P = P_{dyn} + P_{leakage}$, where P_{dyn} is the dynamic power, and $P_{leakage}$ includes the leakage mechanisms affected by V_{th}, especially the subthreshold leakage $P_{subn} = V_{DD} I_{subn}$. For other leakage components, such as body-source/drain junction diode leakage, and band-to-band tunneling, we didn't consider them in our design, yet the optimization method introduced in Section 2.3 could be extended by adding them to the target function.

Thus, by manipulating the body voltage V_B, the adaptive circuit which consists the delay sensor and voltage controller is able to adjust all the V_{th} of NMOS gates in the dashed line rectangle of Figure 2.2, and then affects the leakage power and circuit delay in that area. For example, a positive body voltage ($V_B > 0$) will decrease the adjusted threshold voltage V_{th} and result in a larger current I_{subn} and smaller transistor delay d according to (2.2). This is called the Forward Body Biasing (FBB), while the similar analysis can be applied to Reverse Body Biasing (RBB) if $V_B < 0$. Need to mention that, all the gates within the tuning range of the certain adaptive circuit will follow the same body bias configuration. Therefore, in a FBB-only adaptive circuit, the tuning range is usually applied to cover to the critical path.

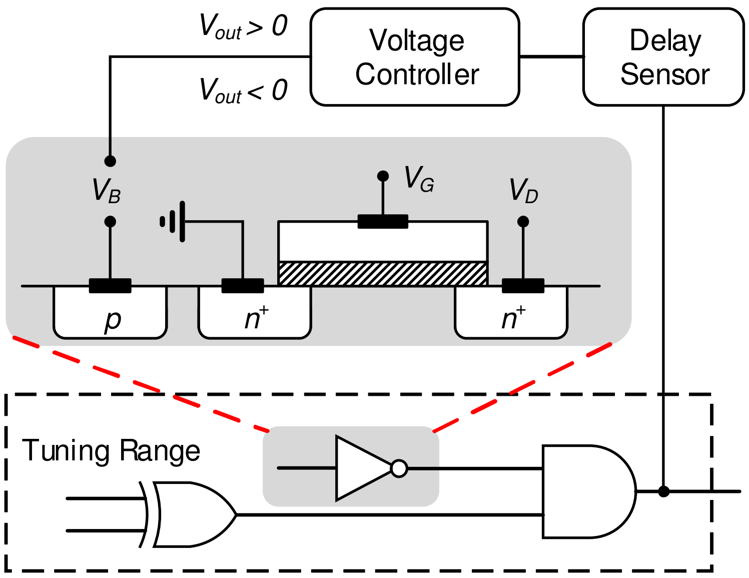


Figure 2.2: Adaptive circuit with the control of body biasing.

2.2.3 Adaptivity Assignment Scheme

Regarded as the redundancy to the original circuit design, the adaptive circuit makes the effectiveness as well as overhead highly dependent on its granularity. Besides, the existing of adaptive circuit also affects the gate sizing and threshold voltage assignment within its control range. Therefore, the adaptivity assignment scheme is highly related with the conventional gate implementation selection.

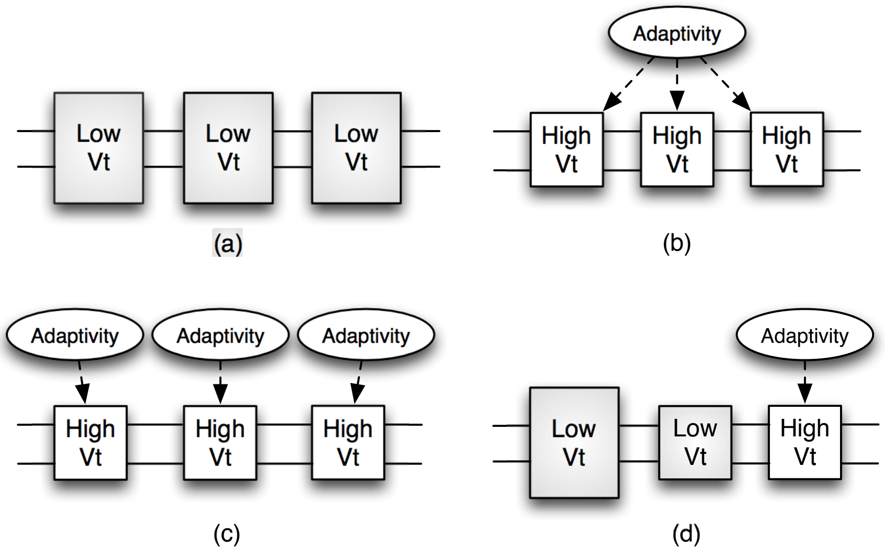


Figure 2.3: Different adaptivity assignment schemes. (a) Over-design in gate implementation selection (gate sizing and V_{th} assignment); (b) Under-design and coarse-grained adaptivity; (c) Under design and fine-grained adaptivity; (d) Collaborative gate implementation selection and fine-grained adaptivity. Reprinted from [1].

Without considering prospective adaptivity, gate implementation selection tends to result in over-design like in Figure 2.3 (a), which entails large power dissipation. An arbitrary anticipation of adaptivity may lead to under design in gate implementation selection. If the under design is fixed by coarse-grained adaptivity like in Figure 2.3 (b), the power efficiency is still poor as the adaptivity power is applied according to the worst place in a circuit. A fine-grained adaptivity like in Figure 2.3 (c) has a large area overhead for compensating the under-design. In our proposed collaborative optimization, gate implementation and adaptivity help each other and thus may provide solution with both high power-efficiency and low adaptivity overhead like in Figure 2.3 (d).

2.2.4 Lagrangian Relaxation

Lagrangian Relaxation is an efficient mathematical optimization framework used to solve the optimization problems with tough constraints. For example, the optimization problem could be formulated as follow

$$\begin{aligned} \text{Minimize} \quad & f(\vec{x}) \\ \text{s.t.} \quad & g_1(\vec{x}) \leqslant 0,\ g_2(\vec{x}) \leqslant 0 \\ & \vec{x} \in X \end{aligned} \tag{2.3}$$

where $f(\cdot)$ is the objective function related to decision variables x, which belong to the finite solution space X. $g_1(\cdot)$ and $g_2(\cdot)$ are the "hard to solve" and "easy to solve" inequality constraints, respectively. By "hard to solve", we actually mean the constraints similar to (2.7) or (2.8) which are hardly to be solved in a linear time or denoted by a straightforward analytical form.

The Lagrangian relaxation is achieved by multiplying the constraint violation with weighting factors (Lagrangian multipliers), and then moving the weighted terms into the objective function as penalties. For example, after the relaxation of $g_1(\vec{x})$, the optimization problem in (2.3) is converted as follow

$$\begin{aligned} \text{Minimize} \quad & \mathcal{L}(\vec{\lambda}, \vec{x}) := f(\vec{x}) + \vec{\lambda} \cdot g_1(\vec{x}) \\ \text{s.t.} \quad & g_2(\vec{x}) \leqslant 0, \ \ \vec{x} \in X \end{aligned} \tag{2.4}$$

where $\mathcal{L}(\vec{\lambda}, \vec{x})$ is named the Lagrangian function, and the vector of Lagrangian multipliers $\vec{\lambda}$ should be non-negative and have the same dimension as $g_1(\vec{x})$. The relaxed problem in (2.4), or namely Lagrangian subproblem, derives the fancy property of concavity since it is the lower envelope of a finite linear functions of $\vec{\lambda}$. Thus, it could be handled by the well-studied convex optimization algorithms.

Moreover, if $\lambda \geqslant 0$ and $g_1(\vec{x}) \leqslant 0$, then it's apparent that the objective function $\mathcal{L}(\vec{\lambda}, \vec{x}) = f(\vec{x}) + \vec{\lambda} \cdot g_1(\vec{x})$ in the relaxed problem (2.4) is less or equal to $f(\vec{x})$ in the original problem (2.3). To minimize the gap, the dual problem is defined in (2.5). It could be further proved that if $\vec{\lambda}^*$ is the optimal solution of the dual problem, then the optimal solution of $\mathcal{L}(\vec{\lambda^*}, \vec{x})$ will also optimize the original problem.

$$\begin{aligned} \text{Maximize} \quad & \mathcal{L}(\vec{\lambda}) := min\mathcal{L}(\vec{\lambda}, \vec{x}) \\ \text{s.t.} \quad & \vec{\lambda} \geqslant 0 \end{aligned} \tag{2.5}$$

2.2.5 Application in Gate Implementation Selection

To formulate the problem of gate sizing and threshold voltage assignment by applying Lagrangian relaxation, the circuit is first described as a directed acyclic graph (DAG) $G(V, E)$, where the vertexes set V includes all the gates, and each edge (v_i, v_j) of set E represents the wire connection between gate v_i and v_j [23]. Besides, arrival times a at the primary input gates $I(G)$ and required arrival times q at the primary output gates of G are also given. Then, the gate size w_i and threshold voltage h_i of each gate $v_i \in V$ is determined as

$$\begin{aligned}
\text{Minimize} \quad & \textstyle\sum_{v_i \in V} p(v_i) && && (2.6)\\
\text{s.t.} \quad & q(v_i) \geqslant a(v_i), && \forall v_i \in I(G) && (2.7)\\
& q(v_i) \geqslant q(v_j) + D(v_j, v_i), && \forall (v_j, v_i) \in E && (2.8)\\
& w_i \in W_i, h_i \in H_i && \forall v_i \in V && (2.9)
\end{aligned}$$

where $p(v_i)$ is the summation of dynamic and leakage power on gate v_i, and $D(v_j, v_i)$ shows the delay from v_j to v_i. W and H are the discrete sets of gate size and threshold voltage provided by the process technology library, respectively.

To solve this problem, the timing constraints are relaxed by multiplying the penalty of Lagrangian multiplier $\vec{\lambda}$ and then integrated into the cost function [24]. The Lagrangian function is given as

$$\begin{aligned}
\mathcal{L}(\vec{\lambda}, \vec{w}, \vec{h}, \vec{a}, \vec{q}) = & \textstyle\sum_{v_i \in V} p(v_i) + \sum_{v_i \in I(G)} \lambda_{i0}(a_i - q_i)\\
& + \textstyle\sum_{(v_j, v_i) \in E} \lambda_{ji}(q_j + D(v_j, v_i) - q_i) \qquad (2.10)
\end{aligned}$$

The Lagrangian function in (2.10) is further simplified by the algebraic transformation in [23], and thus becomes the LR subproblem in (2.11) given the multiplier $\vec{\lambda}$. The Lagrangian dual problem in which the $\vec{\lambda}$ is determined to maximize the Function (2.5) could be solved by the subgradient method and is elaborated in Section (2.5).

$$\begin{aligned}
\text{Minimize} \quad & \mathcal{L}(\vec{\lambda}, \vec{w}, \vec{h}) = \textstyle\sum_{v_i \in V} p(v_i) + \sum_{(v_j, v_i) \in E} \lambda_{ji} D(v_j, v_i) \qquad (2.11)\\
\text{s.t.} \quad & w_i \in W_i, h_i \in H_i \;\; \forall v_i \in V
\end{aligned}$$

2.2.6 Process Variation Modeling

In order to deal with the performance variability during the circuit design, the sources of the variations are analyzed and could be mainly classified into two types. The first one is the inter-die variation which differs from die to die and keeps the identical variation effect to all the gates' parameters on a single die. e.g., the timing analysis on the data path of the design manufactured through the same mask. The second type is intra-die variation which affects the gates' parameters on the same die differently, e.g., the metal width of two inverters locating in different grids.

As for inter-die variation, it is relatively easy to handle with in a variation-aware circuit design algorithm, due to its uniform feature across the entire chip. By contrast, the spatial correlations need to be considered in the intra-die variation. For example, the the delay of a single gate is modeled as [25]

$$d = d_0 + \sum_i \frac{\partial d}{\partial p_i}(p_i - \mu_{p_i}) + N(0, \Sigma) \tag{2.12}$$

where d_0 is the nominal delay, and i is the index of the grid in the circuit. The p_i denotes a location-dependent random component whose mean value and gradient are equal to μ_{p_i} and $\frac{\partial d}{\partial p_i}$, respectively.

Especially, the reason to use grid index i to represent the location-dependent random component is that we assume all gates locate within the same grid will have identical variation distributions. Therefore, it can conclude that there is only one random component in one grid for each random component, or say the grid index i is one-to-one mapping to the random component. As for $N(0, \Sigma)$, it represents the normal distribution of the random components, and Σ is the covariance matrix. For example in Figure 2.4, a chip die with $3row \times 3col = 9$ grids, will have a 9×9 covariance matrix Σ.

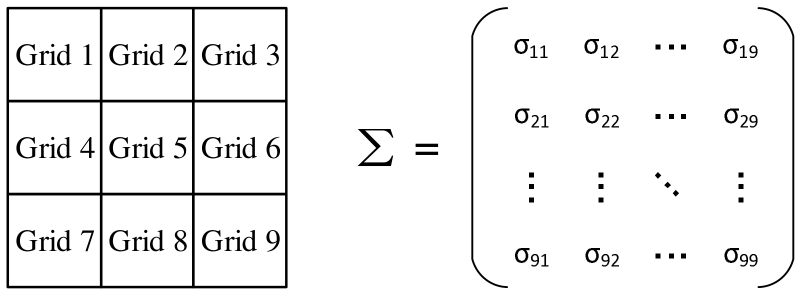

Grid 1
Grid 2
Grid 3
Grid 4
Grid 5
Grid 6
Grid 7
Grid 8
Grid 9
$\Sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{19} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{29} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{91} & \sigma_{92} & \cdots & \sigma_{99} \end{pmatrix}$

Figure 2.4: Chip die with 9 grids and its corresponding 9×9 covariance matrix.

2.3 Problem Formulation

The input to our algorithm is a combinational logic circuit $\mathcal{C}$, which has been clustered into a set of blocks $\mathcal{B} = \{B_1, B_2, ...\}$ [18], timing constraints and adaptivity policy. An adaptivity policy is to control circuit tuning according to results from variation sensors. All gates in the same block follow the same adaptivity configuration. We assume that adaptivities of blocks are independent of each other. If a block is assigned with adaptivity, its tuning is based on its own sensors. This is often true in practice as people tend to avoid high complexity unless it is very necessary. Our algorithm decides whether or not to assign adaptivity to each block. Its objective and constraints include power, timing, robustness to variations and adaptivity area overhead.

2.3.1 Overview of Algorithm Flow

In Figure 2.5, the overall algorithm iterates between gate implementation selection and adaptivity assignment. The gate implementation selection part is handled by Lagrangian relaxation. Its formulation is to minimize power dissipation subject to timing constraints with consideration of variations. Area is not explicitly in the formulation as power and area are correlated in gate sizing. A main reason to use LR here is the runtime cost of timing analysis. For adaptive circuit optimization, variations must be considered and statistical static timing analysis (SSTA) is computationally expensive. By solving the problem in two layers of Lagrangian subproblem and dual problem, the calls to SSTA can be restricted to the dual problem part. Then, the subproblem can be solved using simple models while the overall solution quality is not compromised due to the SSTA guidance in solving the dual problem. The problem size of adaptivity assignment is significantly smaller and allows SSTA to be called more frequently. Therefore, the adaptivity assignment is solved by a sensitivity-based heuristic. In the adaptivity assignment, adaptivity area overhead is explicitly treated as a constraint. The overall flow of our algorithm is shown in Figure 2.5. The outer iteration between gate implementation selection and adaptivity assignment is conducted for only a few times. More iterations are performed within the gate implementation selection part.

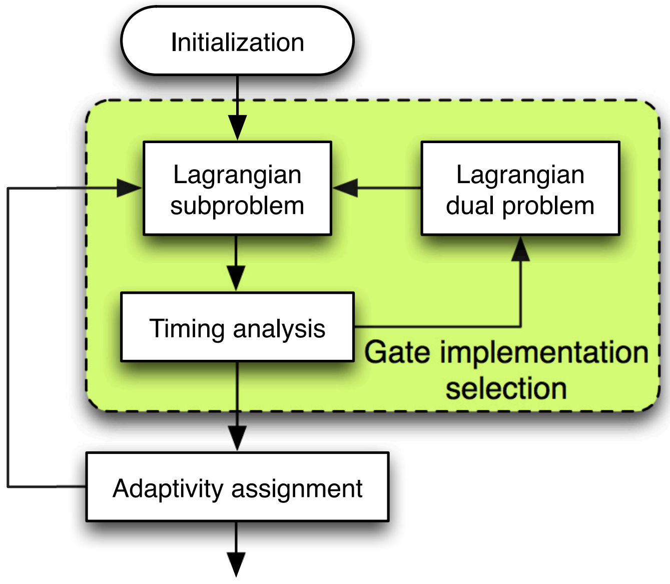


Figure 2.5: Overview of the adaptivity-aware gate implementation selection algorithm flow. Reprinted from [1].

2.3.2 Variation-aware Gate Implementation

Gate implementation selection is to select size and threshold voltage for each gate in a given circuit according to a cell library. Compared to previous works, our method must be adaptivity-aware. That is, if a block is assigned with adaptivity, our implementation selection must be performed with anticipation of performance-power changes due to the adaptivity. Of course, our selection algorithm must take variations into account as well. We make such sophisticated enhancement over a previous work of deterministic gate implementation selection [24]. Moreover, we propose a new technique to avoid redundant counting when candidate solutions are propagated in circuit traversals.

The input includes a combinational logic circuit represented by a directed acyclic graph (DAG) $G(V, E)$, where V is the set of gates and E is a set of edge connections. For each gate $v_i \in V$, we need to select an implementation $\xi(v_i)$, including size and threshold voltage, from a given cell library. The circuit is partitioned into a set of blocks $\mathcal{B} = \{B_1, B_2, ...\}$. There is a binary parameter $\Phi_i \in \{0, 1\}$ to tell if block B_i is assigned adaptivity. If a block is assigned adaptivity, all gates in the block are tuned uniformly according to a given adaptivity policy. We use $\varphi(B_i) \in \{\phi_0, \phi_1, ..., \phi_{max}\}$ to denote adaptive tuning effort level. For example, ϕ_0 means zero body bias and ϕ_{max} indicates the maximum forward body bias. Then, the delay of a gate v_i depends on both its implementation and adaptivity, and is represented by $d_{v_i}(\xi_i, \varphi_i)$. Of course, a gate delay is also affected by its input slew and load capacitance. For the sake of brevity, we omit them in the notation. Similarly, the power dissipation, including dynamic and static power, of a gate v_i is denoted by $w_{v_i}(\xi_i, \varphi_i)$.

When variations are considered, delay $d_{v_i}(\xi_i, \varphi_i)$ becomes a random variable. We employ statistical static timing analysis (SSTA) [25] to capture the variability-aware timing behavior with consideration of spatial correlations. If a block is assigned adaptivity, according to SSTA and the adaptivity policy, we can estimate the probability that a block is at certain tuning level, i.e., $P(\varphi(B_i) = \phi_j)$. Then, we can obtain the expected power of a gate as $\bar{w}_{v_i}(\xi_i) = \sum_{j=0}^{max} P_{v_i}(\phi_j) \cdot w_{v_i}(\xi_i, \phi_j)$. For timing, we only evaluate the case where the maximal tuning effort level is applied if a block has adaptivity. There are two reasons. First, we focus on design-time optimization and only search for a solution that can conform to timing constraints at runtime. If an adaptivity is assigned, an adaptivity policy can always apply the maximal level to satisfy timing constraints based on our solutions. Second, considering probability of tuning configurations on top of probability of variations causes very high estimation complexity and risk of inaccuracy. Our algorithm accommodates general delay, power and variation models, although the Elmore delay model is used here and random variables are assumed to follow Gaussian distributions.

Here is the problem formulation for the gate implementation selection:

$$\text{Min} \quad \sum_{v_i \in V} \bar{w}_{v_i}(\xi_i) \tag{2.13}$$

$$\text{s.t.} \quad a_u + d_{u,v}(\vec{\xi}, \varphi_u(\Phi_u)) \leq a_v, \forall (u,v) \in E \tag{2.14}$$

$$P(a_v \leq Q_v) \geq \Upsilon, \forall v \in PO(V) \tag{2.15}$$

$$\varphi_u(\Phi_u) = \begin{cases} \phi_{max} & if \quad \Phi_u = 1 \\ \phi_0 & if \quad \Phi_u = 0 \end{cases} \tag{2.16}$$

where $P(.)$ indicates probability, a_u is the arrival time at gate u, $\vec{\xi}$ represents the gate implementation selections for all gates, Q_v is the required arrival time, Υ is the constraint for timing yield and $PO(V)$ means primary output gates. If the probability distribution is Gaussian, the probability percentile constraint can be easily represented in terms of mean and standard deviation σ. For example, $\Upsilon = 99.7\%$ requires the mean plus 3σ delay satisfies the required arrival time constraint.

2.3.3 Variation-aware Lagrangian Relaxation

This problem is transformed by Lagrangian relaxation [23] to minimize the following Lagrangian function.

$$\mathcal{L}_{\vec{\lambda}}(\vec{\xi}) = \sum_{v_i \in V} \bar{w}_{v_i}(\xi_i) + \sum_{(u,v) \in E} \lambda_{u,v}(a_u + \tilde{d}_{u,v}(\vec{\xi}, \varphi_u(\Phi_u)) - a_v) \tag{2.17}$$

where $\vec{\lambda}$ is the vector of Lagrangian multipliers and $\tilde{d}$ indicates the mean plus certain σ delay. This is the so-called Lagrangian subproblem. The problem of finding the $\vec{\lambda}$ to maximize the optimal solution to the subproblem is the Lagrangian dual problem. Like in [23, 24], we solve the Lagrangian dual problem using subgradient method guided by SSTA. By doing so, the Lagrangian subproblem is allowed to use simple and less accurate timing and variability models.

By applying the Kuhn-Tucker conditions [23], the subproblem can be further reduced to minimize:

$$\mathcal{L}_{\vec{\mu}}(\vec{\xi}) = \sum_{v_i \in V} \bar{w}_{v_i}(\xi_i) + \sum_{(u,v) \in E} \mu_{u,v}\tilde{d}_{u,v}(\vec{\xi}, \varphi_u(\Phi_u)) \tag{2.18}$$

where $\vec{\mu}$ is a simple function of $\vec{\lambda}$ [23]. We solve the reduced subproblem $\mathcal{L}_{\vec{\mu}}(\vec{\xi})$ by Joint Relaxation and Restriction (JRR) [24]. JRR is a dynamic programming-like solution search. It iteratively propagates candidate solutions in reverse topological order and topological order traversals of G. Such propagation of multiple solutions on a DAG is very challenging as history conflict may happen for solution merging at reconvergence nodes. The work of [24] relaxes the constraint of history consistency in the first reverse topological order search and restores the consistency in subsequent topological order search. In later iterations, additional restrictions are applied so that history conflict no longer occurs.

2.4 Subproblem Solution

For the sake of clarity, only the gate size w is the solution for each gate in the circuit and the threshold voltage h is omitted. To find out the best solutions for all gates, the algorithm first performs a backward search from the primary output to the primary input, and then traces back from the primary input to the primary output.

2.4.1 Backward Search with Inferior Pruning

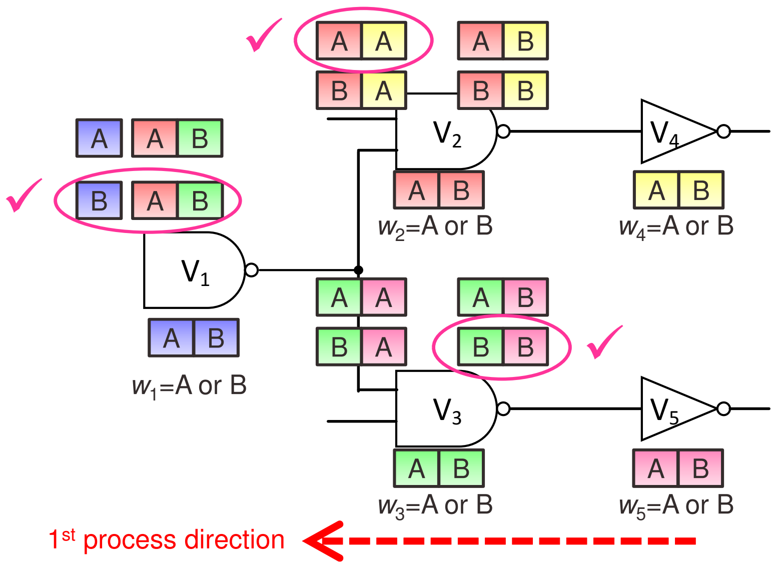


Figure 2.6: Backward topological traverse with inferior pruning.

In Figure 2.6, to find out the w selection for each cell, which has only two options A and B, we starts from the primary output, e.g. v_4. Then, all the solution combinations are evaluated among the fanin cells of v_4 as well as their fanout cones. e.g., v_2 and v_4 are evaluated together, and totally four combinations are generated. After the inferior pruning, one of the four solutions is preserved as ($w_2 = A$, $w_4 = A$). Similar process is used to v_3

and v_5. Thus another four combinations are generated, and still only one solution ($w_3 = B$, $w_5 = B$) is assumed to be kept. Next, when either v_2 or v_3 is under the calculation, v_1, v_2 and v_3 are evaluated together, since v_1 is the common fanin cell for v_2 and v_3. Therefore, the combinations at v_1 are derived, and the one with the minimal cost function and no timing violation is selected as the best solution.

Obviously, without the inferior pruning method, the reverse topological traverse in Figure 2.6 suffers from the exponential solutions at the fanin gate with the increase of circuit hierarchy. Therefore, inferior pruning at the fanin gate is necessary to reduce the process time and storing space. Proved in [24], for a specific size w_i of any gate v_i, at most one non-inferior solution at v_i can be preserved after pruning. This property is justified by the fact that, each solution of a cell v could be represented by the pair of capacitance c and required arrival time q, and then only the solution with the maximum q is not pruned out.

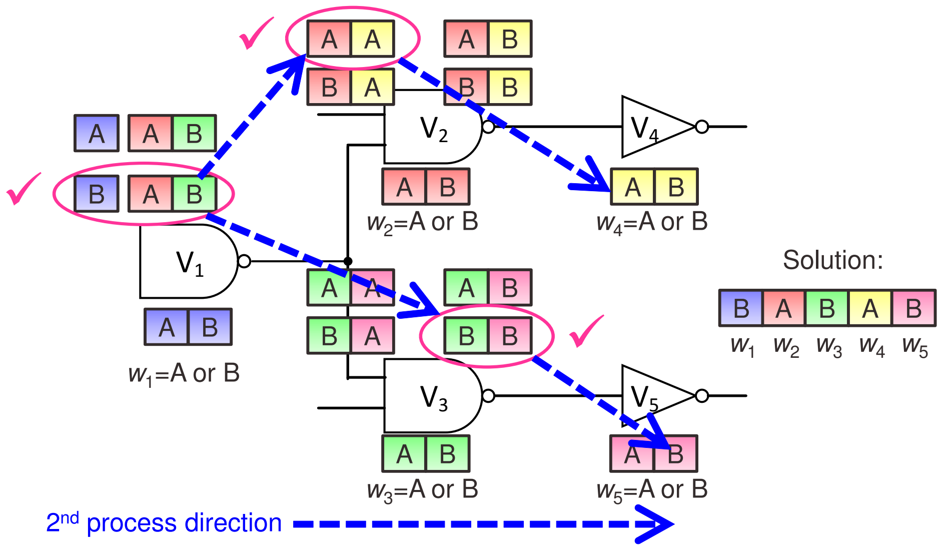


Figure 2.7: Forward topological traverse with solution back trace.

After that, since each solution at v_1 keeps all the historical information, a forward tracking is available in Figure 2.7, and all the solutions for the gates from v_1 till the primary outputs could be determined.

2.4.2 Solution Inconsistence and Forward Restoration

The backward search with inferior pruning works well until it comes to a problem when reconvergence path exists, i.e., the multi-fanin gate. For example in Figure 2.8, suppose after the backward search, the best solution w_1 is get at the primary input v_1, and the forward tracking determines the solutions of v_2 and v_3 as $w_2 = A$ and $w_3 = B$, respectively. However, it fails to tell the w of v_4, since the best solutions for v_2 and v_3 are based on different w of v_4 in the reverse topological traverse.

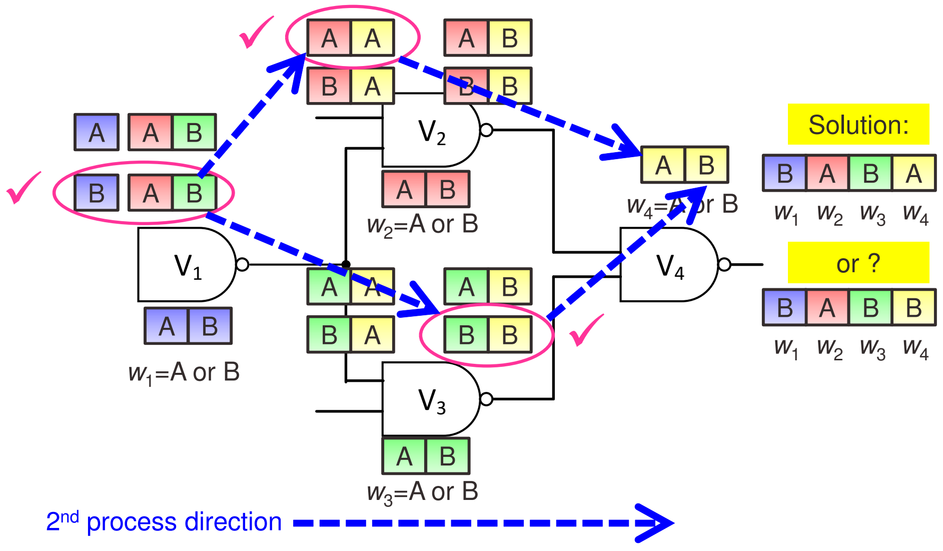


Figure 2.8: Forward topological traverse with solution inconsistence.

A restoration method is applied to solved the solution inconsistence during the forward tracking in Figure 2.9. For the multi-fanin gate v_4, instead of simply applied the solutions inherited from its different fanin cells v_2 and v_3, we derive v_4's solution w_4 by a further calculation based on the best solutions $w_2 = A$ and $w_3 = B$ for fanin gates v_2 and v_3, respectively. By this way, the w solution for v_4 is determined as the one that leads to the minimal cost, and the solution inconsistence is solved.

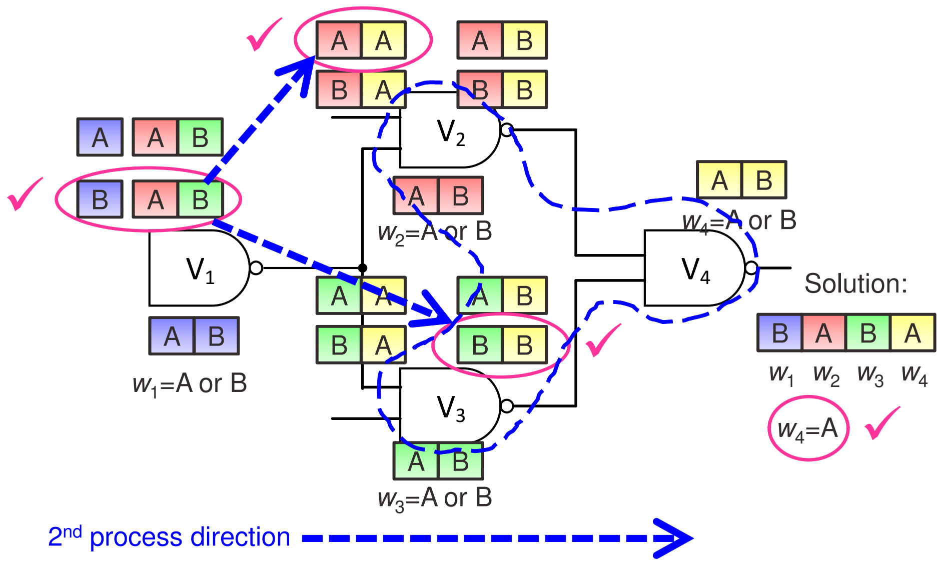


Figure 2.9: Forward topological traverse with solution restoration.

2.4.3 Iterative Refinement

Need to mention that, once the best solution of v_4 is achieved after the forward restoration, the best solutions for the gates v_2 and v_3 in Figure 2.9 may mismatch with those in the backward search, since they are based on different solution of v_4. In Figure 2.10, this disadvantage could be alleviated by performing the 3rd round of backward search during which the solutions are fixed for all the multi-fanin gates. Similarly, the 4th round forward restoration is also needed since the best solutions of v_2 and v_3 are likely changed, and v_4 is affected so on so forth. This iterative refinement [24] is performed until the solutions are converged or the iteration limit is reached.

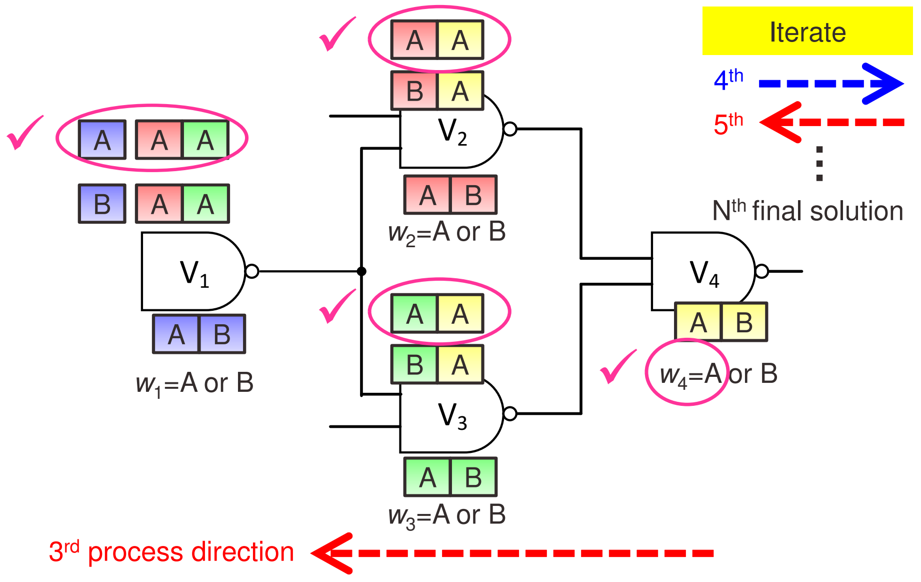


Figure 2.10: Backward topological traverse followed by iterative refinement.

2.4.4 Solution for Redundant Counting

The candidate solution propagations on a DAG faces another problem. That is, a cost may be over-counted repeatedly. For example, let us consider the propagation of cost L on a simple DAG in Figure 2.11 (a) in reverse topological order. The L here corresponds to the $\mathcal{L}_{\vec{\mu}}$ to be minimized in the reduced Lagrangian subproblem. Cost Ld at node d is propagated to both node b and c. When solutions from b and c are merged at node a, Ld is counted twice. In some cases, the count may happen more than twice. This problem is noticed more than two decades ago in works on technology mapping [26]. It was solved by splitting the cost at multi-fanin nodes like Figure 2.11 (b). This cost splitting can avoid the redundant counting. However, some cost estimate, like those at node b and c in Figure 2.11 (b), may see significant error and result in solution quality loss.

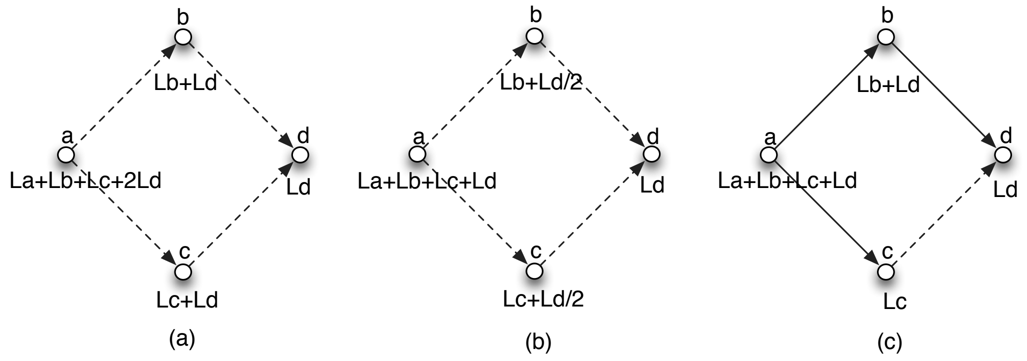


Figure 2.11: Backward topological order solution propagation on DAG. (a) Cost Ld is double-counted at node a. (b) Conventional approach to avoid double counting. (c) Propagation only on spanning tree (in solid edges). Reprinted from [1].

We propose a new technique to solve the over-counting problem with less cost estimate error. This technique is also simple to implement and does not increase the overall algorithm complexity. It consists of the following key steps.

1. *Construction of spanning tree.* Before the solution search, we perform a depth first search (DFS) on G. During the DFS, when the target node of an edge has already been visited, this edge is added into set $\underline{E}$. After the DFS, by removing $\underline{E}$ from G, we obtain a normal spanning tree T, whose edges are solid in Figure 2.11 (c).
2. *Cost propagated on T.* Cost L_T is propagated only along edges in T so that no double counting can happen. This is illustrated in Figure 2.11 (c).
3. *Cost for pruning.* Another cost L_p is maintained for solution pruning at each node. For a single-fanout node, like b and c in Figure 2.11, its L_p is the sum of its local cost and fanout node L_p cost. For example, L_p at node c is $Lc + Ld$. At a multi-

fanout node, its L_p is the sum of its local cost and its fanout L_T cost. For example, the L_p at node a is $La + Lb + Lc + Ld$.

The purpose of L_T is to avoid the double counting while L_p is to reduce under-estimate due to the removal of $\underline{E}$. For example, Ld needs to be counted for the pruning at node c even though edge (c, d) is not in T.

2.5 Dual Problem Solution

As formulated in Equation (2.11), $\mathcal{L}(\vec{\lambda}) := min\mathcal{L}(\vec{\lambda}, \vec{w}, \vec{h})$ features the concavity over $\vec{\lambda} \geqslant 0$. Therefore, in order to maximize $\mathcal{L}(\vec{\lambda})$, the steepest descent algorithm is reasonable to come up with. However, considering that the $\mathcal{L}(\vec{\lambda})$ is not differentiable in the general case, we apply the subgradient based algorithm, which updates the $\vec{\lambda}$ iteratively following the direction of subgradient.

The pseudo code is given by Algorithm 1, which could be divided into two parts, the update of Lagrangian multipliers and the satisfaction to KKT condition. In Steps 1-16, all the edges are evaluated, and each edge $e(u, v)$ denoted by the source gate u and the target gate v is corresponding to one Lagrangian multiplier $\lambda_e(u, v)$. The update function of $\lambda_e(u, v)$ is given by

$$\lambda_{e(u,v)} := \begin{cases} \lambda_{e(u,v)} + \rho(a_u - Q_u), & \text{if } \ v \in PO(V) \\ \lambda_{e(u,v)} + \rho(a_u + D_v - a_v), & \text{if } \ v \notin PO(V) \ \ \&\& \ \ u \notin PI(V) \\ \lambda_{e(u,v)} + \rho(D_v - a_v), & \text{if } \ u \in PI(V) \end{cases} \tag{2.19}$$

according to [23], where $PO(V)$ and $PI(V)$ represent the primary output gates and primary input gates, respectively. ρ is the step size. The nonnegativity of Lagrangian multipliers $\lambda_e(u, v)$ is guaranteed by Steps 13-15. Steps 17-28 are used to adjust the $\lambda_e(u, v)$ in order to satisfy the KKT condition [23] that $\sum_{e_o \in output(u)} \lambda_{e_o} = \sum_{e_i \in input(u)} \lambda_{e_i}, \forall u \in V$.

Input : Combinational circuit $G(V, E)$, cell library L, gate size w and threshold voltage h solution for each $u, v \in V$, and the timing info $\vec{a}$ at each net as well as the $\vec{Q}$ at the primary output of G
Output: $\vec{\lambda}$ for each timing constraint

```
for $each\ e(u,v)\ \in E$ do
    if $u.type = Primary\_Input$ then
        $a_u \leftarrow 0$;
    else
        $a_u \leftarrow u.a$;
    end
    if $v.type = Primary\_Output$ then
        $a_v \leftarrow Q_v;\ d_v \leftarrow 0$;
    else
        $a_v \leftarrow v.a;\ d_v \leftarrow v.R(w_v, h_v) \times \sum_{m \in fanout(v)}(m.C(w_m, h_m))$;
    end
    $\lambda_{e(u,v)} \leftarrow \lambda_{e(u,v)} + step\_size \times (a_u + d_v - a_v)$;
    if $\lambda_{e(u,v)} < 0$ then
        $\lambda_{e(u,v)} \leftarrow 0$;
    end
end
for $each\ u\ \in V$ do
    $\lambda_{out} \leftarrow 0;\ \lambda_{in} \leftarrow 0$;
    for $each\ e_o\ \in out\_edge(u)$ do
        $\lambda_{out} \leftarrow \lambda_{out} + \lambda_{e_o}$;
    end
    for $each\ e_i\ \in in\_edge(u)$ do
        $\lambda_{in} \leftarrow \lambda_{in} + \lambda_{e_i}$;
    end
    for $each\ e_i\ \in in\_edge(u)$ do
        $\lambda_{e_i} \leftarrow \lambda_{e_i} \times (\lambda_{out}/\lambda_{in})$;
    end
end
return $\vec{\lambda}$;
```

Algorithm 1: Subgradient-based approach for updating Lagrangian multipliers.

2.6 Experiment Results

In order to evaluate the effectiveness of our algorithm, we attempt to compare it with other approaches in experiments. To the best of our knowledge, there is no previous work on joint gate implementation selection and adaptivity assignment with consideration of overhead control. Therefore, we compare with the following approaches.

1. Baseline. Variability-aware gate implementation selection without adaptivity. This is to emulate conventional non-adaptive designs.
2. Naïve adaptivity assignment. If only forward body bias (FBB) is considered, adaptivity is assigned to any block that has negative slack in terms of mean plus certain σ value. This is to emulate what designers may do for adaptive circuit design without adaptivity optimization tools. In ABB where both FBB and reverse body bias are allowed, the naïve method simply assigns adaptivity for all blocks. Actually, this is the approach of [18].

In the experiments, gates are modeled by RC switches and the Elmore delay model is employed. We extend a previous SSTA work [25] to perform timing analysis and estimate timing yield as well as variability-aware delay $\tilde{d}_{u,v}(\vec{\xi}, \varphi_u(\Phi_u))$. We consider gate length variations with standard deviation σ being 5% of nominal value, and gate width variations with σ of 2.7% of nominal width. We use adaptive body bias (ABB) [8, 18] as adaptivity. The power model, including dynamic and leakage power, and impact of ABB on delay and power are based on [18]. The adaptivity area overhead includes two parts. Per-gate overhead due to manufacturing process requirement is derived from [18]. Per-block overhead due to sensor and tuning circuits is estimated according to [8]. The experiments are performed on ISCAS85 and ISPD13 [27] benchmark circuits. The largest circuit in the ISPD13 suite has about $150K$ gates. All methods are implemented with C/C++ and the experiments are performed on AMD Opteron processor with 2.2GHz frequency and

Table 2.1: Experimental results of Naïve method with only forward body bias (FBB). Total area overhead and power overhead are denoted by ΔA and ΔW, respectively. Reprinted from [1].

Circuit	#gates	$\|\mathcal{B}\|$	Baseline Yield	Naïve Yield	Naïve ΔA	Naïve ΔW	Naïve CPU (s)
c432	171	4	94.9%	99.3%	707	6564	1
c499	218	5	91.6%	97.7%	1433	10975	1
c880	383	5	96.3%	98.9%	809	5123	1
c1355	562	4	88.8%	99.9%	1587	26442	2
c1908	972	6	75.9%	99.9%	1380	19049	4
c2670	1287	5	94.6%	98.2%	947	6156	5
c3540	1705	5	73.6%	99.9%	1759	21952	8
c5315	2351	6	90.9%	99.8%	2602	29364	10
c6288	2416	6	93.9%	99.9%	1931	50323	11
c7552	3625	5	41.8%	99.9%	3291	42878	18
fft	32281	20	81.2%	99.1%	15742	194576	310
cordic	41601	20	73.9%	99.5%	22618	443511	493
des_perf	112644	22	83.5%	99.2%	43608	204159	750
matrix_mult	155325	20	44.0%	99.1%	78028	1382050	1378
Average			80.4%	99.3%	12603	174509	214

Linux operating system. The gate implementation selection for each method is performed with 14 iterations, i.e., Lagrangian multipliers are updated 14 times. The best solution in terms of problem formulation is selected. In our collaborative optimization approach, the iteration between gate implementation selection and adaptivity assignment is conducted twice.

The first two experiments are to evaluate the effectiveness of our approach for forward body bias (FBB)-only and ABB, which allows both forward and reverse body bias (RBB). Relatively tight timing constraints are applied to FBB cases as FBB is mostly for timing improvement. The ABB cases have relatively loose timing constraints to see the effect of RBB on leakage power reduction. The results are shown in Table 2.1 and 2.3. In both tables, the number of gates and blocks of each circuit are displayed in the second and

Table 2.2: Experimental results of our method with only forward body bias (FBB). Gate area, total area and power overhead are denoted by ΔA_g, ΔA, and ΔW, respectively. Reprinted from [1].

Circuit	#gates	$\|\mathcal{B}\|$	Baseline Yield	Ours Yield	Ours ΔA_g	Ours ΔA	Ours ΔW	Ours CPU (s)
c432	171	4	94.9%	99.9%	7%	323	2524	1
c499	218	5	91.6%	99.9%	-26%	355	3688	3
c880	383	5	96.3%	99.3%	14%	504	1790	3
c1355	562	4	88.8%	99.4%	-17%	388	12922	5
c1908	972	6	75.9%	99.8%	-5%	762	9162	9
c2670	1287	5	94.6%	99.1%	-7%	176	544	12
c3540	1705	5	73.6%	99.6%	-13%	603	13924	16
c5315	2351	6	90.9%	99.2%	0%	293	3350	24
c6288	2416	6	93.9%	98.9%	4%	1248	10549	24
c7552	3625	5	41.8%	99.9%	-16%	404	20053	40
fft	32281	20	81.2%	99.2%	0%	10376	32167	759
cordic	41601	20	73.9%	99.1%	0%	9590	146446	1141
des_perf	112644	22	83.5%	99.4%	0%	15060	5726	1795
matrix_mult	155325	20	44.0%	99.0%	-15%	-47184	-200650	3193
Average			80.4%	99.4%	-5.23%	-558	4443	502
% difference vs. naïve = (ours-naïve)/ |naïve|						-104.4%	-97.5%	

third column. The fourth column is for timing yield of the baseline, where no adaptivity is applied. Columns 5-8 provide results from the naïve method and the rightmost 5 columns are the results from our method. For each method, we examine the power overhead ΔW and total area overhead ΔA in addition to timing yield and CPU runtime. For our method, we also show the gate area overhead ΔA_g. All overheads are with respect to the baseline results. Second to the last row in each table provides the average results and the last row tells the percentage difference of our method versus the naïve approach. In both tables, the top 10 cases are from ISCAS85 benchmark and the last 4 are ISPD13 cases.

For the cases of FBB, our methods can reduce power and area overhead by around 100% compared with the naïve approach. Due to the collaboration between gate implementation selection and adaptivity assignment, our method often reduces gate area from

Table 2.3: Experimental results of Naïve method with forward body bias and reverse body bias (ABB). Total area and power overhead are denoted by ΔA and ΔW, respectively. Reprinted from [1].

Circuit	#gates	$\lvert\mathcal{B}\rvert$	Baseline Yield	Naïve Yield	Naïve ΔA	Naïve ΔW	Naïve CPU (s)
c432	171	4	99.7%	99.5%	689	1857	1
c499	218	5	99.9%	99.9%	1358	664	1
c880	383	5	99.9%	99.6%	921	1452	1
c1355	562	4	99.8%	99.1%	1354	271	2
c1908	972	6	99.5%	99.2%	1550	2623	4
c2670	1287	5	99.9%	99.9%	1543	-435	5
c3540	1705	5	99.9%	99.7%	1821	-1481	7
c5315	2351	6	99.9%	99.9%	2668	-3949	10
c6288	2416	6	99.9%	99.9%	2175	-8302	11
c7552	3625	5	99.9%	99.9%	3103	-3307	17
fft	32281	20	99.8%	99.7%	41061	-137438	297
cordic	41601	20	99.5%	99.3%	43106	-160335	488
des_perf	112644	22	99.4%	99.4%	67013	-204797	734
matrix_mult	155325	20	99.0%	99.3%	90859	-287264	1336
Average			99.7%	99.6%	18516	-57174	209

the baseline. Of course, both methods can largely fix the timing problem from the baseline. In the ABB cases, our method causes 85% less area overhead than the naïve method. It has 42% less power savings than the naïve method, but the power savings compared to the baseline is still significant.

The third experiment is to investigate the power/area versus timing tradeoff of our approach. The result from ISCAS85 circuit C7552 is plotted in Figure 2.12. We relax the required arrival time on the primary outputs of C7552, and then observe the area and power resulted from our algorithm. The result shows that both the area and power increase with increasingly tight timing constraint as expected.

In the last experiment, we investigate the impact of adaptivity granularity. That is, we vary the number of adaptivity blocks $\lvert\mathcal{B}\rvert$ of ISPD13 circuit *fft* and examine the effect on

Table 2.4: Experimental results of our method with forward body bias and reverse body bias (ABB). Gate area, total area and power overhead are denoted by ΔA_g, ΔA and ΔW, respectively. Reprinted from [1].

Circuit	#gates	$\|\mathcal{B}\|$	Baseline Yield	Ours Yield	Ours ΔA_g	Ours ΔA	Ours ΔW	Ours CPU (s)
c432	171	4	99.7%	99.3%	0%	0	0	2
c499	218	5	99.9%	99.1%	0%	0	0	3
c880	383	5	99.9%	99.0%	-3%	-43	-71	3
c1355	562	4	99.8%	99.1%	0%	0	0	5
c1908	972	6	99.5%	99.1%	0%	0	0	9
c2670	1287	5	99.9%	99.7%	0%	458	-562	12
c3540	1705	5	99.9%	99.5%	-2%	522	-1378	16
c5315	2351	6	99.9%	99.8%	0%	0	0	24
c6288	2416	6	99.9%	99.9%	-29%	-1595	-7791	25
c7552	3625	5	99.9%	99.9%	-9%	925	-6316	40
fft	32281	20	99.8%	99.5%	0%	22368	-91955	824
cordic	41601	20	99.5%	99.3%	-7%	-1050	-63189	1121
des_perf	112644	22	99.4%	99.3%	0%	19979	-79359	1805
matrix_mult	155325	20	99.0%	99.3%	-9%	-2734	-213180	3203
Average			99.7%	99.4%	-4.2%	2774	-33129	507
% difference vs. naïve = (ours-naïve)/ |naïve|						-85.0%	42.1%	

power and area overhead. The results from our method are plotted in Figure 2.13. A small (large) number of blocks means coarse-grained (fine-grained) granularity. One can see that the power/area overhead is high when the granularity is too coarse or too fine. When the granularity is too coarse, each block is relatively large and must involve nodes of different timing behaviors. The adaptive tuning in this case must be targeted toward the worst case gates and unnecessary power and area overhead are paid on non-critical gates. When the adaptivity is too fine-grained, the per-block overhead due to sensors, control and tuning circuits becomes very large. Therefore, there is sweet spot for adaptivity granularity.

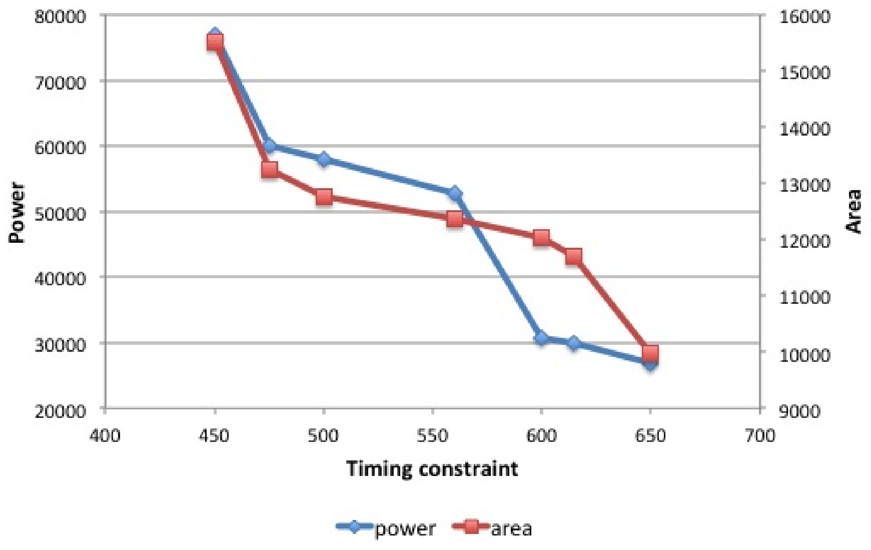


Figure 2.12: Power/area-timing tradeoff for circuit c7552. Reprinted from [1].

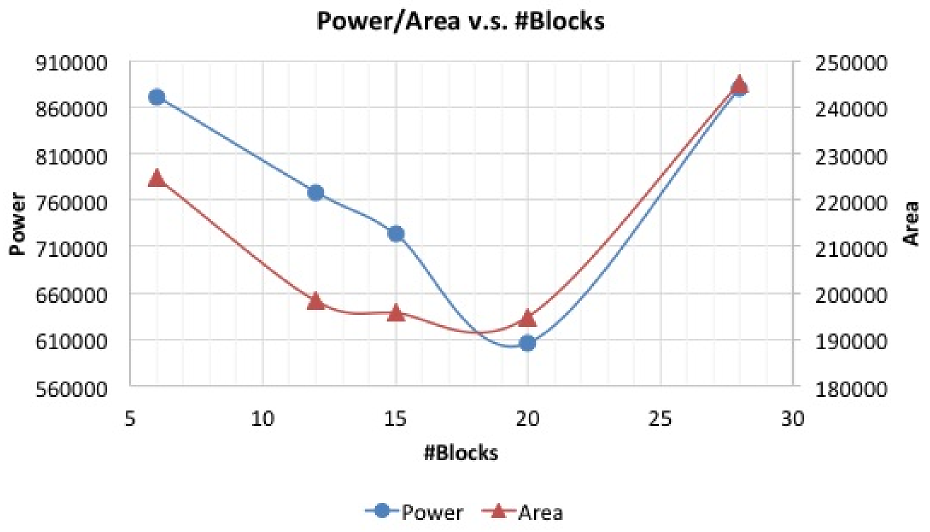


Figure 2.13: Power/area vs. granularity for circuit fft. Reprinted from [1].

3. BUILT-IN SELF OPTIMIZATION FOR VARIATION RESILIENCE OF ANALOG FILTERS*

3.1 Introduction

Nowadays, analog integrated circuit (IC) design continues to be a great challenge. Different from the streamlined digital design using standard cells and automatic tools, analog IC design still relies on hand calculation, SPICE simulation, and designers' personal experiences. In order to improve the efficiency of analog design, automatic optimization methods have been explored. For example, [28] studied analog circuit sizing, which can changes the dimension of transistors, capacitors or resistors. However, as the IC process technology scales down into deep submicron regime, manufacturing process variations become pronounced and often result in remarkable performance deviation from specifications. Moreover, device aging [29], such as bias temperature instability (BTI) and hot carrier injection (HCI), causes additional characteristic changes over time. Circuit optimization techniques have attempted to address these issues [30, 31]. However, design-time optimization implies a uniform solution, which is difficult to achieve for all different variation cases.

To address the individual process variations, [32] studied post-silicon tuning techniques where circuits are designed with a certain configuration, which is performed at manufacturing testing. Each chip instance is tested and tuned by test equipment to compensate for its own variations. As testing is performed only once before chips are integrated into their systems, such tuning is not adequate for tackling aging effects that change over time. An alternative approach is self-tuning, which can be performed at any time in product life. One such approach [33] is self-tuning in communication circuits where the

*Reprinted from "Built-In Self Optimization for Variation Resilience of Analog Filters", by Jiafan Wang, Congyin Shi, Edgar Sánchez-Sinencio, Jiang Hu, ©2015 IEEE.

configuration search is undertaken by the baseband digital signal processor (DSP). Evidently, the dependence on a DSP restricts applications of this technique. A built-in self tuning technique for A/D converter design is described in [34]. The tuning objective function is a parasitic mismatch, which is relatively simple compared to the overall analog system performance, such as gain, linearity, phase margin, etc. A recent work [35] attempts to achieve a built-in self tuning of general analog circuit performance. The tuning is controlled by a neural network circuit, whose training needs external assistance.

In this work, we explore a general framework of build-in self optimization for variation resilience of analog ICs. By “optimization” instead of “tuning”, our framework contains a digital circuit that implements an optimization algorithm rather than simple if-then rules. This is a powerful approach that can handle cases where both performance function and variation-performance relation are complicated. The study platform here is band-pass filter, which is a common analog module existing in many analog IC designs. The filter is designed with configurable parasitics, such as capacitors and resistors. We define a cost function that can capture the mismatch of its frequency response from the specification. The configuration search is realized by a simulated annealing (SA) based approach. Both the cost function and the SA-based configuration search are implemented by digital circuits. As transistor feature size keeps shrinking, more silicon estates become available to support these digital self optimization circuits. Even so, we still strive hard to minimize circuit overhead. The effectiveness of conventional SA is hampered by the limited area budget. To this end, we devise algorithmic techniques that significantly improve search quality. Since the digital optimizer circuit is invoked occasionally, it can be power gated to minimize its power overhead. The main advantages of our approach includes the following:

1. It does not require the test equipment, host processor or training, and is therefore a truly autonomous approach.

2. It captures compounding variation effects without relying on any models but very limited historical information.
3. It is flexible when used in different kinds of analog circuits with only change of the cost function.

3.2 Background

3.2.1 System Architecture

An overview of the proposed built-in self optimization architecture is depicted in Figure 3.1. It consists of an on-chip analog test bench and a digital optimization circuit. In the analog part, a stimulus generator produces a clock signal and a sinusoidal waveform, both of which have the same frequency. The sinusoidal waveform is fed into the circuit-under-test (CUT) and stimulates the CUT to generate an output. A multiplier, which adopts the self-mixing technique in [36], and an analog-to-digital converter (ADC) are employed for measuring the output response. In addition, the measurement path can be selected by an input/output selector. Moreover, coherent detection is used, and an I/Q selector can shift the multiplier's clock by 0 or 90 degrees. I/Q data, which is related to the signal modulation in communication systems, can be retrieved by the ADC, indicating the phase and amplitude changes of the input and the output waveform. Later, the amplitude of the waveform is calculated by the digital part. Compared to the pure analog approach in [36], the digital approach reduces the complexity of the analog parts and is more scalable for advanced IC technologies.

The analog and digital circuits cooperate in a closed loop to perform the self optimization procedure. Based on the CUT's output responses measured by the ADC, a main part of the digital circuit computes the cost function $f(\vec{x})$, which is to be minimized by the optimization. The value of the cost function depends on a vector of CUT configuration variables, $\vec{x}$, whose values are to be decided by the optimization engine. We developed a

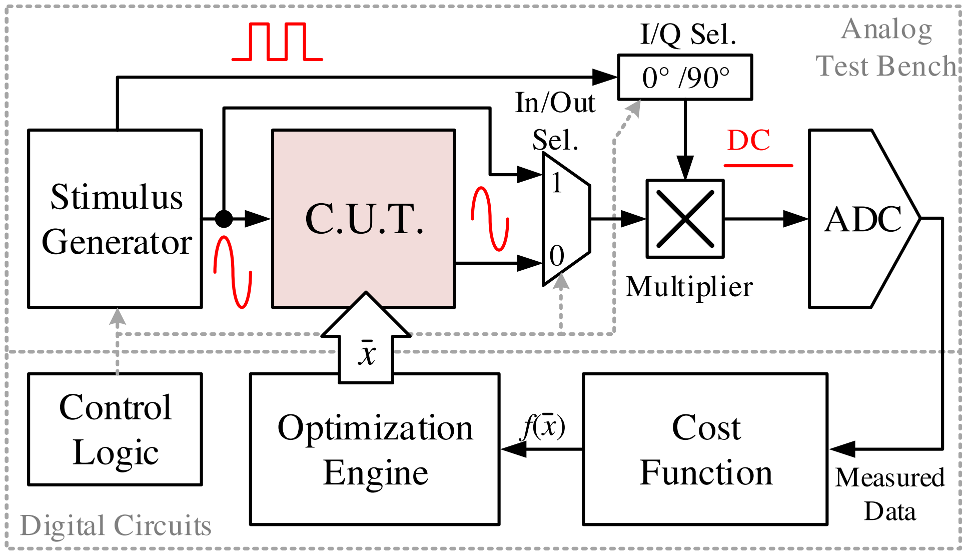


Figure 3.1: Overview of the proposed built-in self optimization system architecture. Reprinted from [2].

digital circuit optimizer based on a hybrid multi-start meta-heuristic. The optimization is an iterative procedure where various configurations of $\vec{x}$ are applied to CUT till the one that minimizes $f(\vec{x})$ is found or the limit to iterations is reached. Apart from the cost function and the core optimization engine, a control logic is needed to coordinate the timing of the entire system. In the proposed approach, the optimization engine is generic for almost any type of analog circuits, while the cost function circuit needs be customized for different analog circuits.

3.2.2 Programmable Bandpass Filter

The second order band-pass filter (BPF) is chosen as the analog circuit platform for our study, because it is a very common module in many analog circuit systems, and its performance description as well as its relation with variations are not straightforward. The schematic of a fully programmable Tow-thomas active-RC band-pass filter (BPF) biquad is illustrated in Figure 3.2. All the resistors and capacitors are digitally controlled arrays.

In this design, a 5-bit resistor bank and a 3-bit ($k = 3$) capacitor bank are implemented. From Figure 3.2, we derive the transfer function of the whole BPF, assuming the amplifiers are ideal.

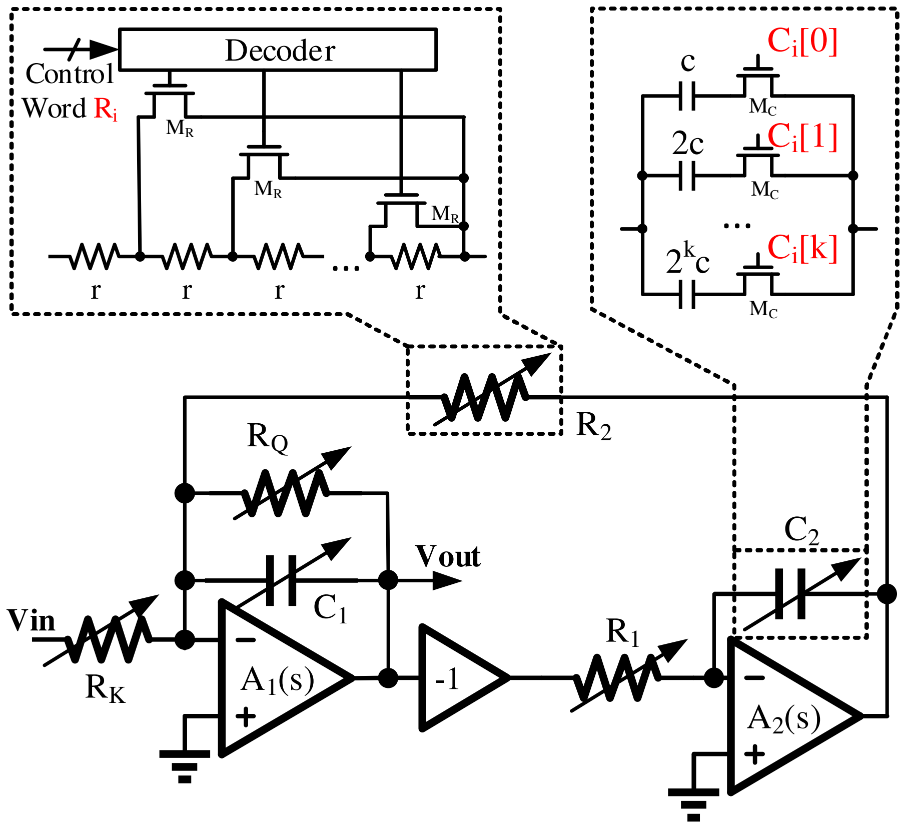


Figure 3.2: Second-order band-pass filter and the configurable resistance/capacitor array structure. Reprinted from [2].

$$\begin{aligned} H(s) &= \frac{V_{out}}{V_{in}} = -\frac{G_{BPF}\frac{\omega_0}{Q}s}{s^2 + \frac{\omega_0}{Q}s + \omega_0^2} \\ &= -\frac{\frac{1}{R_K C_1}s}{s^2 + \frac{1}{R_Q C_1}s + \frac{1}{R_1 R_2 C_1 C_2}} \end{aligned} \tag{3.1}$$

and

$$G_{BPF} = -\frac{R_Q}{R_K}, \quad \omega_0 = \frac{1}{\sqrt{R_1 R_2 C_1 C_2}}, \quad Q = \sqrt{\frac{R_Q^2 C_1}{R_1 R_2 C_2}}$$

where G_{BPF} is the gain, ω_0 is the central angular frequency, and Q is the quality factor of the BPF. Particularly, if we set $R_Q = R_K = R_{x_1}$, $R_1 = R_2 = R_{x_2}$, and $C_1 = C_2 = C$; then, we theoretically have $G_{BPF} = -1$, $\omega_0 = 1/\sqrt{R_{x_1} C}$, and $Q = R_{x_1}/R_{x_2}$. Here, R_{x_1} and R_{x_2} are chosen as the "tuning knobs" of the BPF, which can control the shape of the frequency response. In our design, they're digitally controlled resistor arrays with 5-bit control words; thus, later integer numbers x_1 and x_2 will be used to present the control words instead of the resistance values. Additionally, a fixed C is selected to reduce the number of dimensions for the optimization problem, because x_1 is sufficient to control the BPF central frequency.

3.2.3 Non-ideal Effects

Ideally, the frequency response of a BPF is well defined by (3.1), and deterministically depends on the resistor and capacitor values, which can be properly chosen at design time. But unfortunately, non-idealities in the IC chips will introduce more complexity. Firstly, the amplifiers are not ideal. They have limited gains, limited output impedance and their own frequency responses, which are usually low-pass. The frequency response of the amplifiers are annotated as $A_1(s)$ and $A_2(s)$ in Figure 3.2, which should be inserted into (3.1). But even worse is that $A_1(s)$ and $A_2(s)$ will change due to the aging effects of the MOSFETs in the amplifiers [29]. Moreover, as illustrated in Figure 3.2, MOS switches M_R in the resistor bank and M_C in the capacitor bank introduce ON resistance in series with the resistor and capacitor separately. The parasitic capacitance in these switches also need to be considered. Nevertheless, the most important thing is the process variation.

On the one hand, the transistors are affected by the variations. Simulation reveals 6.4% deviation on the bandwidth of the amplifiers, and 2.7% deviation on the amplifiers' DC gain among chips. However, feedback loop technique is used in the BPF design and, thus, the circuit is robust against variations of MOSFETs. But on the other hand, the standard deviation of the absolute resistance value is about 8% of the model parameter, and it is 3.5% for the capacitance value. These are the main error sources for an on-chip BPF. In other words, deterministic resistor/capacitor values, i.e., fixed x_1 and x_2 values, provide little assurance for the desired frequency response. Therefore, a self optimization mechanism, which finds a proper set of component values and can operate for individual chip instances, is necessary.

The effect of variations, compounded with non-ideal conditions, can be quite complex. It may distort the frequency responses in various ways such as deviating the gain at the central frequency, shifting the central frequency, changing the bandwidth. Moreover, the Q-factor may be affected as well. It is very difficult, if not impossible, to find analytical functions to describe how the performance deviation depends on the variations. This is why we take a model-free multi-start meta-heuristic optimization approach, in this framework.

3.3 Cost Function Design and Implementation

3.3.1 Optimization Cost Function

In general, *optimization* is a procedure to find values for a set of decision variables $\vec{x}$ such that certain objective function $\Phi(\vec{x})$, which is also called a cost function, is minimized subject to certain constraints. The definition of the cost function plays a fundamental role in guiding the optimization solution search. For an analog circuit system, we wish the actual circuit characteristics to match well with the specifications. Hence, a general template of cost function is the mismatch between the actual characteristics and corresponding specifications, and can be described by

$$\begin{aligned} \Phi(\vec{x}) &= \|\beta_1\phi_1(\vec{x}), \beta_2\phi_2(\vec{x}), \ldots, \beta_n\phi_n(\vec{x})\|_2 \\ \phi_i(\vec{x}) &= g_i(\vec{x}) - \alpha_i G_i, \;\; i = 1, 2, \cdots, n \end{aligned} \tag{3.2}$$

where $\phi_i(\vec{x})$ indicates the mismatch between the actual characteristic $g_i(\vec{x})$ and the corresponding specification G_i, and i is the index for the total n measurements. α_i and β_i are constant weighting factors.

For the BPF design described in Section 3.2.2, the decision variables $\vec{x}$ determine the resistor configuration. Frequency response is a main performance metric for the BPF. According to the transfer function (3.1). The highest gain g_{s_2} should be obtained at the central frequency s_2. $3dB$ gain degradation is expected at the two bandwidth frequencies, s_1 and s_4, and we have $s_1 < s_2 = s_3 < s_4$. Thus, the cost function for the BPF is defined to be the mismatch of gains g_{s_1}, g_{s_2}, g_{s_3} and g_{s_4}.

Instead of directly implementing the cost function as in (3.2), we make a couple of changes in order to limit the circuit implementation overhead. First, the L2-norm $\|\cdot\|_2$ is replaced by $|\cdot|$ so as to reduce the circuit complexity. Second, our cost function focuses on the normalized frequency response curve, where only the central frequency location and the gain drops at s_1 and s_4 matter. In fact, we just enforce based on the gain at s_2 being $3dB$ greater than those at s_1 and s_4. The vertical offset of the response curve can be easily handled, and we merely include a penalty term in the cost function to make the measured gain at s_2 to be greater than 1. Therefore, the cost function for the BPF is defined as

$$\begin{aligned}\Phi_{BPF}(\vec{x}) &= \beta_1 \left| \frac{A(s_2)_{out}}{A(s_2)_{in}} - \alpha_1 \frac{A(s_1)_{out}}{A(s_1)_{in}} \right| \\ &+\beta_2 \left| \frac{A(s_3)_{out}}{A(s_3)_{in}} - \alpha_2 \frac{A(s_4)_{out}}{A(s_4)_{in}} \right| \\ &+\beta_3 \left| \frac{A(s_1)_{out}}{A(s_1)_{in}} - \alpha_3 \frac{A(s_4)_{out}}{A(s_4)_{in}} \right| \\ &+\beta_4 \left| \frac{A(s_2)_{out}}{A(s_2)_{in}} - \alpha_4 \frac{A(s_3)_{out}}{A(s_3)_{in}} \right| + P \qquad (3.3)\end{aligned}$$

where $A(s_i)_{out}$ and $A(s_i)_{in}$ are the signal amplitudes at the ith frequency point for the output and the input, respectively. The ratio $A(s_i)_{out}/A(s_i)_{in}$ is the measured gain $g_{s_i}(\vec{x})$ at frequency s_i. The last term, P, in the right-hand side of (3.3) is the penalty function, which deals with any violation to the constraint that the gain at s_2 must be no less than 1. This is an effective technique in the optimization theory to simplify the constrained problem into an unconstrained one. Especially, it is described by

$$P = \begin{cases} 0 & \text{if } \frac{A(s_2)_{out}}{A(s_2)_{in}} \geqslant 1, \\ 1 - \dfrac{A(s_2)_{out}}{A(s_2)_{in}} & \text{otherwise.} \end{cases} \qquad (3.4)$$

The constant coefficients in (3.3) are decided as follows. We set $\alpha_1 = \alpha_2 = \sqrt{2}$ and $\alpha_3 = \alpha_4 = 1$ such that the gains at s_1 and s_4 are the same and both have 3dB drop compared to the gain at s_2. We set $\{\beta_1, \beta_2, \beta_3, \beta_4\} = \{1, 1, 1, 1\}$ for our case of BPF.

3.3.2 Circuit Implementation

To implement the cost function equation in (3.3), Figure 3.3 illustrates the complete circuit diagram with an area-saving manner. Generally, the dashed and light-shaded rectangle, or namely gain module $g(\cdot)$, transforms the measured I/Q data into the amplitude $A(s_i)$ and then calculates the frequency gain g_{s_i} by $A(s_i)_{out}/A(s_i)_{in}$ at frequency point s_i.

Finally, the summation circuit belove $g(\cdot)$ generates $\Phi(\vec{x})$ and forwards it to the optimization engine introduced in Section 3.4.

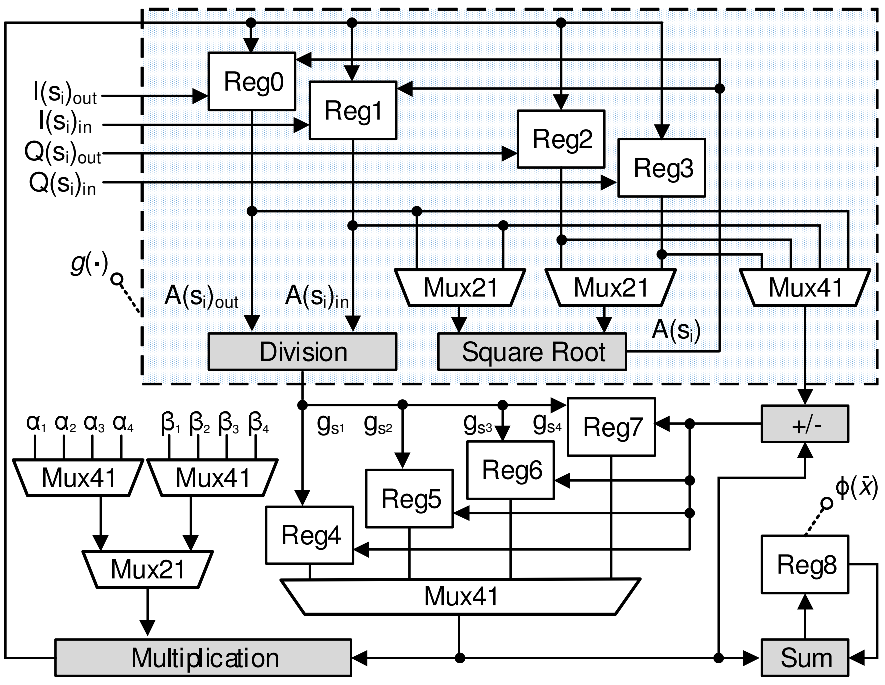


Figure 3.3: Cost function circuit diagram. Each gray rectangle is an arithmetic block, and each white rectangle is a register. "Reg" implies the resistor, while "Mux21" and "Mux41" represent the 2-to-1 and 4-to-1 multiplexers, respectively.

To be specific, during the transformation of $g(\cdot)$ in Figure 3.3, data of $I(s_i)_{out}$ and $Q(s_i)_{out}$ is first stored in "Reg0" and "Reg2", respectively. Then they are delivered to the "Square Root" process. Later, the derived $A(s_i)_{out}$ updates "Reg0" and thus needs no extra register. Similar process will be applied to "Reg1" and "Reg3". After they keep the input data of $I(s_i)_{in}$ and $Q(s_i)_{in}$, respectively, "Reg1" will be updated with $A(s_i)_{in}$ created by the "Square Root" process. Finally, $A(s_i)_{out}$ and $A(s_i)_{in}$ are provided to the "Division" process, and the gain g_{s_i} is obtained.

Notice that, the transformation of $g(\cdot)$ in Figure 3.3 could be regarded as the function of frequency point s_i. Therefore, to get another gain g_{s_j}, we first save $g_{s_{i \neq j}}$ in one of the registers, e.g., Reg4, $\cdots$, Reg7 in Table 3.1, and then reuse the gain module by invoking it upon the new I/Q data measured at frequency point s_j. As such, only a outer logic to control the frequency s_i is added, and this minor change to $g(\cdot)$ simplifies the overall circuit.

Table 3.1: Usage of registers within the summation circuit.

registers	$g(\cdot)$ stage	intermediate stage	summation stage
Reg0		α_1·Reg4	|Reg4|
Reg1		α_2· Reg7	|Reg5|
Reg2		α_3· Reg7	|Reg6|
Reg3		α_4· Reg6	|Reg7|
Reg4	g_{s_1}	$g_{s_2}-$ Reg0	β_1·Reg0
Reg5	g_{s_2}	$g_{s_3}-$ Reg1	β_2·Reg1
Reg6	g_{s_3}	$g_{s_1}-$ Reg2	β_3·Reg2
Reg7	g_{s_4}	$g_{s_2}-$ Reg3	β_4·Reg3
Reg8	0	P	$P + \sum_{i=4}^{7}$Reg$i$

* *Note*: In each stage, registers update from top to bottom.

Moreover, after all the g_{s_i}'s are collected, the rest part of the calculation in (3.3) could be done through the circuit below the $g(\cdot)$ block in Figure 3.3. As in Table 3.1, reusing resisters in the $g(\cdot)$ block, e.g., Reg0, $\cdots$, Reg3, could maintain the intermediate result of $\alpha_i A(s_j)_{out}/A(s_j)_{in}$ and avoid extra registers. Thus, the circuit size is further reduced. Eventually, the cost function result $\Phi(\vec{x})$ is stored in "Reg8".

3.3.3 CORDIC Algorithm Based Calculation

The value of $A(s_i)$ is obtained by computing $\sqrt{I^2 + Q^2}$, where the I/Q data is retrieved from the measurement of analog circuit, which has been discussed in Section 3.2. A direct computing of $\sqrt{I^2 + Q^2}$ with square and square-root units entails a large circuit area. Instead, we adopt the CORDIC (COordinate, Rotation DIgital Computer) algorithm [37], which is an iterative procedure of addition/subtraction and shifting. The pseudo code for computing $\sqrt{I^2 + Q^2}$ is given by Algorithm 2 where N is the number of iterations, and i indicates the current step.

```
Input  : I/Q data stream $I_0$ and $Q_0$
Output: Amplitude of the I/Q modulation
if $|Q_0| > |I_0|$ then
    $temp \leftarrow |I_0|; I_0 \leftarrow |Q_0|; Q_0 \leftarrow temp;$
end
for $i \leftarrow 0$ to $N-1$ do
    $d_i = -sign(I_i Q_i);$
    $I_{i+1} = I_i - 2^{-i} d_i Q_i;$
    $Q_{i+1} = Q_i + 2^{-i} d_i I_i;$
end
return $I_N$;
```

Algorithm 2: CORDIC-based Square Root of Power Sum.

The main idea of Algorithm 2 is to change the point (I_0, Q_0) by a series of rotations along the circumference of a circle, which has the radius of $\sqrt{I_0^2 + Q_0^2}$ and is centered at $(0, 0)$, until its y-coordinate reaches 0. Then the absolute value of its x coordinate is equal to the desired result. The entire rotation process takes the total N iterations. In each iteration, the point (I_i, Q_i) is rotated by an angle of $arctan(2^{-i})$, and the direction of the rotation is determined by the sign of $I_i Q_i$. The rotated angle keeps decreasing and, finally,

I_i converges to $K\sqrt{I_0^2 + Q_0^2}$, where K is a constant and can be omitted since we care only the amplitude ratio in (3.3). It should be noted that the swap of operands in Steps 1 and 2 is critical for an implementation with limited precision. If $I_0 < Q_0$ and the swap are not performed, after several right shifts in Step 7, I_i would reduce to 0, and Q_i would not converge. Empirically, we set $N = 8$ and find it suffices to make the algorithm converge.

3.4 Optimization Engine and Implementation

3.4.1 Hybrid Optimization Algorithm

The proposed optimization engine is a multi-start meta-heuristic [2] composed by simulated annealing (SA) and sensitivity-based search. SA [38] is a famous stochastic search algorithm that can reduce the chance of being trapped at a local optimal solution. The stochastic nature of SA requires many iterations to obtain good coverage of the solution space. To overcome this drawback, we repeat multiple SA procedures with different initial solutions that are randomly distributed in the solution space. This is why our optimization is a multi-start approach. In general, it often takes many iterations for SA to reach near global optimal solution. In order to accelerate the convergence, we run a limited number of SA iterations and then take the best solution to start a sensitivity-based search. A sensitivity-based search is good at carefully searching a local solution space. Although it can be easily trapped at local optima in general, this weakness is largely avoided when combined with SA. Neither SA nor sensitivity-based searches depend on system models. Therefore, they can be directly applied with measurement based cost function.

The pseudo code for the multi-start meta-heuristic is provided in Algorithm 3. Vector $\vec{x}$ denotes decision variables $\{x_1, \ldots, x_l, \ldots\}$, where x_l is a multi-bit control signal for the lth tuning knob. The two tuning knobs for the BPF are R_{x_1} and R_{x_2} as described in Section 3.2.2. $\Phi(\vec{x})$ is the cost function to be minimized and is defined in (3.3). T is the temperature in simulated annealing, and t is the temperature decrease at each SA iteration.

```
Input  : Initial temperature $T_{max}$, cooling speed $t$, iteration limit $M$ in one
         procedure of SA search, max number of SA iterations $MAX\_SA$, max
         number of SS iterations $MAX\_SS$ and cost function threshold $\theta$
Output: The best solution $\vec{x}_{best}$ and the corresponding cost function value $\Phi_{best}$
1  Initialize $\Phi_{best} \leftarrow \infty$ and set global counter $i \leftarrow 0$
2  while $i < MAX\_SA$ and $\Phi_{best} > \theta$ do
3      if $mod(i, M) == 0$ then
4          $\vec{x}_{new} \leftarrow \vec{x}_{old} \leftarrow RANDOM; T \leftarrow T_{max};$
5          $\Phi_{new} \leftarrow \Phi_{old} \leftarrow \Phi(\vec{x}_{old});$
6      else
7          $\vec{x}_{new} \leftarrow Neighbor(\vec{x}_{old}, T);$
8          $\Phi_{new} \leftarrow \Phi(\vec{x}_{new});$
9          if $P(\Phi_{old}, \Phi_{new}, T) \geqslant random(0, 1)$ then
10             $(\vec{x}_{old}, \Phi_{old}) \leftarrow (\vec{V}_{new}, \Phi_{new});$
11         end
12     end
13     if $\Phi_{new} \leqslant \Phi_{best}$ then
14         $(\vec{x}_{best}, \Phi_{best}) \leftarrow (\vec{x}_{new}, \Phi_{new});$
15     end
16     $T \leftarrow T - t; i + +;$
17 end
18 for $i = 0, \Phi_{tmp} \leftarrow \infty$; $i < Max\_SS$ and $\Phi_{best} < \Phi_{tmp}$; $i + +$ do
19     $\Phi_{tmp} = \Phi_{best};$
20     for each neighbor $\vec{x}_j$ of $\vec{x}_{best}$ do
21         if $\Phi(\vec{x}_j) < \Phi_{best}$ then
22             $(\vec{x}_{best}, \Phi_{best}) \leftarrow (\vec{x}_j, \Phi(\vec{x}_j));$
23         end
24     end
25 end
26 return $(\vec{x}_{best}, \Phi_{best});$
```

Algorithm 3: Multi-start meta-heuristic.

If the cost function value is smaller than threshold θ, the SA search is terminated.

After the initialization, algorithm 3 begins with MAX_SA iterations of SA, which could be decomposed into $\lceil MAX\_SA/M \rceil$ fundamental blocks of SA search from Steps 3-16. Within each SA search block, it is only in the first iteration that the temperature T

is reset to T_{max}, and an initial solution $\vec{x}_{new}$ starts from a random selection. Other $M-1$ iterations in Steps 7-11 perform the key functionality of SA search. For example, in Step 7, a new candidate solution $\vec{x}_{new}$, which is T distance away from $\vec{x}_{old}$, is randomly selected for the subsequent evaluation. To be specific, assuming $\vec{x}_{old}$ has two decision variables (x_1, x_2), then the random update to (x_1', x_2') could be made by $(x_1', x_2') = (x_1, x_2) + (a, b)$ and $|a| + |b| = T$. The meaning of this process is illustrated in Figure 3.4 (a), where the signs of two random variables a and b define one of the four directions on the 2-Dimension plane, and their magnitudes denote the step size.

Steps 9-11 are the essential part of SA. The new candidate $\vec{x}_{new}$ derived from Figure 3.4(a) will be accepted with the probability $P(\Phi_{old}, \Phi_{new}, T)$, which is defined as $P(\Phi_{old}, \Phi_{new}, T) = \min(1, \exp(-\Delta\Phi/T))$, where $\Delta\Phi$ = $\Phi_{new} - \Phi_{old}$. As shown in Figure 3.4(b), it's clear that the new candidate $\vec{x}_{new}$ with smaller cost Φ_{new}, or say $\Delta\Phi < 0$, is always adopted; as for those higher cost candidates, they could be accepted with a probability which is related to the cost increment $\Delta\Phi$ and temperature T, i.e., with the same $\Delta\Phi$, $Prob_1$ is higher than $Prob_2$ when $T_1 > T_2$. Moreover, to avoid large circuit area, we implement the exponential function $\exp(-\Delta\Phi/T)$ by Taylor expansion $\sum_{i=0}^{m}(-\Delta\Phi/T)^m/m!$. By keeping only the 0th and the first order term, i.e., $m = 1$, it is approximated by $1 - \Delta\Phi/T$. As the probability precision is not critical to the solution search, such approximation is reasonable.

After multi-start SA reaches the iteration limit or the cost $\Phi(\vec{x}_{best})$ is within the acceptance range defined by θ, Steps 18-25 in Algorithm 3 continue as a local exploiter to perform MAX_{SS} iterations of sensitivity-based search upon the best solution $\vec{x}_{best}$ acquired from SA. The main effort is focused on Step 20, where *every immediate* neighbor of the current solution is examined. By “immediate", we mean the Hamming distance between $\vec{x}_j$ and $\vec{x}_{best}$ is 1 as depicted in Figure3.4(c). Different from multi-start SA search, we terminate SS immediately when there is no improvement to the cost function thus to avoid any redundant search.

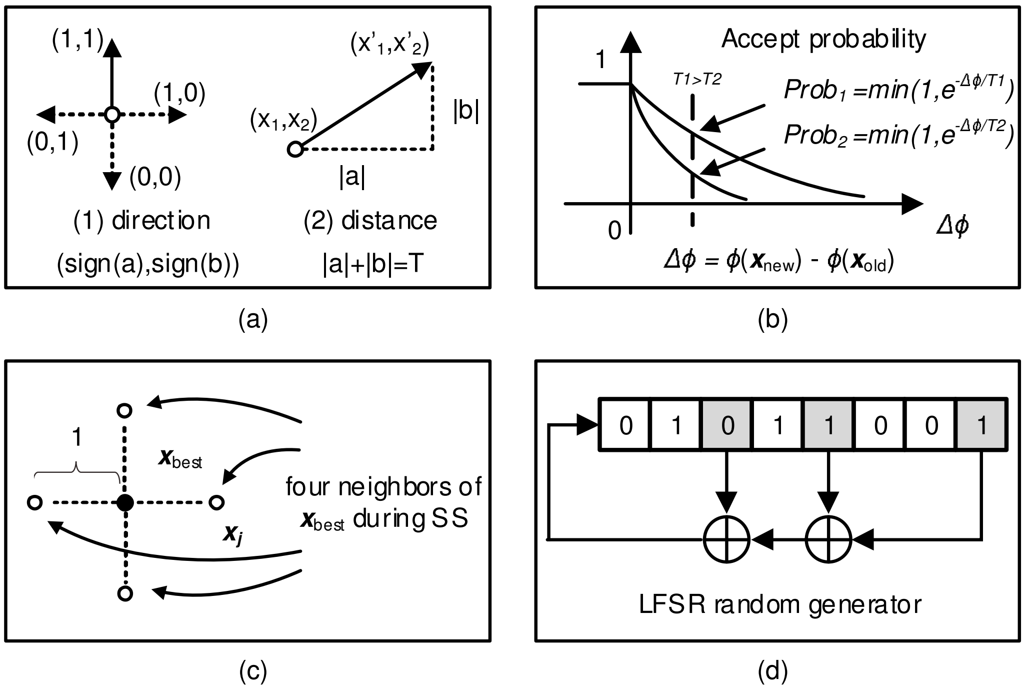


Figure 3.4: Specific processes during the multi-start meta-heuristic. (a) random update to SA neighbor (x'_1, x'_2) based on the old solution and temperature T in SA. (b) curve of SA acceptance probability based on $\Delta\phi$ and the temperature T. (c) all four neighbors around the $\vec{x}_{best}$ during the SS. (d) LFSR based pseudo random process.

Need to mention that, in Algorithm 3, all the random process related steps, such as Step 4,7,9, are realized by the pseudo random number generator based on the linear feedback shift register (LFSR) as drawn in Figure 3.4 (d). In our design, an 8-bit generator provides enough randomness in the proposed hybrid algorithm.

3.4.2 State Machine Diagram

The optimization engine is designed as a high-level state machine depicted in Figure 3.5. CF_ready and Φ_{new} are the input from cost function, $outputx$ is the solution delivered to BPF and others are all registers. This process starts from “Idle” when Reset signal is low and then keeps on waiting for the Φ_{new} in “Tuning_SA” after $\vec{x}_{new}$ is sent. Once Φ_{new} is available for reading by a notice of CF_ready signal, it’s used to update $\vec{x}_{best}$ in “Update_SA”. The acceptance probability is computed in “Compare” and shows whether to update $\vec{x}$ by $\vec{x}_{new}$ in “Accept_SA”. “Judge_SA” works as a controller to select “Neighbor” to continue in the current SA or “Multi_Start” to open a new SA or “Sensitivity” to begin the local search.

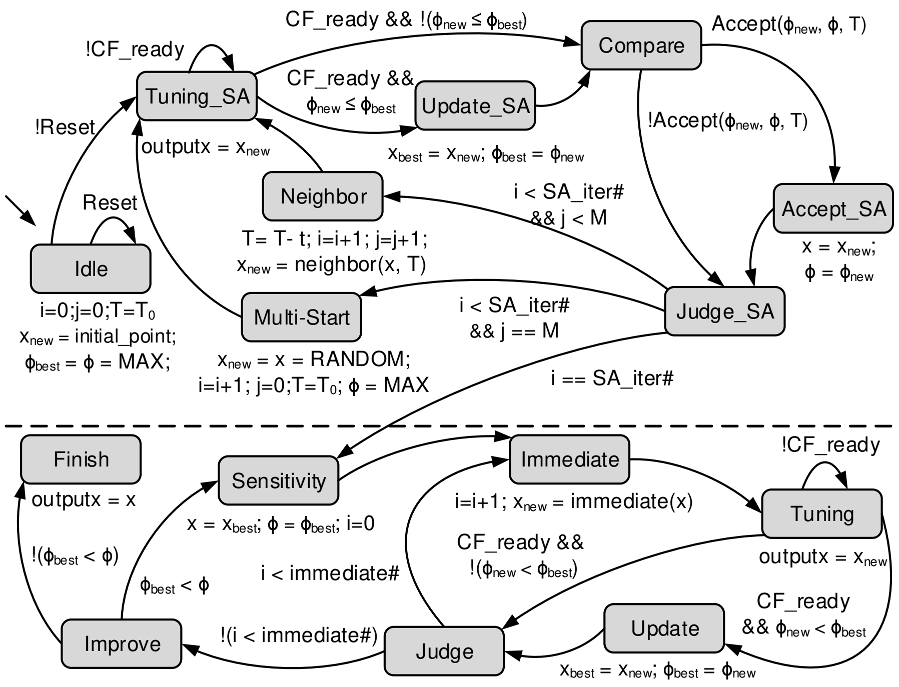


Figure 3.5: The high-level state machine of multi-start meta-heuristic, in which the upper multi-start SA and the lower sensitivity-based searches are divided by the horizontal dashed line. Reprinted from [2].

While in the “Sensitivity” state, Φ_{best} will be backup in Φ before enumerating each 1-distance neighbor in “Immediate”. The Φ_{new} of each immediate neighbor is recorded during “Tuning” state and helps to improve $\vec{x}_{best}$ in “Update”. “Judge” guarantees that all the neighbors are covered and then moves ahead to “Improve”. “Improve” compares the Φ_{best} with its old backup Φ and then decides to go on to “Sensitivity” or stop in “Finish” in which $\vec{x}$ is reported as the best found solution.

3.5 Reconfigurable Circuit Design and Implementation

3.5.1 Post-silicon Configuration

The art of the circuit design coming from the reconfigurability after the fabrication. For the cost function circuit in Figure 3.3, by changing the value of vector $\{\beta_1, \beta_2, \beta_3, \beta_4\}$, it could be reconfigured for different types of filters. For example, we set the control pattern as $\{1, 1, 1, 1\}$ for the band-pass filter. By changing the pattern to $\{0, 1, 0, 1\}$, we make the effect of g_{s_1} which is the gain calculated at the lower $3dB$ frequency point s_1, ignored from the Equation (3.3). After that, since the effect of the higher $3dB$ frequency point s_4 still exists, the modified Equation (3.5) now contributes as a low-pass filter. With similar analysis to pattern $\{1, 0, 0, 1\}$, it could be figured out as the control patterns for a high-pass filter due to the removal of effect at higher $3dB$ frequency point g_{s_4}.

$$\begin{aligned}\Phi_{LPF}(\vec{x}) &= \beta_2 \left| \frac{A(s_3)_{out}}{A(s_3)_{in}} - \alpha_2 \frac{A(s_4)_{out}}{A(s_4)_{in}} \right| \\ &\quad + \beta_4 \left| \frac{A(s_2)_{out}}{A(s_2)_{in}} - \alpha_4 \frac{A(s_3)_{out}}{A(s_3)_{in}} \right| + P \end{aligned} \tag{3.5}$$

The optimization engine in Algorithm 3 could also be configured to different algorithms by modifying the iteration parameters MAX_SA, MAX_SS and M. e.g., simply to let $MAX\_SA = 0$ or $MAX\_SS = 0$, this configuration will turn off one of the searching methods in the hybrid algorithm and form the stand-alone SS search or SA search accordingly. Furthermore, if $MAX\_SA < M$, the multi-start property of the SA search is prohibited, and thus leads to the single-start SA.

3.5.2 General Microprocessor

However, the above reconfigurability is still not enough to the requirement of a complete change of the cost function structure or the optimization algorithm for the calibration of a general CUT. Inspired from the Field Programmable Gate Array (FPGA), which is programmable after fabrication, we replace the digital circuits of cost function and optimization engine in Figure 3.1 with the reconfigurable chip structure depicted in Figure 3.6.

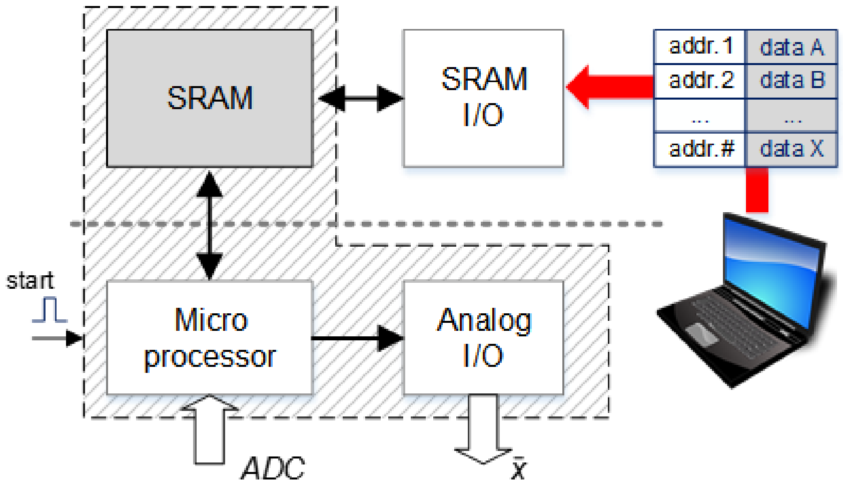


Figure 3.6: Components of the designed microprocessor and its working flow.

The microprocessor in Figure 3.6 consists of four components, namely the SRAM I/O, SRAM, micro processor and Analog I/O. It has two modes, i.e., "loading" and "working" modes which are differentiated by the horizontal dashed line. During the loading mode, only the SRAM I/O and SRAM components are activated, while in the working mode, the components included in the shaded area participate.

Especially, in the loading mode, the bit steams are serially loaded from the external computer to the SRAM I/O component, and parsed as SRAM address and the processorrecognizable data. Next the data part is loaded to the SRAM component with the guidance of the address through the internal parallel connections. When all the data is loaded, the microprocessor will receive a start signal and then switch to the working mode. In the second mode, the microprocessor retrieves binary data from SRAM, performs as the cost function (Section 3.3) and optimization engine (Section 3.4), and collaborates with the analog tested circuit by retrieving the ADC samplings and sending the tuning vector $\vec{x}$. The high-level state machine of the microprocessor is shown in Figure 3.7.

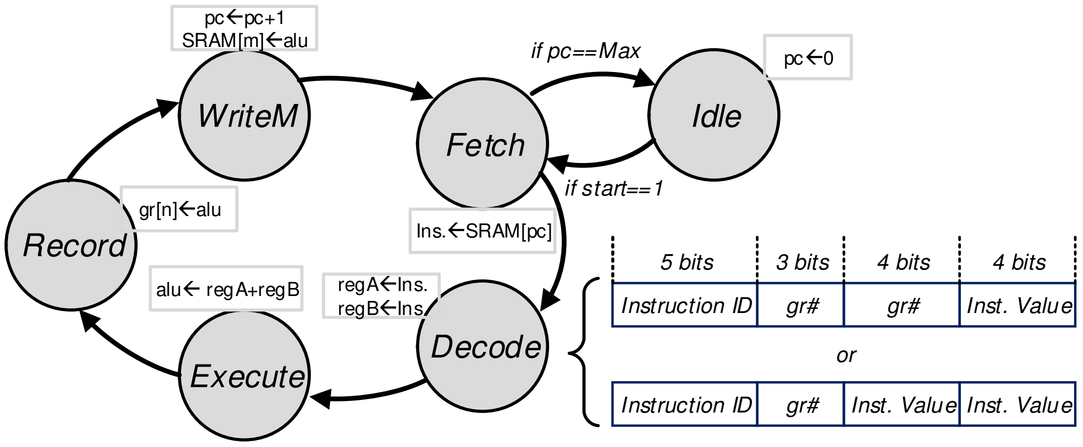


Figure 3.7: High-level state machine of the microprocessor and 16-bit instructions.

In Figure 3.7, the microprocessor stays at "Idle" with the pc pointer set to the zero address of the SRAM. When the start signal is high, the state machine begins the process loop which consists of five states. In the "Fetch" state, The SRAM data is fetched into the register ins from the address indexed by pc, and each SRAM address keeps 16 bits data. The next state is "Decode". During this state, the first 5 bits of the register ins is analyzed as the instruction ID, based on which one of the two instruction patterns is chosen to parse the rest bits. Then, two registers A and B are updated with the values patterned in the

11 bits. Later in the state “Execute”, some simple calculations, e.g., $A \pm B$, $A \oplus B$ etc., could be performed, and the result is saved in register alu. After that, in the “Record” state, the n-th element of the register array is checked and recorded as the value of alu if necessary. Finally, the m-th element of the SRAM is checked and written as the value of alu if necessary, while the pc pointer is also increased by 1 in the “WriteW” state. This five-state process loop continues until pc pointer reaches its limit. By then, “Fetch” state will jump back to “Idle” and keep waiting there for another start signal.

3.6 Experiment Results for Non-Reconfigurable Circuit

3.6.1 Test Chip Measurement Results

The proposed built-in self optimization system with BPF as CUT was fabricated using 180 nm IBM process technology. Measurement was performed on-chip to confirm that the system works as expected. Set the central frequency $f_c = 31MHz$ and the bandwidth $BW = 8MHz$. We first enumerated all combinations of the tuning knobs x_1 and x_2 and measured the BPF frequency response for each configuration. Based on the measurement results, we plotted the cost function $\Phi_{BPF}(\vec{x})$ in Figure 3.8. We found that the global minimum of the cost function is at $\vec{x} = (22, 13)$. The chip testing results showed that our optimization engine was able to find this global minimum solution.

We set the BPF according to the global minimum cost function where $\vec{x} = (22, 13)$, and measured the frequency response from $20MHz$ to $42MHz$. The results are plotted as the red curve with small circles in Figure 3.9. Its central frequency is near $31MHz$ and the $3dB$ drop frequencies are at $27.5MHz$ and $36MHz$. Thus, the bandwidth is $8.5MHz$, which is very close to the specification. We also measured and plotted frequency responses for two other solutions at $\vec{x} = (20, 13)$ and $\vec{x} = (12, 26)$. They are shown as the blue curve with small triangles and the black curve with small squares, respectively, in Figure 3.9. The solution at $\vec{x} = (20, 13)$ has a bandwidth of $7MHz$, which implies a small deviation

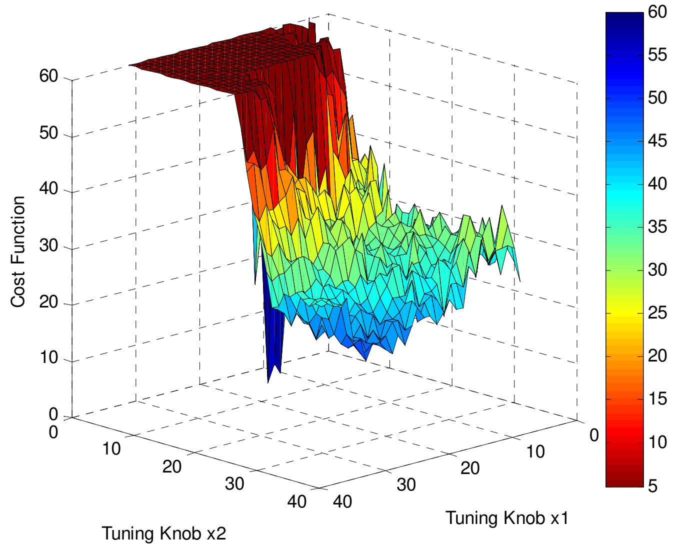


Figure 3.8: Cost function $\Phi(\vec{x})$ for decision variables $\vec{x} = (x_1, x_2)$ corresponding to R_{x_1} and R_{x_2} in the BPF with $f_c = 31MHz$ and $BW = 8MHz$. The measurements are based on the test chip. Reprinted from [2].

compared to the optimal solution, while the frequency response for $\vec{x} = (12, 26)$ is not only far from the optima, it is also not a BPF response. These results confirmed that our cost function definition leads to desired BPF performance.

3.6.2 Evaluation of Variation Resilience

To validate the effectiveness of our approach on handling variations, applying statistical results to different instances of the CUT are necessary. We performed the statistical analysis through Monte Carlo simulations. Based on the variation data collected from the circuit simulation of the SPICE model, a 5,000-run Monte Carlo simulation for the BPF design was performed. We evaluated the mean squared error (MSE) of frequency response, which is similar to results of (3.2). The probability density functions of the MSE before and after the self optimization are plotted in Figure 3.10. On average, the self optimization can reduce the mean of MSE by around 71.3%.

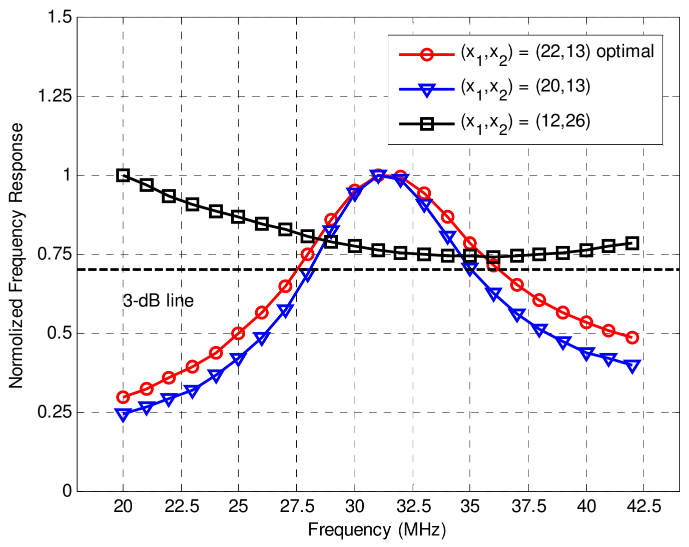


Figure 3.9: Frequency responses under different decision variables. The design target is the BPF with $f_c = 31MHz$ and $BW = 8MHz$. Reprinted from [2].

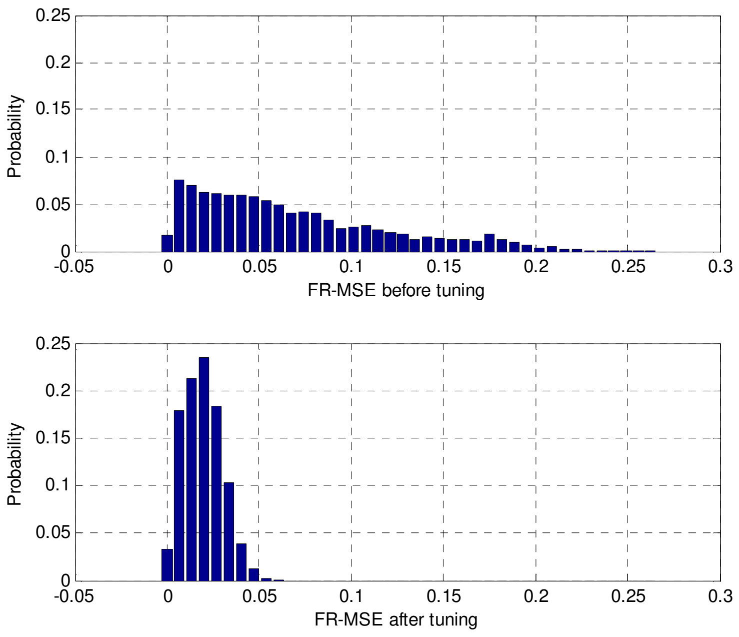


Figure 3.10: Probability density functions of frequency response mean squared error (MSE) before and after the self optimization from the 5,000-run Monte Carlo simulation on BPF with $f_c = 25MHz$ and $BW = 15MHz$. Reprinted from [2].

3.6.3 Verification of Reconfigurable Circuit

Instruction Based Cost Function

In order to verify the cost function implemented by the microprocessor, we compare it with the multi-start meta-heuristic circuit towards the CUT of BPF. Similar to the measurement in Section 3.6.1, we enumerated all the combinations of the tuning knobs x_1 and x_2, measured the BPF frequency response for each configuration, and saved the response data into a look up table (LUT) indexed by the tuning vector (x_1, x_2). Here, the BPF that we measured has central frequency $f_c = 74MHz$ and the bandwidth $BW = 13MHz$. Based on the measurement, the cost function $\Phi_{BPF}(\vec{x})$ is plotted in Figure 3.11, and the global minimum is 79 at $\vec{x} = (23, 11)$.

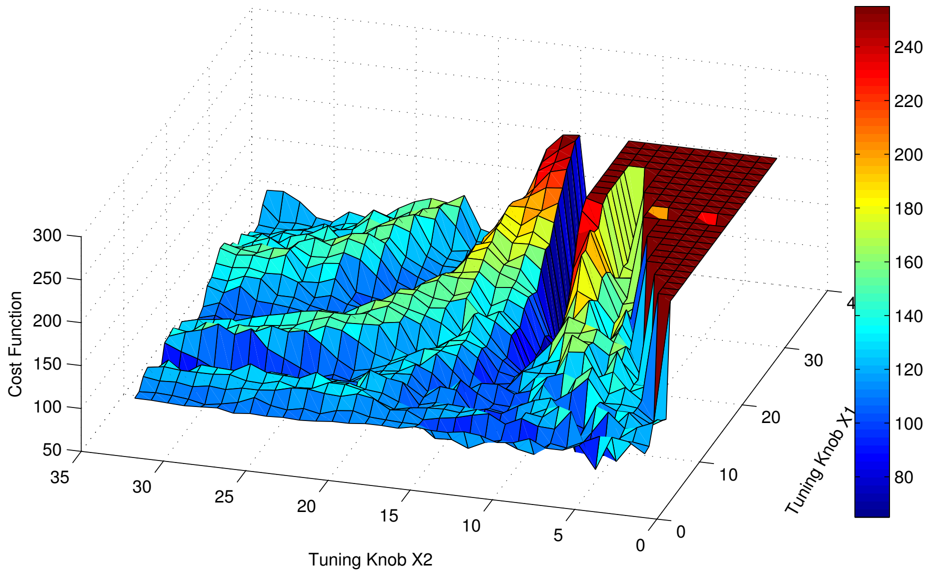


Figure 3.11: Cost function $\Phi(\vec{x})$ for decision variables $\vec{x} = (x_1, x_2)$ corresponding to R_{x_1} and R_{x_2} in the BPF with $f_c = 74MHz$ and $BW = 13MHz$. The measurements are based on the test chip.

Next, we reprogrammed the microprocessor with the instructions which implement the functionality of cost function, enumerated all the combinations of the tuning knobs x_1 and x_2, searched each corresponding data from the LUT according to each tuning vector (x_1, x_2), and provided the LUT data to the microprocessor. Based on these steps, the cost function is depicted in Figure 3.12.

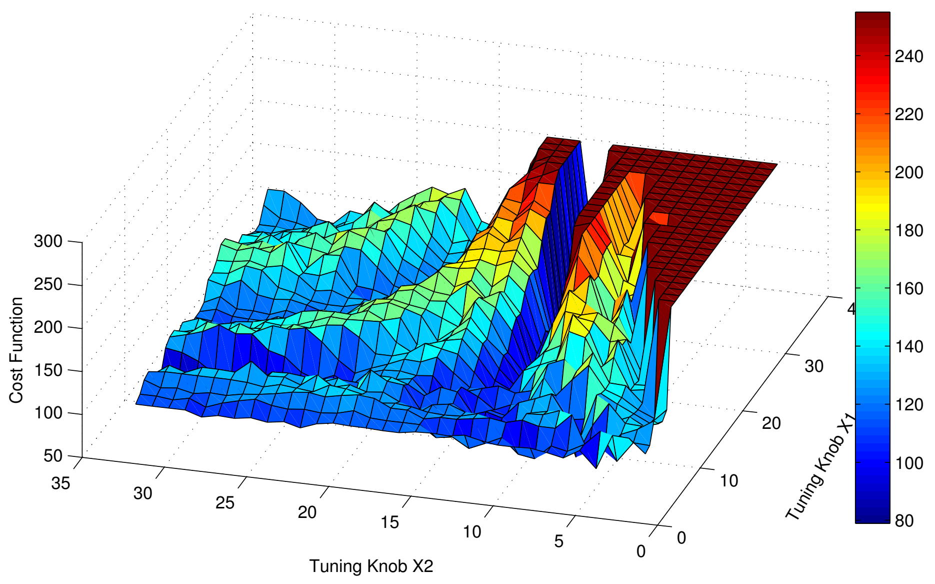
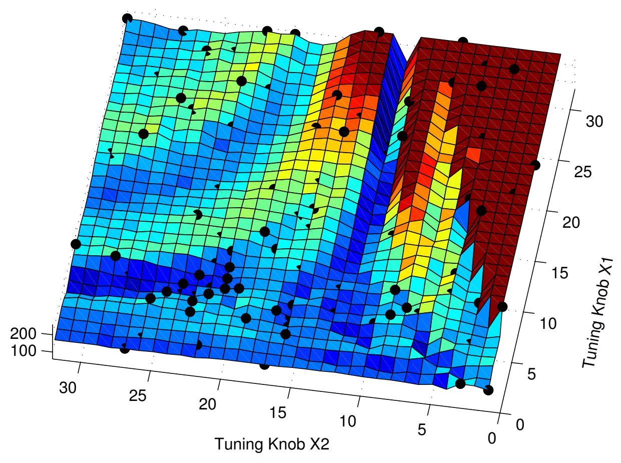


Figure 3.12: Cost function curve calculated by the microprocessor with the same configuration in Figure 3.11.

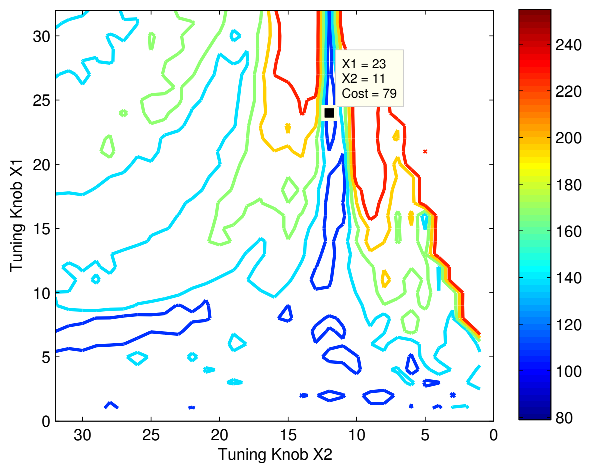
Obviously, the cost function curve calculated by the microprocessor is almost the same as that derived from the integrated circuit. The optimal point which locates at $\vec{x} = (23, 11)$ in Figure 3.12 is identical to that in Figure 3.11.

Instruction Based Optimization Engine

To verify the microprocessor could work as the optimization engine, we reprogrammed the microprocessor with the instructions of the multi-start meta-heuristic algorithm. With the SA iteration $MAX\_SA = 160$, the temperature $T_{max} = 32$, the cooling speed $t = 2$, iteration limit $M = 16$, the SS iteration $MAX\_SS = 10$, and the cost function threshold $\theta = 1$, the microprocessor found the optimal point (23,11) at the 66th iteration. The search trace of the optimization algorithm is shown on the cost function curve in Figure 3.13, where each black point denotes a tuning variable generated by the microprocessor.



(a) Multi-start Meta-heuristic search trace.



(b) Cost function contour and the global minimal point.

Figure 3.13: Search trace and optimal point found by the microprocessor.

3.6.4 Comparison of Algorithm Performance

We further compared our multi-start meta-heuristic with two other optimization approaches - standalone multi-start sensitivity-based search (Mul-Sen) and standalone SA. Figure 3.14 shows the cost function value changes over iterations for these methods. It can be found that the proposed hybrid approach converges to a better solution in less evaluation iterations.

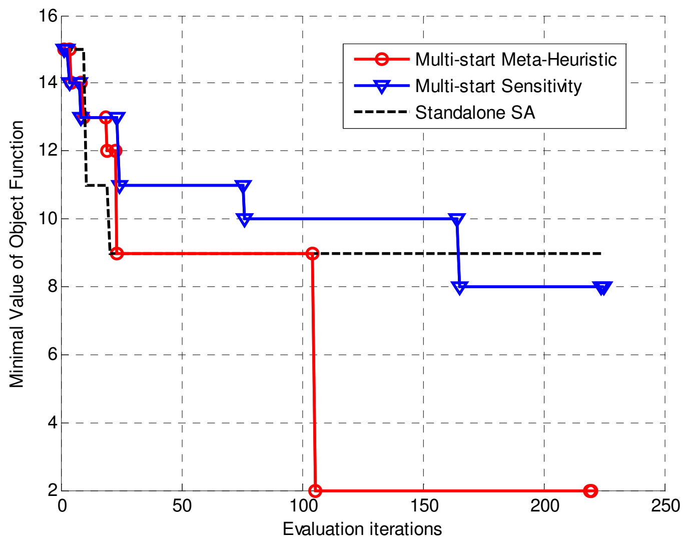


Figure 3.14: Cost function changes over iterations for our multi-start meta-heuristic, multi-start sensitivity and standalone SA on BPF with $f_c = 23MHz$ and $BW = 6MHz$. Reprinted from [2].

As a heuristic algorithm, the proposed method has some probability of failing to match the desired frequency response. The failure rate and the error of the outcome were evaluated by performing on 4,000 BPFs, whose central frequencies increase from $12MHz$ to

$31 MHz$ and Q-factors change from 1 to 4. The cumulative distribution functions from all three methods are plotted in Figure 3.15, where the horizontal axis indicates the percentage error from the optimal solution. The results showed that the proposed approach produced more accurate solutions with lower error. Particularly, 77.6% of the solutions given by the hybrid algorithm have an error rate of less than 1%. In contrast, the standalone SA and the standalone sensitivity-based search had only 52.7% and 49.6% solutions with an error rate of less than 1%, respectively.

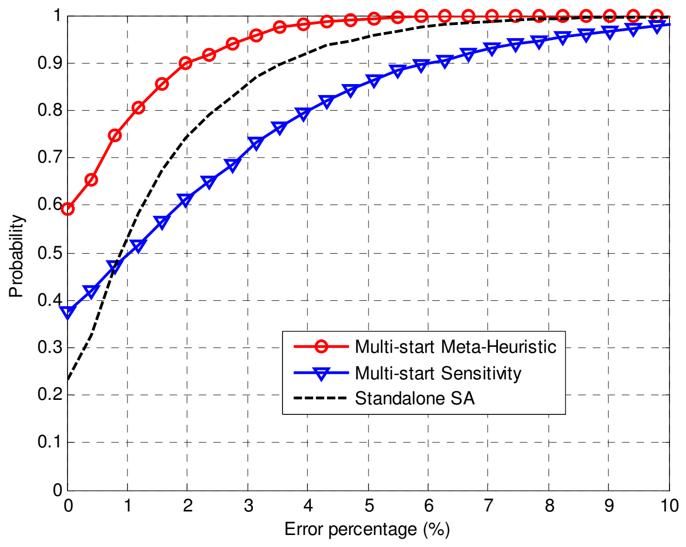


Figure 3.15: Cumulative distribution functions of solution errors vs. the optimal solution from the simulation performed on 4,000 BPFs. Reprinted from [2].

The area and power consumption comparisons among different meta-heuristics are summarized in Table 3.2, where each gate count includes 3509 logic gates in the cost function circuit. A $10 MHz$ clock is used to drive the simulation, and only the processing time for optimization engine is recorded. Considering that a medium performance ASIC chip may have millions of gates, thousand gates is a very small chip area overhead. Besides, since the self optimization is conducted very occasionally for a chip, the power is also acceptable. As for the processing time, although the proposed method is almost three times slower than Mul-Sen, it is still feasible because the total time for analog circuits being stable after each tuning is the dominant part in the whole process.

Table 3.2: Gate count, power and processing time comparison. Reprinted from [2].

	Gate Count	Power	Processing Time
Multi-start Meta-Heuristic	6744	$1.15mW$	$352.8\mu s$
Multi-start Sensitivity	4817	$0.56mW$	$95.6\mu s$
Standalone SA	5439	$0.93mW$	$335.8\mu s$

As for the microprocessor, since all the functionality of cost function and optimization engine could be implemented by instructions (each of our instruction has 16 bits data, or namely 2 Bytes), we evaluate the overhead of different algorithms in terms of the SRAM memory consumption. Note that, the total SRAM memory listed in Table 3.3 is 1024 Bytes, and a 246 Bytes' cost function nests in each of the algorithm. In our design, the clock frequency is $50MHz$ for the whole circuit during the working mode, while in the loading mode, the frequency is reduced to $1MHz$ for scan chains which serially load instructions into the SRAM.

Table 3.3: SRAM memory space consumption for microprocessor.

	Memory (Bytes)	Percentage
Multi-start Meta-heuristic	626	61.1%
Multi-start Sensitivity	415	40.5%
Standalone SA	450	43.9%

3.6.5 Die Photographs for Tapeout Chips

Case 1: Integrated Cost Function and Optimization Engine

The integrated cost function/optimization engine design is fabricated in 180nm standard CMOS technology, and the $1.8mm \times 1.8mm$ chip die photograph is shown in Figure 3.16. Compared with the analog circuit which totally takes $236,300\mu m^2$, the digital area is $165,753\mu m^2$.

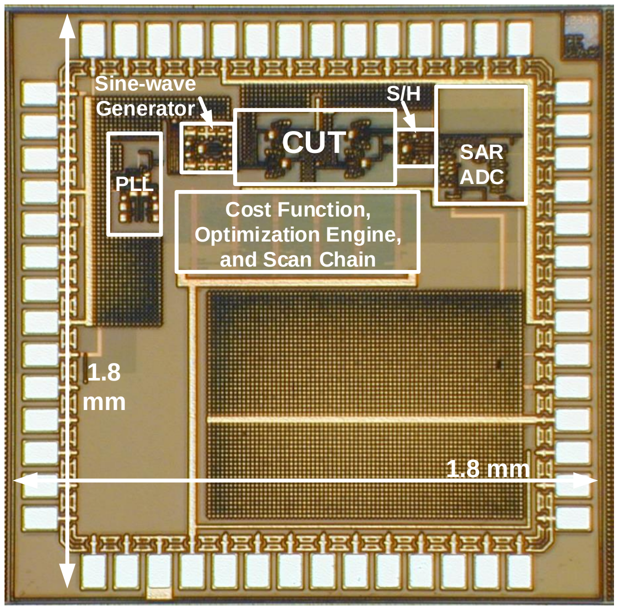


Figure 3.16: Chip die photograph of the proposed built-in self-optimization system. CUT includes the active-RC BPF, and the digital circuits are fully integrated in one block.

Especially, the scan chains used for chip testing are considered as the digital components which contribute 33% to the whole digital area consumption. Thus, the area of the kernel part of the digital circuit including the cost function and optimization engine is just

$111,054\mu m^2$. Therefore, the area ratio between the kernel part of digital circuit and the analog circuit is about 1:2. The meaning of this ratio represents the area overhead caused by the additional circuit with respect to the original design. Obviously we hope to make this ratio as small as possible. One of the approach is to share the digital tuning circuit with more than one analog CUT. By this way, the analog area will relatively increase while the ratio could be reduced, and it makes the application of this proposed automatic tuning mechanism feasible.

Case 2: Reconfigurable Microprocessor with SRAM

Figure 3.17 shows the $1.5mm \times 1.5mm$ chip die photograph of the reconfigurable circuit design fabricated in 130nm standard CMOS technology. For the test purpose, the analog counterpart is not integrated but only the digital circuit which includes the microprocessor, the SRAM, and scan chains.

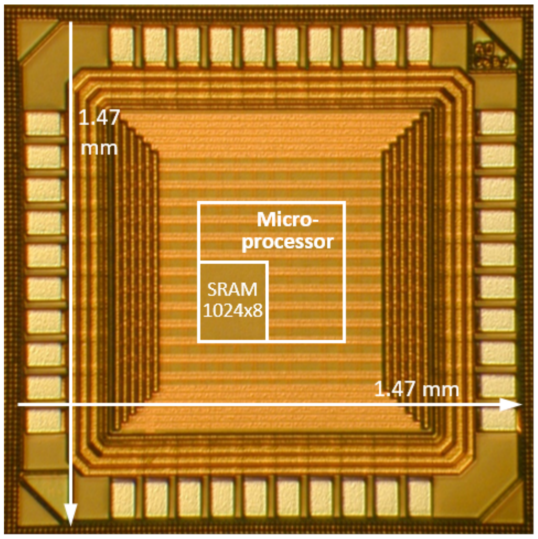


Figure 3.17: Chip die photograph of the reconfigurable optimization system including the microprocessor and on-chip SRAM 1024x8 bit.

Although the digital circuit locates in a $400 \times 400\mu m^2$ region in Figure 3.17, the actual area is just $79,598\mu m^2$ due to a floor plan usage ratio as low as 35%. This low usage ratio is reasonable in the test chip just for proving the concept of our reconfigurable design.

In order to show the benefit of the reconfigurable design with respect to the integrated design in the same technology, we synthesize the digital circuit in ***Case 1*** by using the 130nm technology, and make its area decreases from $165,753\mu m^2$ to $71,088\mu m^2$. Therefore, it is straightforward that the reconfigurable design ($79,598\mu m^2$) only increases 12% of the digital area but provides a reconfigurable feature to the fabricate chip. The designer could choose either the ***Case 1*** design or the ***Case 2*** design at his/her own discretion.

4. THWARTING ANALOG IC PIRACY VIA COMBINATIONAL LOCKING*

4.1 Introduction

In addition to the interior technique challenges of IC design discussed in the previous chapters, the exterior security challenges of IP infringement keep on threatening the semiconductor industry. According to SEMI [39], a semiconductor industry consortium, the annual loss due to semiconductor IP infringement is up to $4 billion. Besides the tremendous economic lost, these products cause even serious casualties when applied for medical or military. A further study by IHS technology [7] indicates that analog integrated circuit is the topmost counterfeited among all semiconductor products. An analog IC typically has hundreds to thousands of transistors of relatively big size while a digital IC could easily contain millions to billions of usually smaller transistors. Thus, it is conceivable that this difference makes analog ICs an easy target of reverse engineering, which is a main approach of chip piracy and counterfeit [40].

Interestingly, most previous works on hardware security are focused on digital ICs while the security of the topmost counterfeited IC product has received much less research attention. One related work is [41], which applies split manufacturing to RF circuits for security defense at untrusted foundry. In [42], a locking technique is introduced for sense amplifiers in memory circuits to hamper the evil-maid attack [43]. This technique relies on the use of memristor, whose manufacturing process is not always available, and therefore is restrictive in applications. The idea of combinational locking for analog biasing circuit is also reported in [44]. However, it does not show how to make correct key unique and ensure significant performance degradation for incorrect keys.

In this work, we suggest a combinational locking technique for analog IC security [3].

*Reprinted from "Thwarting Analog IC Piracy via Combinational Locking", by Jiafan Wang, Congyin Shi, Adriana Sanabria-Borbon, Edgar Sánchez-Sinencio, Jiang Hu, ©2017 IEEE.

It will mainly defend against reverse engineering and recycling-based counterfeit. The core idea is to make current mirror, a component existing in many analog ICs, configurable and the configuration allowing correct system operation is decided by a digital key. Without the correct key, it is very difficult to make reverse-engineered or recycled chip work properly. The locking will also increase the difficulty of piracy and over-production at foundry, provided that manufacturing test is conducted at separated service company. To the best of our knowledge, this is the first general locking technique for analog ICs, where the mainstream CMOS devices suffice and no memristor or other special process is required.

Combinational locking is a kind of logic locking technique [45, 46, 47] that is originated from digital IC security[†]. However, the locking design for analog ICs is quite different from that in digital chips, and in fact significantly more difficult. In a digital circuit, a single bit error in the key can easily result in malfunction of entire system. In contrast, a small error in configuration of analog circuit often causes limited deviation in performance. Only when the error is large enough, functional failure may happen. As such, many wrong inputs to the key merely lead to performance degradation of various degrees. To overcome this difficulty, we make use of Satisfiability Modulo Theories (SMT) to carefully design the current mirror configuration circuits such that most wrong inputs result in large deviation or unacceptable performance. By leveraging existing chip identification techniques, we can further make the correct key to each chip instance distinctively unique.

Attacks to the locking defense are quite different for analog and digital ICs. First, it takes much longer time to evaluate analog circuit responses for each attempt of key input. Second, advanced attack techniques to digital ICs are mostly based on Boolean logic [47], which are not applicable in analog domain. Overall, locking defense for analog ICs exhibits considerably different characteristics from its digital counterpart and our work

[†]A taxonomy of counterfeit digital ICs and review for security techniques are provided in [40].

is remarkably distinct from the existing techniques for digital circuits.

The proposed technique is implemented and simulated on band-pass filter, class-D amplifier and other analog designs. The results show that our technique can generate a unique key that enables desired performance while all incorrect keys result in large deviation of circuit characteristics. In addition, circuit output is highly non-monotone with respect to key values and therefore systematic attack becomes very difficult. Short key bitwidth is used for small circuits and manual attack to such protection takes a half month to unlock one specific chip instance. Long key bitwidth is applied to relatively large circuits and even automated attack normally needs more than a year to unlock entire design. The area overhead of our technique is usually a few percent.

4.1.1 Previous Works

There are very few previous works on security of analog ICs. A split manufacturing technique for RF circuits is proposed in [41] for security defense against untrusted foundry. The work of [42] is a locking technique that uses a memristor-based voltage divider to bias the body voltage of transistors in an amplifier. The voltage divider output is programed by a memristor crossbar, which can be properly configured only by a correct 16-bit key. This scheme conceptually works well, but its practical applicability is quite restrictive due to its dependence on memristor, which is not widely available yet. The most recent work [44] proposes the idea of combinational locking on biasing current of analog circuits, which is similar as ours. A straightforward realization of this idea can easily have three drawbacks. First, there could be multiple correct keys and therefore the security effect is weakened. Second, the circuit performance degradation from an incorrect is small and again the security from such locking is quite limited. Third, all chip instances share the same keys. As such, a successful attack to one chip instance implies unlocking of all chip instances. These issues are not discussed in [44] while they are main focus of our work.

In the past, locking techniques are mostly for digital circuits. Although they are largely different from our technique for analog circuits, a brief review is provided here to show the related rationale. An early work of logic locking is [45], where unused states in finite state machine (FSM) or additional FSMs are used to configure a circuit into a locking state at power-on or reset, and only certain digital key can change the circuit to normal operation states. Later, the work of [46] suggests to use additional XOR gates instead of FSM for the locking. An original signal in a circuit is XOR with a key signal and the XOR output is the same as the original signal only when the key is correct. Multiple such XOR gates constitute a long word key. Moreover, the public key technology in cryptography is applied in [46] such that even the foundry is not able to unlock a chip without IP owner's permission. It is noticed in [47] that an attacker can purchase a functional chip from market and compare with the chip under attack. Then, by sensitizing a key input according to manufacturing test principle, the attacker may observe the correct key value from the functional chip. Such attack takes linear time with respect to the key bit-width. By increasing the interactions among key bits, the work of [47] can restore the attack complexity back to exponential. More recently, locking techniques are further geared toward defense against Trojan insertions [48]. In [49], a SAT-based attack method is introduced and can successfully unlock many circuits defended by the aforementioned techniques.

4.2 Overview and Scope of This Work

An overview of the locking system for analog ICs is depicted in Figure 4.1. It makes the current mirror, a component available in many analog ICs, configurable. Only the correct configuration allows the entire system to function properly. The correct configuration is specified by a common digital key, which is shared by all chip instances of the same design.

Locking by a common key is often insufficient. If an attacker manages to obtain the

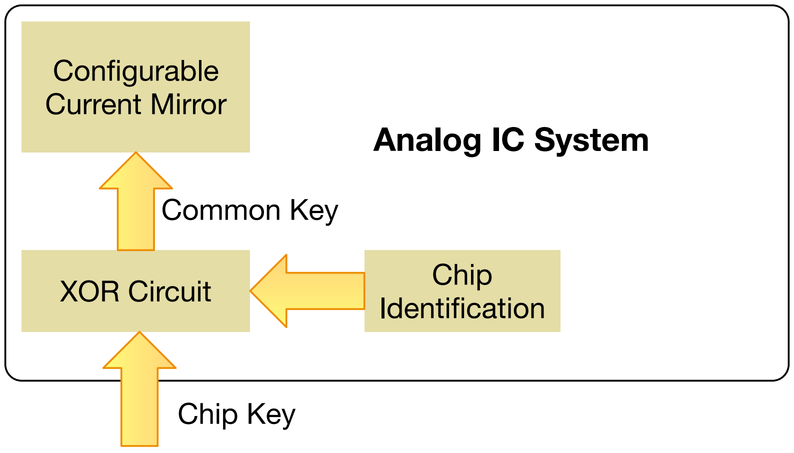


Figure 4.1: Locking system overview. Reprinted from [3].

common key, the attacker is able to unlock all chips of this design. In order to enforce a distinctively unique key for each individual chip, we leverage the chip identification technique as in digital circuits [45]. For each chip instance, we can obtain its unique identification using existing techniques [50, 51]. Chip identification is XOR with chip key to produce the common key so that each chip key is distinctively unique. For example, if the common key for a design is 1010, consider chip A with identification 1100 and chip B with identification 1001. By the design, the chip keys for A and B are 0110 and 0011, respectively, as $0110 \oplus 1100 = 1010$ and $0011 \oplus 1001 = 1010$. Please note the common key is enforced through the configurable current mirror without explicit storage anywhere and a chip key is provided to only authorized user of the specific chip.

The centerpiece of the locking design, configurable current mirror, is very different from locking in digital circuits. In digital circuits, a single bit error at the key input can easily result in malfunction of the entire circuit. In contrast, a small change of the current mirror may just cause small performance deviation and can be far from locking an analog circuit system. We solve this difficulty using SMT, which will be elaborated in later sections.

The proposed locking technique is mostly to defend against reverse engineering and recycle-based counterfeit. This is different from most locking techniques for digital circuits, which emphasize protection from piracy and over-production at foundry. However, our technique can help the security at foundry for the case where manufacturing test is conducted at a separated service company. This case will be discussed with more details in Section 4.3.

Our locking technique intends to be applied for relatively large analog and mixed signal designs. For small designs, the overhead is not well justified. The overhead includes the area of configurable current mirror, XOR circuit, chip identification generation (or storage), and additional I/O pin. By serialization, the multi-bit key can be loaded through a single I/O pin.

4.3 Attack Analysis

Reverse engineering is a primary attack to analog ICs and therefore the major security scenario that our work is focused on. In reverse engineering, attackers polish chips layer by layer and attempt to restore circuit netlist according to the layout observed at each layer [52]. With the proposed locking system, even a netlist is reverse engineered and chips are reproduced illegally, these chips cannot operate properly without knowing the common key or how to configure the current mirror. The time and effort for recovering the correct key is substantial. When attackers attempt to sell recycled analog IC chips with the locking system, they cannot demonstrate that the chips work properly unless they obtain the keys for these chips.

Although piracy at foundry is not the main scenario of our defense, but our locking system still benefits the case where manufacturing test is performed at a separated service company. If the manufacturing test is not conducted at the foundry, the common key is not provided to the foundry. As such, the foundry cannot unlock the circuits unless its spends extra effort to reverse engineer the key value. Hence, the foundry piracy becomes more difficult.

We further discuss possible attack methods and our defense under these attacks as follows.

- Brute-force attack. An attacker tries all combinations of key values and evaluates the circuit response for each of them to find the correct key value. Typically, evaluating analog circuit responses, such as frequency response and settling time, is orders of magnitude slower than that for digital circuits. As such, brute-force attack to a 32-bit key would normally take one year. Therefore, our locking technique is quite effective in defense against brute-force attack in reverse-engineering.
- ATPG and SAT attack. The work of [47] suggests to apply ATPG technique to sensitize one key bit and observe the output at a functional chip purchased from market. It reduces attack complexity from exponential to linear with respect to the bit-width of key. In [49], a SAT-based attack is developed assuming that attackers have complete access to circuit netlist. Its simulation results show that the SAT attack is quite successful in unlocking designs protected by many locking techniques. These attacks are based on Boolean logic and incompatible with analog circuits. More specifically, analog output responses, such as gain and linearity, are different from those in digital circuits, and the conversion from one to the other is not straightforward. Hence, it is not obvious how to launch ATPG/SAT attacks to analog circuits protected by our locking system.
- Optimization-based attack. An attacker can use optimization algorithms, such as simulated annealing and genetic algorithm, to search for the correct key value that minimizes the difference between the actual circuit output and specification. In general, such attack is effective when the circuit output is well behaved with respect

to key values. The well behavior here means the output function exhibits certain pattern. In our defense, we deliberately design the locking system such that the output-key function is close to random noise. As such, an optimization attack is still inefficient and requires very long time to find the correct key value.

- Smart guess by experienced analog IC designer. In such attack, an experienced analog IC designer can make smart guess on the correct configuration of current mirror. This is somewhat like asking the designer to complete a partial design. Our locking technique cannot completely defeat such attack, but can raise the bar that makes the attack non-trivial compared to unprotected designs. Relying on experienced designer has already made the attack expensive or restrictive.
- SMT attack. SMT is a formal verification technique we employ to design the locking system. One may consider if an attacker can use the same technique to break the locking system. The answer is that an attacker can do so under a very restrictive condition. That is, the attacker needs to know the desired current value of the current mirrors. Since this value is used only during the original design and not disclosed in system specification, the attacker has to make guess and rely on design experience. Having both analog design experience and formal verification knowledge is evidently a tough requirement to attacks.

Overall, our locking technique can considerably increase the difficulty of reverse engineering and recycle-based counterfeit attack to analog ICs.

4.4 Current Mirror and Its Role in Analog ICs

4.4.1 Basic MOSFET Current Mirror

Current mirror (CM) is a basic circuit block that provides current bias to enable proper operations of many different types of analog circuits.

A simple current mirror structure is depicted in Figure 4.2. All transistors operate in

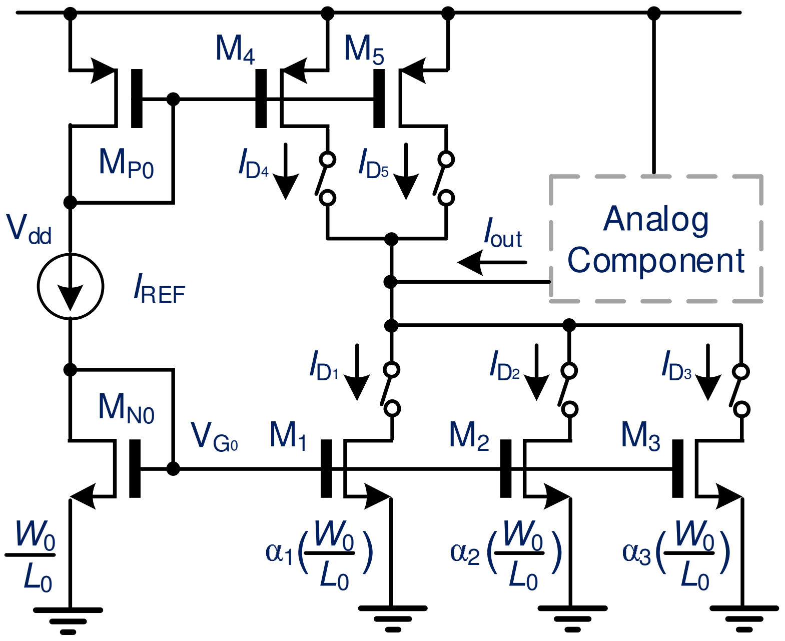


Figure 4.2: Basic MOSFET current mirror with five finger branches. Reprinted from [3].

saturation mode, since their drain-to-gate voltage $V_{DG} = 0$. The reference current I_{REF} is equal to the drain current I_{D_0} and satisfies [53]

$$I_{REF} = I_{D_0} = \frac{1}{2} K_0 (\frac{W_0}{L_0})(V_{GS_0} - V_{th_0})^2 \tag{4.1}$$

where K_0 is a technology-specific constant, W_0 and L_0 are respectively the channel width and the channel length of transistor M_{N_0}. Since the gates of all transistors are tied together, $V_{GS_0} = V_{GS_1} = \cdots$, we can derive the drain current of M_1 as

$$I_{D_1} = \frac{\frac{1}{2} K_1 (\alpha_1 \frac{W_0}{L_0})}{\frac{1}{2} K_0 (\frac{W_0}{L_0})} I_{D_0} = \alpha_1 I_{D_0} \tag{4.2}$$

where α_1 is the size ratio between M_1 and M_0. Similarly, by applying different widths of $\alpha_i W_0$ for M_i, different output current I_{D_i} can be obtained for the i-th branch.

Conceptually, branches of PMOS transistors M_4, M_5 provide negative bias current.

The negative bias current can be exploited to create non-monotone behaviors in our locking system and thereby facilitate improved security.

4.4.2 Importance of Bias Current

Bias current largely determines the performance of analog circuits through transconductance (g_m) of MOSFET transistors [53]. Transconductance g_m can be estimated by

$$g_m = \frac{dI_D}{dV_{GS}} = \sqrt{2K\frac{W}{L}I_D} \tag{4.3}$$

This is a fundamental parameter for most analog circuits. For instance, the DC gain of a differential pair is defined as

$$A_v = g_m \cdot R_L \tag{4.4}$$

where R_L is the equivalent load resistance of the differential pair. The gain-bandwidth product (GBW) is defined as

$$GBW = \frac{g_m}{2\pi C_L} \tag{4.5}$$

where C_L is the load capacitance. Moreover, the root mean square (RMS) thermal noise current density of a transistor is also a function of g_m,

$$i_n^2 = 4\beta T\gamma g_m \tag{4.6}$$

where β is the Boltzmann constant, T is the temperature, and γ is a process-specific constant.

It should be mentioned that analog circuits are usually designed in a top-down manner. On one hand, we can generate differential pairs (simple amplifiers) from basic MOSFET transistors. Amplifiers are then used as the building blocks for constructing more complicated analog systems, such as filters, oscillators, buffers, low-dropout (LDO) regulators, and so on. On the other hand, the bias current affects the g_m of a MOSFET, and then Equation (4.4)-(4.6) can be obtained based on g_m. These parameters further affect the performance of higher level systems, e.g., filter transfer function, oscillator's output frequency, buffer's drivability, LDO's stability, etc. To conclude, bias current is so critical that any significant change on it would remarkably improve or degrade the performance of entire analog IC system.

4.4.3 Application in Gm-C Band Pass Filter

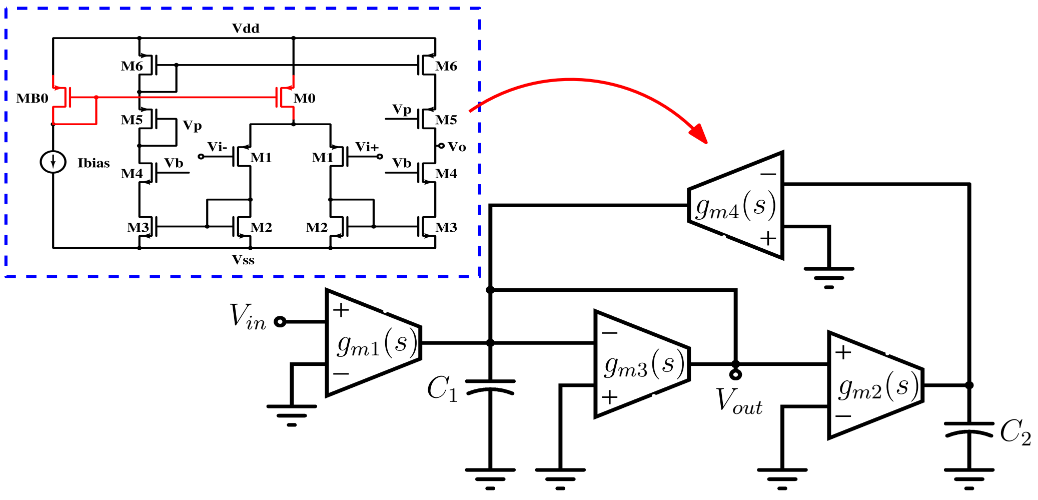


Figure 4.3: Gm-C implementation of BPF with differential amplifiers. Reprinted from [3].

As a study case, consider the Gm-C biquad filter shown in Figure 4.3. This second order structure is formed by four operational transconductance amplifier (OTA) and two capacitors [54]. Figure 4.3 also presents one possible transistor level implementation of each OTA. Its transconductance is a function of the tail current of the input differential pair, which is proportional to the bias current. The proportional factor is the current mirror ratio, which can be modified to make the transconductance tunable in a defined range while keeping all transistors operating in the proper region.

The operation of this circuit is based on current-to-voltage transformations and vice versa. In Figure 4.3, the transconductance g_{m_1} converts the input voltage into current. Then, that current is integrated in capacitor C_1. The transconductor g_{m_3} is connected in unity feedback in order to mimic a resistor. The combination of g_{m_1}, g_{m_3} and C_1 forms a lossy integrator structure. In a similar way, the output current of g_{m_2} is integrated in C_2 to the lossless integrator. The negative feedback loop is completed with g_{m_4}. The transfer function (at the band pass output) is given by

$$\begin{aligned} H_{Gm-C}(s) &= \frac{V_{out}(s)}{V_{in}(s)} = \frac{G_{BPF}\frac{\omega_0}{Q}s}{s^2 + \frac{\omega_0}{Q}s + \omega_0^2} \\ &= \frac{g_{m_1}C_1 s}{s^2 C_1 C_2 + g_{m_3}C_2 s + g_{m_2}g_{m_4}} \end{aligned} \tag{4.7}$$

according to [54] and we have

$$G_{BPF} = \frac{g_{m_1}C_1}{g_{m_3}C_2}, \quad \omega_0 = \sqrt{\frac{g_{m_2}g_{m_4}}{C_1C_2}}, \quad Q = \frac{1}{g_{m_3}}\sqrt{\frac{g_{m_2}g_{m_4}C_1}{C_2}}$$

where G_{BPF} is the gain, ω_0 is the central angular frequency, and Q is the quality factor of the BPF. Particularly, if let $C_1 = C_2 = C$ and $g_{m_1} = g_{m_2} = g_{m_4} = g_m$ then we theoretically have $G_{BPF} = g_m/g_{m_3}$, $\omega_0 = g_m/C$, and $Q = g_m/g_{m_3}$.

Therefore, the parameters of the filter are determined by the values of all g_m, and they will be affected by the size ratio of current mirror based on Equation (4.3). Thus, in our proposed work, the correct bias current is the armor that we employed to protect our design.

4.5 Configurable Current Mirror for Locking

Making a circuit configurable is a common approach to locking-based hardware security, especially for digital ICs [46, 55]. However, a straightforward application of this technique for analog ICs faces significant difficulties.

4.5.1 Difficulties of Naïve Configurable Design

Suppose we need to design a current mirror that provides current I^*. To make the current mirror configurable, we can split the current path into multiple branches, each of which can be turned on/off by an additional transistor switch controlled by a binary bit. All the control bits together form the combinational lock. A such simple design with 4-bit control is shown in Figure 4.4, where the four transistors SW_1, SW_2, SW_3 and SW_4 are controlled by digital key lines q_1, q_2, q_3 and q_4, respectively. The rest of the current mirror design is the same as in Figure 4.2.

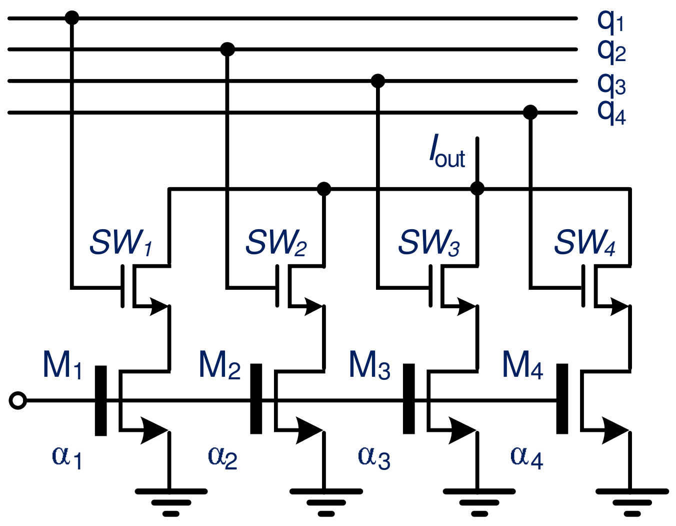


Figure 4.4: A naïve design of configurable current mirror. Reprinted from [3].

The current provided by this circuit is decided by the transistor sizes and the value of control key. Let us start with the simplest case where all four branches have the same transistor size, e.g., $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 1$. If the desired current $I^* = 3I_{REF}$, then we need to turn on exactly 3 branches. There are totally 16 combinations of keys and 4 keys, $(1\ 1\ 1\ 0)$, $(1\ 1\ 0\ 1)$, $(1\ 0\ 1\ 1)$ and $(0\ 1\ 1\ 1)$, satisfy this requirement. In other words, there is 25% of chance that an attacker can enable the desired current. Obviously, the security from such locking is rather weak. The security improvement from increasing the number of key bits is also limited.

To allow only a unique (or very few) correct key, one idea is to have non-uniform transistor sizes among different branches. For the example in Figure 4.4, we can let $\alpha_1 = 0.5$, $\alpha_2 = 1$, $\alpha_3 = 2$ and $\alpha_4 = 4$. As such, only one key $(0\ 1\ 1\ 0)$ can satisfy $I_{out} = I^* = 3I_{REF}$. However, this approach is not good enough. In this design, even a wrong key $(q_1\ q_2\ q_3\ q_4) = (1\ 0\ 1\ 0)$ causes current of $2.5I_{REF}$, which has limited deviation from the desired value. Therefore, the related analog IC may still function but with some small performance degradation. This is in sharp contrast to the locking in digital ICs, where one bit error in the key can completely disallow the circuit to function properly.

The naïve design in Figure 4.4 has another weakness. That is, its output current increases monotonically with respect the number of 1s in the key. This monotone property allows an attacker to narrow down search space. This problem can be solved by using PMOS current branches, which effectively generate negative current. Then, the total current is no longer monotone function of the key. We will show how to overcome the other difficulties by exploring a general locking architecture using Satisfiability Modulo Theories.

4.5.2 A General Locking Architecture

We propose a general locking architecture that has a large design space for generating secure lock. This architecture consists of an $R \times N$ array of transistors and the connection between the key lines and the array.

One example of this architecture is illustrated in Figure 4.5. It has an array of $3 \times N$ control transistors. To allow further flexibility, some transistors in the array can be omitted. For example, there is no transistor in the second row and the second column. All transistors in the same branch (or column) have the same size. The sizes can be represented by an N-dimensional ratio vector $\vec{\alpha} = (\alpha_1 \; \alpha_2 \; \cdots \; \alpha_N)$, where $\alpha_j > 0$ and $\alpha_j < 0$ represent the NMOS branch and PMOS branch, respectively.

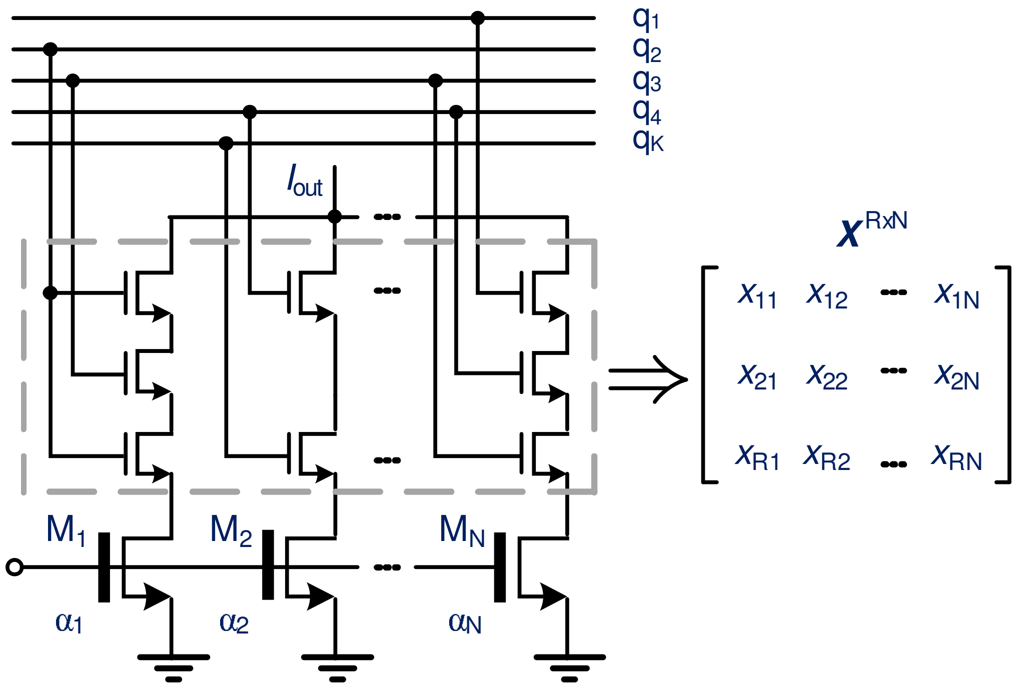


Figure 4.5: An example of the proposed locking architecture. Reprinted from [3].

The architecture has up to K key lines and the key variables are $\vec{Q} = (q_1\ q_2\ \cdots\ q_k)$, $q_k \in \{0, 1\}, k = 1, 2, \cdots, K$. Please note K can be smaller than the number of control transistors and one key line can be connected with multiple control transistors. For example, in Figure 4.5, line q_3 is connected with at least two control transistors. The connections can be specified by a control matrix $\boldsymbol{X}^{R\times N}$. Each entry $x_{ij} \in \{0, 1, \cdots K\}$ of the matrix tells which key line the transistor at row i and column j is connected with. If an entry is 0, the transistor is absent at the corresponding place. The control matrix for Figure 4.5 is given below.

$$\boldsymbol{X}^{3\times N} = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1N} \\ x_{21} & x_{22} & \dots & x_{2N} \\ x_{31} & x_{32} & \dots & x_{3N} \end{bmatrix} = \begin{bmatrix} 2 & 4 & \dots & 1 \\ 3 & 0 & \dots & 4 \\ 2 & 5 & \dots & 3 \end{bmatrix} \tag{4.8}$$

For a specific design using this architecture, the bias current is decided by

$$I_{out} = \sum_{j=1}^{N} \alpha_j \prod_{i=1}^{R} \phi(x_{ij}) \cdot I_{REF} \tag{4.9}$$

where $\phi(x_{ij})$ is the control signal at transistor of row i and column j and expressed by

$$\phi(x_{ij}) = \begin{cases} q_k & \text{if } x_{ij} = k \neq 0, \\ 1 & \text{else } x_{ij} = 0. \end{cases} \tag{4.10}$$

Please note the case $x_{ij} = 0$ is for no transistor exists at row i and column j. For given parameters R, N and K, Equation (4.9) tells that the current is decided by $\vec{\alpha}$, the control matrix and the key values.

One may notice that the inputs to transistors in the same branch must be correlated to turn on the branch and this correlation may exploited by attackers to reduce search space.

However, circuit netlist and the correlation are not accessible by recycle-based counterfeit attacks. Moreover, an user of our locking system can always choose $R = 1$ at his/her own discretion.

4.5.3 Locking Design by Satisfiability Modulo Theories

The goal for the locking design is to find a Configurable Current Mirror (CCM) design such that only one key can make the current mirror and thereby the entire analog IC system function properly and all the other keys would result in large system performance deviation or failure. In order to quantify the goal, we define parameters $\Delta \in [0, 1]$ and $\Theta \in [0, \infty]$ for specifying the lower and upper ranges of current deviation for incorrect keys, respectively. More specifically, we wish the current to be I^* only for the correct key, and to be outside the range of $[1 - \Delta, 1 + \Theta] \cdot I^*$ for the other keys. The design problem to be solved is formulated as follows.

Secure Configurable Current Mirror (CCM) Design: *For a CCM architecture with specific R, N and K, find branch size vector $\vec{\alpha}$ and control matrix $\boldsymbol{X}^{R\times N}$ such that only one key $\vec{Q}^* = (q_1\ q_2\ \cdots\ q_K)$ can make the CCM generate desired current I^* and all the other key values result in current outside of range $[1 - \Delta, 1 + \Theta] \cdot I^*$.*

This problem here is to find a feasible solution that satisfies some complicated constraints involving logic operations on equality and inequality. Mathematical programming generally handles only equality and inequality constraints. Boolean satisfiability (SAT) deals with only logic operations on Boolean variables. Therefore, it is difficult for mathematical programming or SAT to solve this problem. We propose to solve this problem using Satisfiability Modulo Theories (SMT) [56], which is a fundamental extension to SAT. SMT can describe much wider range of properties than SAT. It is usually employed as a verification tool or constraint solver, which is the case in our work. Given an SMT constraint, an SMT solver, iSAT [57] which is applied in [58], can find solutions of the variables satisfying the constraints, if those solutions ever exist.

4.5.4 Formulation of SMT Constraints

In order to facilitate the SMT solving, Equations (4.8)-(4.10) need to be transformed into the SMT format. We introduce new variables and detailed mathematical formulations as follows. First, we introduce a connection variable

$$y_{ij,k} = \begin{cases} 1 & \text{if transistor } t_{ij} \text{ is connected with key } q_k \\ 0 & \text{otherwise.} \end{cases} \tag{4.11}$$

Transistor t_{ij} is in row i and column j of the array and the values of $y_{ij,k}$ can be easily mapped to control matrix $\boldsymbol{X}^{R\times N}$. In order to ensure that each transistor is connected with no more than one key line, we enforce the following constraints

$$\sum_{\forall k} y_{ij,k} \leq 1, \ \ \forall i, j \tag{4.12}$$

Then, the on/off state of transistor t_{ij} can be described by an on/off state variable

$$p_{ij} = \sum_{\forall k} y_{ij,k} q_k + \prod_{\forall k} \bar{y}_{ij,k}, \ \ \forall i, j \tag{4.13}$$

Please note the last term on the right-hand side of the equation above is the case when no transistor exists at row i, column j, and this is equivalent to an always-on transistor.

For a key $\vec{Q}$, the bias current of Equation (4.9) can be rewritten as

$$I(\vec{Q}) = \sum_{j=1}^{N} \alpha_j \prod_{i=1}^{R} p_{ij}(\vec{Q}) \cdot I_{REF} \tag{4.14}$$

Please note $I(\vec{Q})$ is also a function of the connection variables $y_{ij,k}$ and branch sizes $\vec{\alpha}$.

Then, the key SMT constraint is described as

$$
\begin{aligned}
(I(\vec{Q}^*) = I^*) \quad &\wedge \quad (I(\vec{Q}_1) < (1-\Delta)I^* \vee I(\vec{Q}_1) > (1+\Theta)I^*) \\
&\wedge \quad (I(\vec{Q}_2) < (1-\Delta)I^* \vee I(\vec{Q}_2) > (1+\Theta)I^*) \\
&\wedge \quad (I(\vec{Q}_3) < (1-\Delta)I^* \vee I(\vec{Q}_3) > (1+\Theta)I^*) \\
&\wedge \quad \cdots
\end{aligned}
\tag{4.15}
$$

where $\vec{Q}^*$ denotes the correct key and $\vec{Q}_i, i = 1, 2, \cdots$ represent all the other keys. Please note $\vec{Q}^*$ is a variable whose value would be found by SMT solver. If $\vec{Q}^*$ is represented by $(q_1\ q_2\ \cdots\ q_K)$, then a key different from $\vec{Q}^*$ can be obtained by flipping one or multiple bits of $\vec{Q}^*$, e.g., $(\bar{q}_1\ q_2\ \cdots\ q_K)$ and $(q_1\ \bar{q}_2\ \cdots\ \bar{q}_K)$.

Since $I(\vec{Q})$ is also a function of $\vec{\alpha}$, which is not Boolean variable, this SMT constraint is difficult to be directly solved by SAT. The logic operations in the constraint are also difficult for mathematical programming to directly handle. If any feasible solution exists, the SMT solver would return values for branch sizes $\vec{\alpha}$, the connection $y_{ij,k}$ between control transistors and key lines, and the unique correct key $\vec{Q}^*$.

The values of Δ and Θ affect both security and the chance of finding feasible solution by SMT. When Δ and Θ are large, the currents from incorrect keys correspondingly have large difference from the desired current I^* and hence imply strong security. However, if they are too large, it is likely no feasible solution exists for the SMT problem. On the other hand, feasible SMT solutions can always be found for small Δ and Θ.

Although this SMT-based approach works in theory, it faces a difficulty in practice. That is, the number of clauses $I(\vec{Q}_i) < (1-\Delta)I^* \vee I(\vec{Q}_i) > (1+\Theta)I^*, i = 1, 2, \cdots$, is exponential to the number key bits. To mitigate this difficulty, we partition a long key vector into separated groups, and allocate each group to one part of analog circuit design. We solve the SMT for each group individually, and then chain the solutions from different groups together.

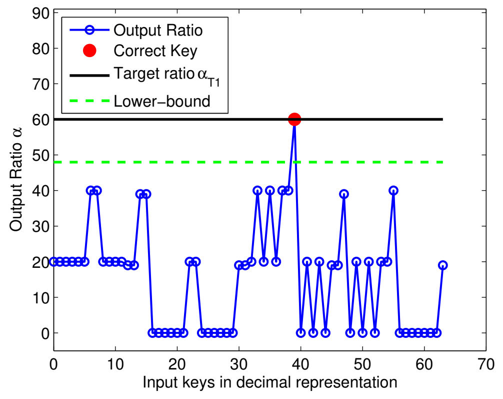
4.6 Experiment Results and Discussion

Our proposed combinational locking technique is evaluated on four different analog IC designs: (1) band pass filter, (2) quadrature oscillator, (3) LC oscillator and (4) class-D amplifier. In this section, we first describe the experiment results of these four designs, then attack to our design is discussed and area overhead result is shown at the end.

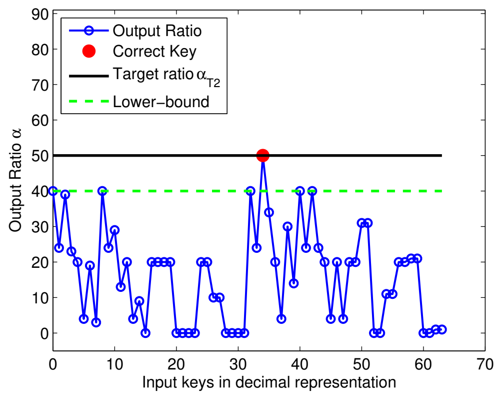
4.6.1 Experiment Result of BPF (Band-Pass Filter)

This testcase is a 4th order Gm-C BPF, which is characterized by the central frequency $f_c = 250kHz$, bandwidth $BW = 150kHz$, transition band $200kHz$, and amplitude of $0dB$. Implemented by two cascaded stages of 2nd order BPFs, this 4th order BPF has the capacitances $C_{11} = C_{12} = 78.95pF$, $f_{c_1} = 201.6kHz$, and $BW_1 = 83.6kHz$ in its first stage, while keeping $C_{21} = C_{22} = 51.34pF$, $f_{c_2} = 310kHz$, and $BW_2 = 128.5kHz$ in the second stage. According to Figure 4.3, one stage of 2nd order BPF contains four operational transconductance amplifiers, each of which needs its own bias current. In total, there are eight current mirrors in the circuit and six of them are made to be configurable. Let α_{T_i} represent the total relative branch size of current mirror i that leads to its desired bias current. The ideal sizes are $\alpha_{T_1} = 60, \alpha_{T_2} = 50, \alpha_{T_3} = 45, \alpha_{T_4} = 70, \alpha_{T_5} = 80$ and $\alpha_{T_6} = 55$.

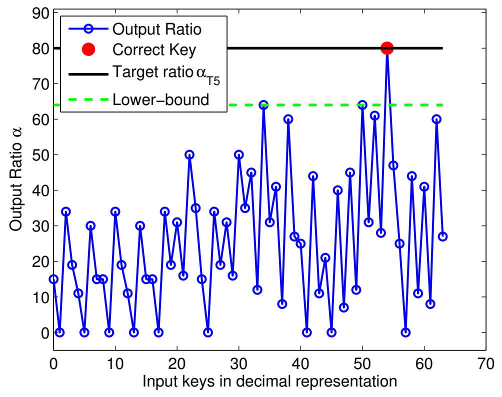
The control transistor array $\boldsymbol{X}^{R\times N}$ is shared for the 6 current mirrors. We set $R = 2$ and $N = 37$. Please note too many control transistors in a branch would degrade the current mirror performance due to the stacking effect. Hence, R must be a small number. The number of key bits is set to be $K = 33$. The 37 branches and 33 key bits are partitioned into the 6 current mirrors. The lower bound for bias current deviation is set to $\Delta = 20\%$, which is regarded as significant. Since a positive deviation, even if it is large, would rarely degrade BPF performance, it is intentionally excluded by using $\Theta = \infty$.



(a) Current mirror with $\alpha_{T_1} = 60$



(b) Current mirror with $\alpha_{T_2} = 50$



(c) Current mirror with $\alpha_{T_5} = 80$

Figure 4.6: Bias currents from different keys for three current mirrors in the 4th order BPF. Reprinted from [3].

Using our method and the SMT solver [57], the CCM-based lock is designed with a unique correct key. Figure 4.6 shows the bias currents from different keys for 3 current mirrors (the other current mirrors are not shown due to space limit). The horizontal solid lines indicate the desired current levels I^* while the horizontal dashed lines show the range of $[1 - \Delta, 1 + \Theta] \cdot I^*$. For $\Theta = \infty$, we can see only one key (in red dot) can produce current at the solid line and all the other keys result in current out of the range specified by

Δ and Θ. Moreover, the current (or size ratio) is not a monotone function with respect to key values.

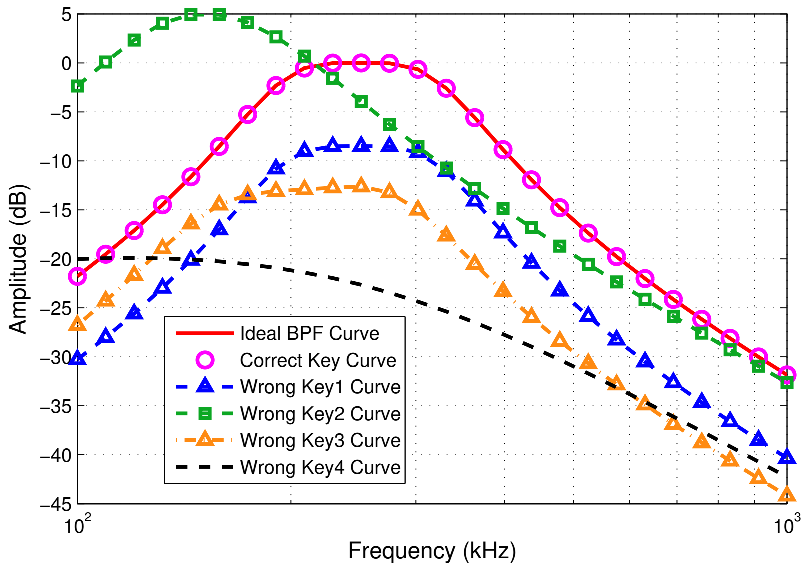
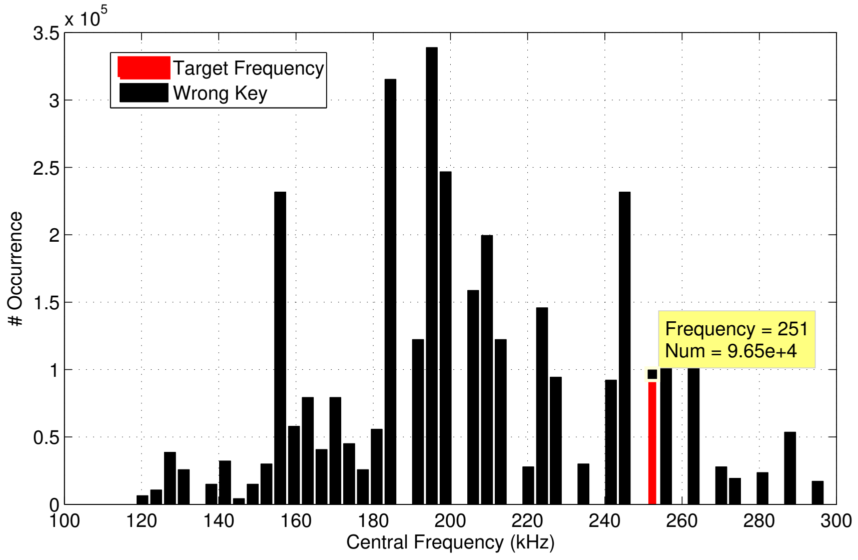


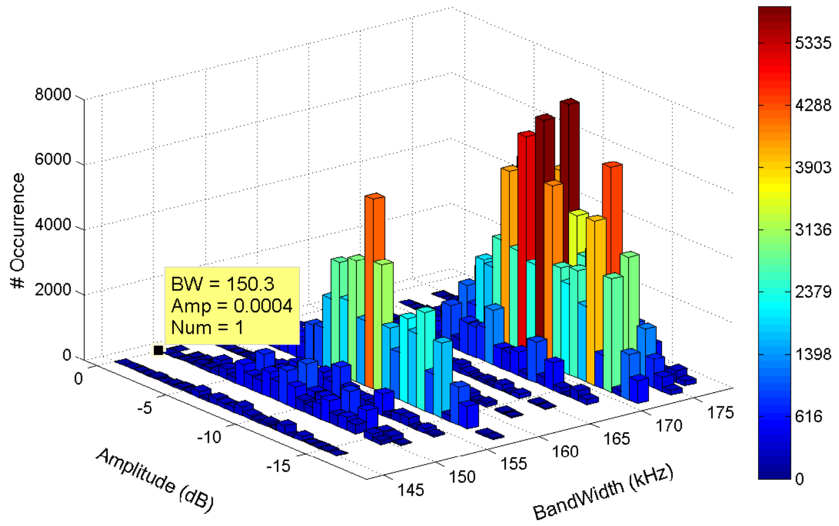
Figure 4.7: Normalized frequency responses of the 4th-order BPF with $f_c = 250kHz$ and $BW = 150kHz$, for different keys. Reprinted from [3].

Figure 4.7 presents the frequency response of the 4th order BPF for five different keys. The response from the correct key (in magenta circles) matches exactly with the ideal response (the red curve). The responses from the four wrong keys exhibit remarkable deviations from the desired one.

Statistical results for the 4th order BPF are obtained from by simulating over 8 million different keys. Among them, results from 3.4M keys with non-zero bias current are shown in Figure 4.8. The histogram in Figure 4.8 (a) indicates that there exist 96K keys with central frequency within the range from $250kHz$ to $251kHz$. However, only one key among these 96K keys satisfies the specifications of $0dB$ amplitude and $150kHz$ bandwidth, as shown in the histogram in Figure 4.8 (b).



(a) Central frequency histogram for BPF with different keys.



(b) Amplitude and bandwidth histogram for keys satisfying the central frequency f_c specification.

Figure 4.8: Only one key makes the BPF satisfying $f_c = 250kHz$, $BW = 150kHz$ and $Amplitude = 0dB$. Reprinted from [3].

4.6.2 Experiment Result of Quadrature Oscillator

Another testcase is a resistorless second-order quadrature oscillator, which is used in many communication circuits for generating sinusoidal signals. According to [59], it consists of two OTAs and two grounded capacitors C_1, C_2. Similar to the implementation of OTA in Figure 4.3, each OTA could be represented by the transconductor g_m and thus controlled by a CCM-based lock. In our test circuit, the capacitances are $C_1 = C_2 = 68pF$, transconductances are $g_{m_1} = g_{m_2} = 1mS$, the target oscillation frequency is $f_{osc} = 2.34MHz$ and the target amplitude is $1V$. The ideal sizes of the two current mirrors are $\alpha_{T_1} = 72$ and $\alpha_{T_2} = 63$, and the lower and upper deviation bound are 20% and ∞, respectively.

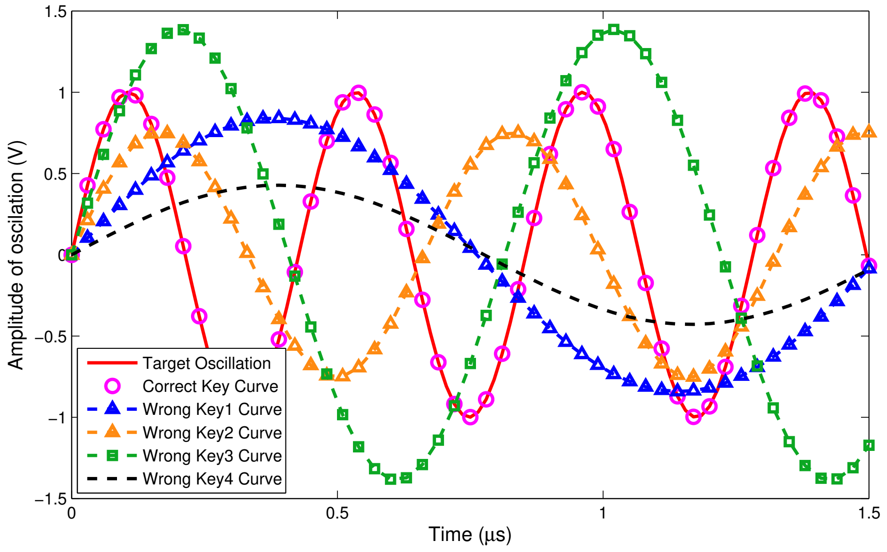


Figure 4.9: Normalized oscillation frequencies of the quadrature oscillator with $f_{osc} = 2.34MHz$, affected by different keys. Reprinted from [3].

Figure 4.9 shows output waveforms of the quadrature oscillator for five different keys.

It is clear that only the correct key can generate the sinusoid wave (in magenta circles) with the same frequency and amplitude same as the target red curve. All the other key values lead to waveforms that are quite different from the specification. Figure 4.10 is the histogram of frequency and amplitude from 32K keys with non-zero bias current while totally more than 65K keys are evaluated. It indicates that only one key satisfies both frequency and amplitude specification.

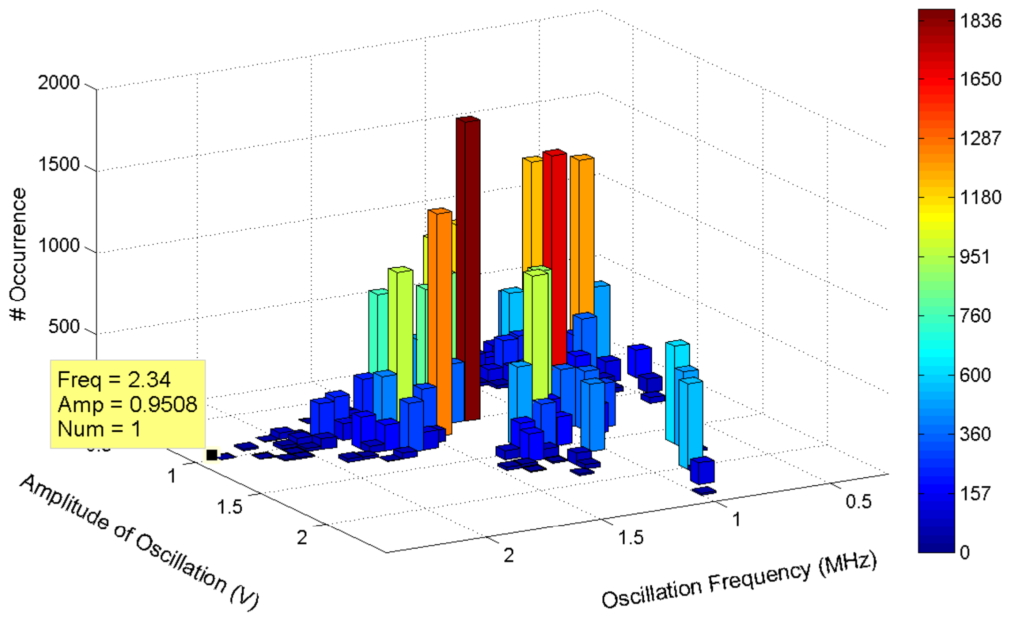


Figure 4.10: Oscillation frequency and amplitude histogram of the quadrature oscillator with $f_{osc} = 2.34MHz$, for different keys. Reprinted from [3].

4.6.3 Experiment Result of LC Oscillator

In the LC oscillator, an inductive coil L and a capacitor C form the tank circuit to store the current oscillating at the resonant frequency $f_{osc} = \frac{1}{2\pi\sqrt{LC}}$. The current mirror is applied to compensate the power loss during the oscillation and thus stabilizes the frequency. According to [60], the oscillation amplitude V_0 is related to f_{osc}^2, the reciprocal of serial resistance R_s, and the bias current I, which could be affected by the CCM. In our circuit, the inductance L is $2nH$ and the load capacitance C is $3pF$, so the target oscillation frequency is $f_{osc} = 2GHz$. The serial resistance R_s is 100Ω, and the target voltage amplitude is $2.3V$. We partition the 8 current mirrors into two groups with ideal sizes $(\alpha_{T_1}\ \alpha_{T_2}\ \alpha_{T_3}\ \alpha_{T_4}) = (80\ 75\ 52\ 64)$ and $(\alpha_{T_5}\ \alpha_{T_6}\ \alpha_{T_7}\ \alpha_{T_8}) = (50\ 90\ 100\ 67)$. The bias current deviation bound is $(\Delta, \Theta) = (20\%, \infty)$.

In Figure 4.11, all combinations of the six-bit keys for the 7th current mirror are presented in decimal format along the horizontal axis. Only the correct key (in red dot) reaches the target voltage $V_0 = 2.3V$, and all the other key values lead to degraded amplitude. Again, the amplitude does not change monotonically with respect to key values.

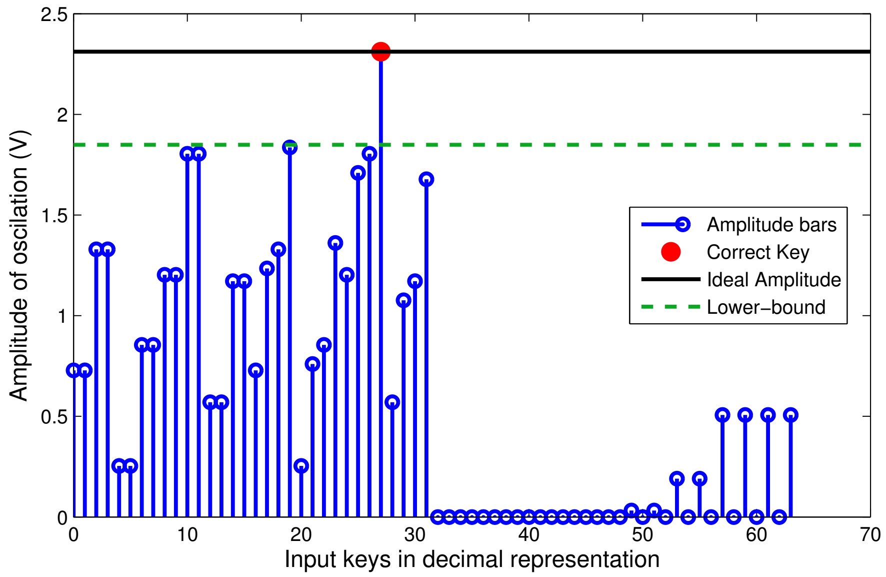


Figure 4.11: Oscillation amplitudes of the LC oscillator with $f_{osc} = 2GHz$ for all combinations of 6-bit key for CCM7. Reprinted from [3].

4.6.4 Experiment of Class-D Amplifier

Before performing signal amplification, a class-D amplifier needs to transform audio input waveform into pulse-width modulation (PWM) signal by comparing it with a triangular reference signal. Embedded in class-D amplifier [61], the triangle generator provides this reference signal by connecting the periodic charging current I_{Chg} and discharging I_{DChg} current, which are supplied by the CCMs, to a load capacitor C_{TRI} and extracting the voltage across it. In our test, the circuit is characterized by the clock frequency $f_{REF} = 2.5GHz$, load capacitance $C_{TRI} = 1pF$, the high voltage $V_H = 600mV$, and the low voltage $V_L = 400mV$. To protect this system, we make the four current mirrors configurable. Their ideal sizes are $(\alpha_{T_9}\ \alpha_{T_{10}}\ \alpha_{T_{11}}\ \alpha_{T_{12}}) = (73\ 73\ 92\ 110)$. The key has 74 bits and the other designs are similar to those of the LC oscillator in Section 4.6.3.

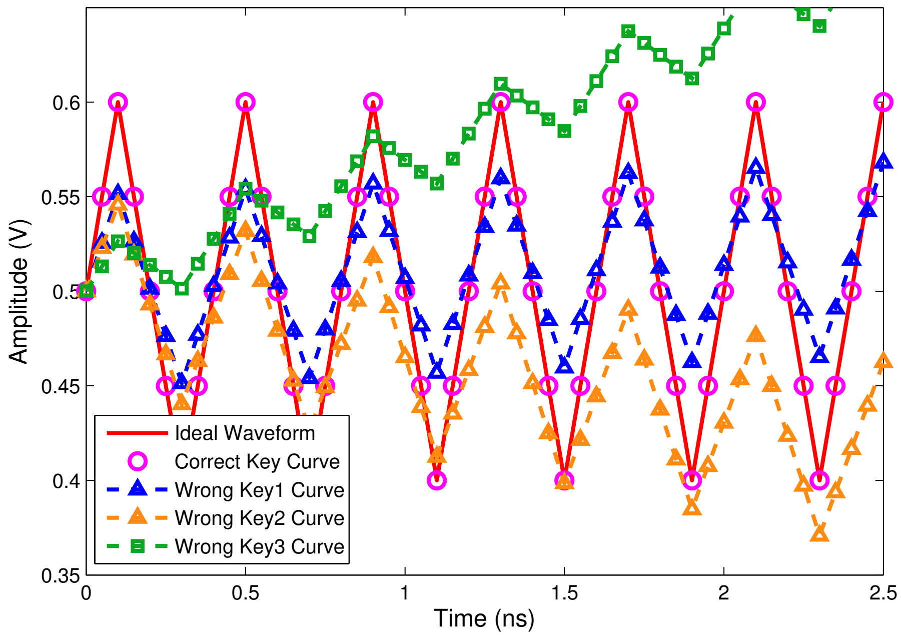


Figure 4.12: Triangular waveforms of the generator for different keys for two current mirrors, CCM9 and CCM10. Reprinted from [3].

In Figure 4.12, results for a triangular waveform generator are shown with four different keys. In the ideal waveform, since the charging current I_{Chg} and discharging current I_{DChg} are balanced, the voltage over the capacitor exhibits symmetric rising and falling slopes. Again, only the curve (in magenta circles) for the correct key has the same behavior as the specification, and the waveforms from the other keys either has too low amplitude or amplitude drifting. The amplitude error histogram is shown in Figure 4.13, where over 65K keys are evaluated and only 56K keys have non-zero bias current. Only one key induces zero error as expected.

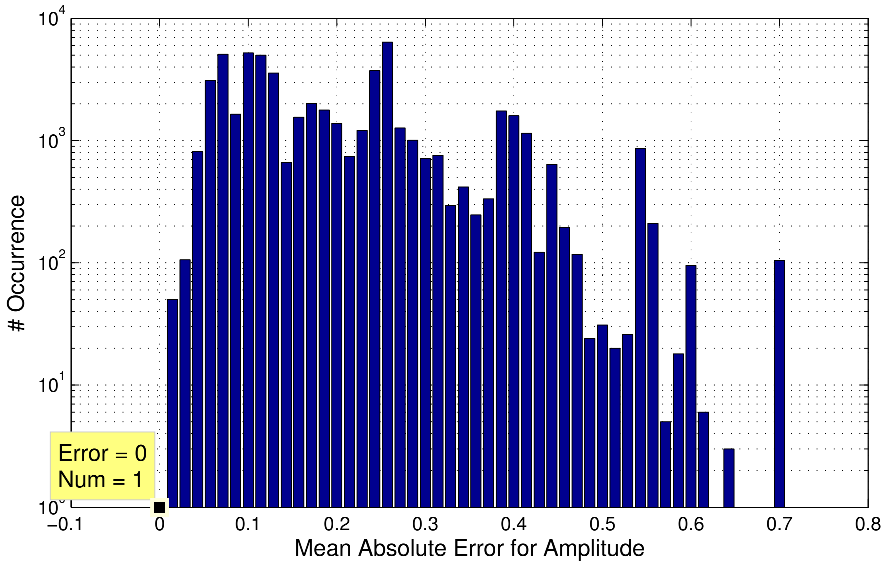


Figure 4.13: Amplitude error histogram of the triangle generator with $\Delta = 30\%$ and $\Theta = \infty$, for different keys. Reprinted from [3].

4.6.5 Security Protection Level and Attack Analysis

Our technique is mainly to defend against reverse engineering and recycle-based counterfeit. In general, the security level or attack effort for the proposed locking system is exponential to the key bitwidth. On the other hand, key bitwidth is associated with area overhead. We design key bitwidth for two different cases.

- Case 1: short key bitwidth, $16 - 32$ bits, protecting small analog IC against ad hoc attacks.

  An ad hoc attack is usually by individuals or a small team with very basic equipment. They tend to perform random or brute force attack manually. More specifically, they input each key value manually and watch circuit output using instrument like oscilloscope to judge if a key value is correct. Each of inputing key value and simple analysis of circuit output takes several seconds. Thus, it is reasonable to spend 10 seconds for evaluating one key value. If one works 12 hours a day to conduct brute force attack to a 16-bit key, it would take more than a half month to ensure success in finding the correct key value. Please note such attack can only find the correct key value of one specific chip. In order to know the correct common key for all chips of one design, the attackers need to know chip identification, which is embedded inside each chip. Ad hoc attack teams can only measure signals at chip I/O pins and hence cannot access chip identification. Spending a half month to unlock only one specific small chip is not economically worthwhile. Therefore, short key bitwidth is generally effective to defend against ad hoc reserve engineering and recycle-based counterfeit.

- Case 2: long key bitwidth, > 32 bits, protecting large IC against sophisticated attacks.

  In this case, an attack is conducted by a team with advanced equipment and related expertise. As such, they can perform the attack automatically. That is, they can program an equipment to generate and feed trial key values to the analog IC and analyze the output response. In general, analyzing analog output is much more time consuming than that for digital circuits. Consider an example that an attacker attempts to find the central frequency of BPF. Using Agilent PXA X-Series signal analyzer, the attacker needs to sweep $10MHz$ span with a $30kHz$ resolution bandwidth and it takes $73.73ms$ to analyze one output result [62]. Other analog characteristics, such as settling time and linearity, also require long analyzing time. If one output evaluation takes $10ms$, a 35-bit key would require 10 years of continuous trials to find the correct key value.

Sophisticated attackers may use optimized approaches instead of brute force method. Like in [2], they can define the error between observed output and desired output in specification, and use optimization algorithms, such as simulated annealing and genetic algorithm, to search for the key value that minimizes the error. To defeat such attack, our design makes the bias current and thereby circuit output non-monotonic and non-convex with respect to key values. For example, the amplitude mean square error (MSE) of the 4th order BPF with respect to sub-key values of two current mirrors is plotted in Figure 4.14.

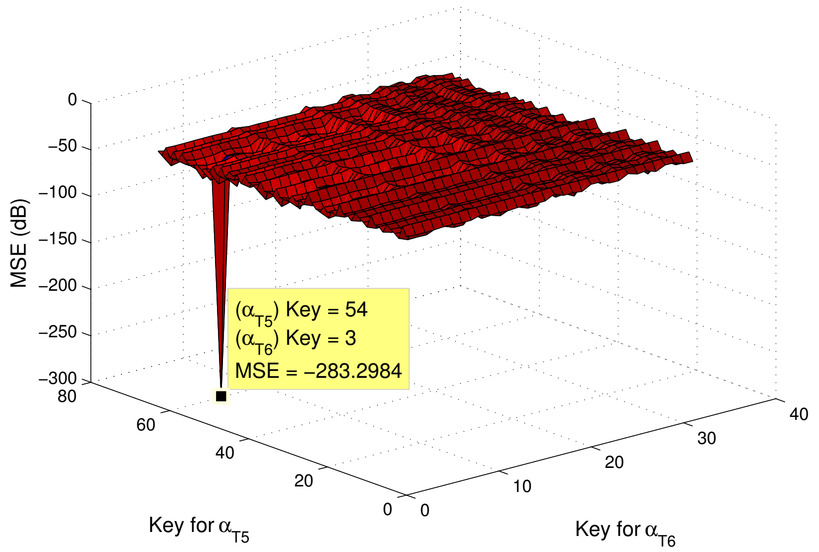


Figure 4.14: MSE curve in the second stage of the 4th order BPF satisfying $f_c = 250kHz$, $BW = 150kHz$ and $Amplitude = 0dB$. Reprinted from [3].

In this figure, only one point corresponds to the correct sub-key value while the MSEs for the other values are not monotone. As such, the required attack time for optimization algorithms is not much different from brute force search. Please note the complete case of the 4th order BPF is much more complicated than Figure 4.14 as there are 4 other current mirrors, and central frequency as well as bandwidth have to be correct in addition. For the 4th order BPF design, we use 33 bits key. To test attack in an easier case, we fix 10 bits of the key with correct values and apply simulated annealing attack to the remaining 23 bit keys. Even after 10K iterations, the simulated annealing cannot find the correct key value.

4.6.6 Area and Design Overhead

The area overhead introduced by the combinational locking design mainly includes: (1) additional current mirror branches, (2) switch transistors, (3) XOR gates, (4) access circuit, and (5) chip identification such as PUF. Item (1), (2) and (3) are part of our design and can be estimated directly. We estimate the area of access circuit assuming the use of scan chain. The chip identification area overhead is based on the PUF design in [63] with $0.18\mu m$ technology. The overall area overhead accounting for these 5 items is summarized in Table 4.1. The 4th order BPF is designed with two cascaded 2nd order BPFs [54].

Table 4.1: Area overhead and key bitwidth. Reprinted from [3].

Circuit	Original Area (μm^2)	Area Overhead (μm^2)	Percent	Tech. Node (μm)	Key bits
Quadrature Osc. [59]	68,210	3,380	4.96%	0.18	16
2nd order BPF [54]	79,202	4,807	6.07%	0.18	22
4th order BPF	131,126	7,171	5.47%	0.18	33
LC Oscillator [60]	95,000	6,304	6.64%	0.13	46
Class-D Amp. [61]	1,500,000	11,558	0.77%	0.13	74

Table 4.2 lists the computing runtime for iSAT to solve the CCM design. In this experiment, the SMT based CCM design is implemented in C/C++ and run on a Linux server with AMD-V 2.3GHz processor. To generate a CCM design and 33-bit key, the SMT solver [57] takes about 4 minutes runtime and thus the design overhead is small.

Table 4.2: Runtime for solving the SMT for different bitwidths. Reprinted from [3].

	16 bits	22 bits	33 bits	46 bits	74 bits
CPU (s)	31.86	96.89	214.80	327.14	462.75

5. SUMMARY AND CONCLUSIONS

In this dissertation, we discuss three circuit adaptivity applications to overcome the challenges coming from the variations and pirated devices. Firstly, a variation aware technique is proposed to achieve the joint gate implementation selection and adaptivity assignment of adaptive body bias circuit. The novelty of solving the redundant counting on the reconvergence path is included without increasing the time complexity of our algorithm. Experiments show that this technique leads to the substantial reduce of adaptivity overhead. In the second work, we implement an on-chip self validation platform for the post calibration of diverse analog circuits. The combination of Simulated Annealing and Sensitivity algorithm provides a balance between exploring the whole solution space and exploiting a local solution area. The effectiveness of our work is demonstrated by chip measurement and simulation. A further extension is to implement all the functionality through a reconfigurable chip which provides comprehensive flexibility to the cost function and optimization engine with a feasible cost of area consumption. In the last application, we proposed a novel idea to design a digital lock with the current mirror which is prevailing in analog circuits. The difficulty of finding the unique key as well as the non-Boolean size of the analog component is solved by the Satisfiability Modulo Theories. The maximal area overhead of this design is around 7%, yet it could guarantee a more than 10 years' security level to beat the sophisticated hackers with a negligible design time.

REFERENCES

[1] H. He, J. Wang, and J. Hu, "Collaborative Gate Implementation Selection and Adaptivity Assignment for Robust Combinational Circuits," in *International Symposium on Low Power Electronics and Design*, pp. 122–127, 2015.

[2] J. Wang, C. Shi, E. Sanchez-Sinencio, and J. Hu, "Built-In Self Optimization for Variation Resilience of Analog Filters," in *Computer Society Annual Symposium on VLSI*, pp. 656–661, 2015.

[3] J. Wang, C. Shi, A. Sanabria-Borbon, E. Sánchez-Sinencio, and J. Hu, "Thwarting Analog IC Piracy via Combinational Locking," in *IEEE International Test Conference*, 2017.

[4] S. Ozdemir, D. Sinha, G. Memik, J. Adams, and H. Zhou, "Yield-Aware Cache Architectures," in *Proceedings of the International Symposium on Microarchitecture*, (Washington, DC, USA), pp. 15–25, IEEE Computer Society, 2006.

[5] B. Lovejoy, "Supply-Chain Report Claims Low Yield Rates of TSMC's 10nm Chips Could Delay 2017 iPad Production," *https://9to5mac.com/2016/12/23/tsmc-10nm-a10x-chips-2017-ipad-delay-claim/*, 2016.

[6] E. Maricau and G. Gielen, "Reliability Simulation of Analog ICs under Time-Varying Stress in Nanoscale CMOS," *IEEE Workshop on Design for Reliability and Variability*, 2008.

[7] IHS Technology Press Release, "Top 5 Most Counterfeited Parts Represent A $169 Billion Potential Challenge for Global Semiconductor Market," *http://technology.ihs.com/405654/top-*, 2012.

[8] J. W. Tschanz, J. T. Kao, S. G. Narendra, R. Nair, D. A. Antoniadis, A. P. Chandrakasan, and V. De, "Adaptive Body Bias for Reducing Impacts of Die-to-Die and within-Die Parameter Variations on Microprocessor Frequency and Leakage," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 11, pp. 1396–1402, 2002.

[9] X. Liang, G.-Y. Wei, and D. Brooks, "Revival: A Variation-Tolerant Architecture Using Voltage Interpolation and Variable Latency," *IEEE Micro*, vol. 29, no. 1, pp. 127–138, 2009.

[10] K.-N. Shim and J. Hu, "Boostable Repeater Design for Variation Resilience in VLSI Interconnects," *IEEE Transactions on VLSI Systems*, vol. 21, no. 9, pp. 1619–1631, 2013.

[11] M. Agarwal, B. C. Paul, M. Zhang, and S. Mitra, "Circuit Failure Prediction and Its Application to Transistor Aging," in *Proceedings of the IEEE VLSI Test Symposium*, pp. 277–286, 2007.

[12] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "Mathematically Assisted Adaptive Body Bias (ABB) for Temperature Compensation in Gigascale LSI Systems," in *Proceedings of Asia and South Pacific Design Automation Conference*, pp. 559–564, 2006.

[13] K. M. Brownell, A. D. Khan, G.-Y. Wei, and D. Brooks, "Automating Design of Voltage Interpolation to Address Process Variations," *IEEE Transactions on VLSI Systems*, vol. 19, no. 3, pp. 383–396, 2011.

[14] K.-N. Shim, J. Hu, and J. Silva-Martinez, "Dual-Level Adaptive Supply Voltage System for Variation Resilience," *IEEE Transactions on VLSI Systems*, vol. 21, no. 6, pp. 1041–1052, 2013.

[15] C. R. Lefurgy, A. J. Drake, M. S. Floyd, M. S. Allen-Ware, B. Brock, J. A. Tierno, J. B. Carter, and R. W. Berry, "Active Guardband Management in Power7+ to Save Energy and Maintain Reliability," *IEEE Micro*, vol. 33, no. 4, pp. 35–45, 2013.

[16] A. Agarwal, D. Blaauw, and V. Zolotov, "Statistical Timing Analysis for Intra-Die Process Variations with Spatial Correlations," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 900–907, 2003.

[17] M. Mani, A. Singh, and M. Orshansky, "Joint Design-Time and Post-Silicon Minimization of Parametric Yield Loss Using Adjustable Robust Optimization," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 19–26, 2006.

[18] S. H. Kulkarni, D. M. Sylvester, and D. T. Blaauw, "Design-Time Optimization of Post-Silicon Tuned Circuits Using Adaptive Body Bias," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 3, pp. 481–494, 2008.

[19] C. Zhuo, D. Blaauw, and D. Sylvester, "Variation-Aware Gate Sizing and Clustering for Post-Silicon Optimized Circuits," in *Proceedings of the International Symposium on Low Power Electronics and Design*, pp. 105–110, 2008.

[20] Y. Kunitake, T. Sato, and H. Yasuura, "A Replacement Strategy for Canary Flip-Flops," in *Proceedings of the IEEE Pacific Rim International Symposium on Dependable Computing*, pp. 227–228, 2010.

[21] M. Guthaus, N. Venkateswaran, C. Visweswariah, and V. Zolotov, "Gate Sizing Using Incremental Parameterized Statistical Timing Analysis," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 1029–1036, 2005.

[22] D. Sinha, N. V. Shenoy, and H. Zhou, “Statistical Timing Yield Optimization by Gate Sizing,” *IEEE Transactions on VLSI Systems*, vol. 14, no. 10, pp. 1140–1146, 2006.

[23] C. P. Chen, C. C.-N. Chu, and D. F. Wong, “Fast and Exact Simultaneous Gate and Wire Sizing by Lagrangian Relaxation,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 7, pp. 1014–1025, 1999.

[24] Y. Liu and J. Hu, “A New Algorithm for Simultaneous Gate Sizing and Threshold Voltage Assignment,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 2, pp. 223–234, 2010.

[25] H. Chang and S. S. Sapatnekar, “Statistical Timing Analysis under Spatial Correlations,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 9, pp. 1467–1482, 2005.

[26] K. Chaudhary and M. Pedram, “A Near Optimal Algorithm for Technology Mapping Minimizing Area under Delay Constraints,” in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 492–498, 1992.

[27] M. Ozdal, C. Amin, A. Ayupov, S. Burns, G. Wilke, and C. Zhuo, “An Improved Benchmark Suite for the ISPD-2013 Discrete Cell Sizing Contest,” in *Proceedings of the ACM International Symposium on Physical Design*, pp. 168–170, 2013.

[28] B. Liu, G. Gielen, and F. V. Fernández, *Automated Design of Analog and High-frequency Circuits*. Springer, 2013.

[29] P. D. Wit and G. Gielen, “Degradation-Resilient Design of A Self-Healing xDSL Line Driver in 90 nm CMOS,” *IEEE Journal of Solid-State Circuits*, vol. 47, no. 7, pp. 1757–1767, 2012.

[30] Y. Xu, K. Hsiung, and X. Li, “OPERA: OPtimization with Ellipsoidal Uncertainty for Robust Analog IC Design,” in *Proceedings of the ACM/IEEE Design Automation*

Conference, pp. 632–637, 2005.

[31] T. McConaghy and G. G. E. Gielen, “Globally Reliable Variation-Aware Sizing of Analog Integrated Circuits via Response Surfaces and Structural Homotopy,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 11, pp. 1627–1640, 2009.

[32] X. Li, B. Taylor, Y. Chien, and L. Pileggi, “Adaptive Post-Silicon Tuning for Analog Circuits: Concept, Analysis and Optimization,” in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 450–457, 2007.

[33] D. Han, B. S. Kim, and A. Chatterjee, “DSP-Driven Self-Tuning of RF Circuits for Process-Induced Performance Variability,” *IEEE Transactions on VLSI Systems*, vol. 18, no. 2, pp. 305–314, 2010.

[34] S. Ray and B. Song, “A 13-b Linear, 40-MS/s Pipelined ADC with Self-Configured Capacitor Matching,” *IEEE Journal of Solid-State Circuits*, vol. 42, no. 3, pp. 463–474, 2007.

[35] D. Maliuk and Y. Makris, “On-Chip Intelligence: A Pathway to Self-Testable, Tunable, and Trusted Analog/RF ICs,” in *Proceedings of the Midwest Symposium on Circuits and Systems*, pp. 1077–1080, 2014.

[36] A. Valdes-Garcia, F.-L. Hussien, J. Silva-Martinez, and E. Sanchez-Sinencio, “An Integrated Frequency Response Characterization System with A Digital Interface for Analog Testing,” *IEEE Journal of Solid-State Circuits*, vol. 41, no. 10, pp. 2301–2313, 2006.

[37] L. Vachhani, K. Sridharan, and P. Meher, “Efficient CORDIC Algorithms and Architectures for Low Area and High Throughput Implementation,” *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 56, no. 1, pp. 61–65, 2009.

[38] J. Haddock and J. Mittenthal, "Simulation Optimization Using Simulated Annealing," *Computers and Industrial Engineering*, vol. 22, no. 4, pp. 387–395, 1992.

[39] SEMI, "Innovation Is at Risk: Losses of up to $4 Billion Annually due to IP Infringement," *http://semi.org/en/innovation-risk-losses-4-billion-annually-due-ip-infringement*, 2008.

[40] U. Guin, K. Huang, D. DiMase, J. M. Carulli, M. Tehranipoor, and Y. Makris, "Counterfeit Integrated Circuits: A Rising Threat in the Global Semiconductor Supply Chain," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1207–1228, 2014.

[41] Y. Bi, J. Yuan, and Y. Jin, "Beyond the Interconnections: Split Manufacturing in RF Designs," *Electronics*, vol. 4, no. 3, pp. 541–564, 2015.

[42] D. H. K. Hoe, J. Rajendran, and R. Karri, "Towards Secure Analog Designs: A Secure Sense Amplifier Using Memristors," in *Computer Society Annual Symposium on VLSI*, pp. 516–521, 2014.

[43] B. Schneier, "Evil Maid Attacks on Encrypted Hard Drives," *https://www.schneier.com/blog/archives/2009/10/evil_maid_attac.html*, 2009.

[44] V. V. Rao and I. Savidis, "Protecting Analog Circuits with Parameter Biasing Obfuscation," in *Latin American Test Symposium*, pp. 1–6, 2017.

[45] Y. M. Alkabani and F. Koushanfar, "Active Hardware Metering for Intellectual Property Protection and Security," in *USENIX Security Symposium*, pp. 1–16, 2007.

[46] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending Piracy of Integrated Circuits," in *Design, Automation and Test in Europe*, pp. 1069–1074, 2008.

[47] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security Analysis of Logic Obfuscation," in *Design Automation Conference*, pp. 83–89, 2012.

[48] B. Liu and B. Wang, “Embedded Reconfigurable Logic for ASIC Design Obfuscation against Supply Chain Attacks,” in *Design, Automation Test in Europe Conference Exhibition*, pp. 1–6, 2014.

[49] P. Subramanyan, S. Ray, and S. Malik, “Evaluating the Security of Logic Encryption Algorithms,” in *IEEE International Symposium on Hardware Oriented Security and Trust*, pp. 137–143, 2015.

[50] Y. Su, J. Holleman, and B. Otis, “A 1.6pJ/bit 96% Stable Chip-ID Generating Circuit Using Process Variations,” in *International Solid-State Circuits Conference. Digest of Technical Papers*, pp. 406–611, 2007.

[51] C. Herder, M. D. Yu, F. Koushanfar, and S. Devadas, “Physical Unclonable Functions and Applications: A Tutorial,” *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1126–1141, 2014.

[52] R. Torrance and D. James, *The State-of-the-Art in IC Reverse Engineering*, pp. 363–381. Berlin, Heidelberg: Springer, 2009.

[53] B. Razavi, *Design of Analog CMOS Integrated Circuits*, pp. 16–18. New York, NY: McGraw-Hill, 2001.

[54] A. C. Sanabria-Borbon and E. Sanchez-Sinencio, “Efficient Use of Gain-bandwidth Product in Active Filters: Gm-C and Active-R Alternatives,” in *Latin American Symposium on Circuits Systems*, pp. 1–4, 2017.

[55] M. Yasin, J. J. Rajendran, O. Sinanoglu, and R. Karri, “On Improving the Security of Logic Locking,” *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 9, pp. 1411–1424, 2016.

[56] C. W. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli, “Satisfiability Modulo Theories,” *Handbook of Satisfiability*, vol. 185, pp. 825–885, 2009.

[57] K. Scheibler, "ISAT: Tight Integration of Satisability & Constraint Solving," *http://projects.informatik.uni-freiburg.de/projects/isat3/*, 2014.

[58] H. Lin and P. Li, "Parallel Hierarchical Reachability Analysis for Analog Verification," in *Design Automation Conference*, pp. 1–6, 2014.

[59] R. Sotner, J. Jerabek, N. Herencsar, K. Vrba, and T. Dostal, "Features of Multi-loop Structures with OTAs and Adjustable Current Amplifier for Second-order Multiphase/Quadrature Oscillators," *International Journal of Electronics and Communications*, vol. 69, no. 5, pp. 814 – 822, 2015.

[60] Z. Zahir and G. Banerjee, "A Multi-Tap Inductor Based 2.0-4.1 GHz Wideband LC-Oscillator," in *Asia Pacific Conference on Circuits and Systems*, pp. 330–333, 2016.

[61] K. E. Khadiri and H. Qjidaa, "Integrated 60-V Class-D Power Output Stage with 95% Efficiency in A 0.13 um SOI BCD Process," in *Intelligent Systems and Computer Vision*, pp. 1–6, 2015.

[62] Agilent Technologies, Inc., "Using Fast-Sweep Techniques to Accelerate Spur Searches," *http://cp.literature.agilent.com/litweb/pdf/5991-3739EN.pdf*, 2014.

[63] S. Lin, Y. Cao, X. Zhao, X. Wang, and X. Pan, "A Compact Ultra-Low Power Physical Unclonable Function Based on Time-Domain Current Difference Measurement," in *International Symposium on Circuits and Systems*, pp. 277–280, 2016.