



Technische Universität München  
TUM School of Computation, Information and Technology

# Performance Prediction of Microcontrollers based upon integrated Technology Monitors

**Tobias Kilian**

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung eines

**Doktors der Ingenieurwissenschaften (Dr.-Ing.)**

genehmigten Dissertation.

**Vorsitz:**

Prof. Dr.-Ing. Georg Sigl

**Prüfende der Dissertation:**

1. Prof. Dr.-Ing. Ulf Schlichtmann
2. Prof. Matteo Sonza Reorda, Ph.D.

Die Dissertation wurde am 10.04.2024 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 24.01.2025 angenommen.



# Abstract

Microcontrollers (MCUs) are used for a broad range of applications. The testing of such MCUs is an essential topic for MCU manufacturers, particularly in safety-critical MCUs. Such MCUs are used in the automotive industry to operate braking, steering, and airbag systems. A critical test of the MCU testing process is the performance screening in which the maximum clock frequency of the MCU is determined under worst-case conditions. Indirect monitors, like Ring Oscillators (ROs), are used for this performance screening. This work presents the functional path ROs as an indirect monitoring structure used for performance screening. A holistic overview of the functional path ROs from the pre-silicon to the post-silicon is presented.

The implementation of such ROs and the associated advantages in terms of area consumption, leakage, and routing are presented. In addition, advanced implementation methodologies are proposed to save further routing resources. A considerable framework is suggested to select promising functional paths based on sensitivity analysis, and the entire implementation selection flow is validated. In the post-silicon phase, a statistically significant amount of MCUs that contain the functional path ROs are manufactured and tested. The measured functional path RO frequencies are correlated with the MCU performance using machine learning approaches and compared with traditional RO structures. The performance screening uses the implemented RO types, and the results yield is evaluated.

## Acknowledgments

This thesis results from my work as a doctoral candidate at Infineon Technologies AG, Neubiberg, Germany.

First of all, I thank Prof. Dr.-Ing. Ulf Schlichtmann for offering me the generous opportunity for this research work. He guided me through the work with his passion, constructive advice, and valuable discussions. I truthfully appreciate his genius academic support and feedback on publishing my papers.

I also would like to thank the co-examiner, Prof. Matteo Sonza Reorda, Ph.D., for his interest in my thesis and his time.

Also, I would like to thank my colleagues at Infineon Technologies AG. Thank you to my managers, Carsten Doerrhoefer and Dr. rer. nat. Jochen Roeder, for admitting me to the opportunity in your team. A special thanks to my supervisors, Dr.-Ing. Martin Huch and Dr.-Ing. Daniel Tille. Martin, thank you for the inspiring discussions and technical guidance through this work. Daniel, thank you for the academic guidance in the test community, the constructive discussions, and the distinguished feedback.

In addition, thank you to the teams that supported and helped me make this research work successful from the concept to the first silicon and beyond, especially the DFX, chip integration, test, and product engineering teams. With your support, this work is in the state as it is now. I want to thank Dr.-Ing. Juergen Alt for the review of this thesis and the discussions.

I would also like to thank the students who contributed to this work, especially Markus Hanel. Also, I am deeply thankful for the great collaboration and research conducted with the Politecnico di Torino (Turin, Italy), thanks to Nicolò Bellarmino, Prof. Riccardo Cantoro, and Prof. Giovanni Squillero.

Finally, I would like to thank my family and friends for their support and understanding of my lack of time and presence over the recent years. Mainly, I sincerely thank my wife, Kerstin, for her continuous support and motivation.

# Contents

<b>Abstract</b>	<b>i</b>
<b>Acknowledgments</b>	<b>ii</b>
<b>1. Introduction</b>	<b>1</b>
1.1. Motivation and Problem Statement . . . . .	1
1.2. Related Work . . . . .	3
1.3. Contribution of this Work . . . . .	6
1.4. Outline of this Dissertation . . . . .	7
<b>2. Background</b>	<b>8</b>
2.1. Digital MCU Development Flow . . . . .	8
2.2. Timing and Performance of digital Circuits . . . . .	10
2.3. PVT Variations . . . . .	12
2.4. Testing of MCUs . . . . .	19
2.5. Performance testing . . . . .	24
2.6. Machine Learning in IC Testing . . . . .	27
2.7. Automotive Quality . . . . .	29
<b>I. PRE-Silicon</b>	<b>31</b>
<b>3. Functional Path Ring Oscillators</b>	<b>32</b>
3.1. Basic Concept . . . . .	32
3.2. Advanced Implementation Concepts . . . . .	35
3.3. Self Enabling . . . . .	38
3.4. Control Infrastructure . . . . .	43
3.5. Implementation Flow . . . . .	46
3.6. Results . . . . .	49
<b>4. Path Selection Methodology</b>	<b>61</b>
4.1. Selection Flow . . . . .	61

4.2. Results . . . . .	64
<b>5. PRE-Silicon Verification and Validation</b>	<b>68</b>
5.1. PRE-Silicon Verification of the Functional Path ROs . . . . .	68
5.2. SI Bring-up Preparation . . . . .	71
5.3. PRE- Silicon Validation and Improvement . . . . .	75
<b>6. PRE-SI Summary</b>	<b>83</b>
<b>II. POST-Silicon</b>	<b>85</b>
<b>7. Measurement and Validation of the Functional Path RO</b>	<b>87</b>
7.1. Functional Path RO Measurement Results . . . . .	87
<b>8. Performance Screening Using Functional Path RO</b>	<b>91</b>
8.1. Method and Data Set for the Performance Screening . . . . .	91
8.2. Preprocessing of the Data Set . . . . .	97
8.3. Performance Screening Flow . . . . .	97
8.4. Deploy the Performance Screening . . . . .	104
8.5. Results of the Performance Screening . . . . .	104
<b>9. POST-SI Summary</b>	<b>115</b>
<b>10. Conclusion</b>	<b>116</b>
10.1. Summary of Key Contributions . . . . .	116
10.2. Obstacles in an Industrial Context . . . . .	117
10.3. Future Work . . . . .	118
<b>List of Figures</b>	<b>119</b>
<b>List of Tables</b>	<b>122</b>
<b>Acronyms</b>	<b>123</b>
<b>Bibliography</b>	<b>126</b>

# 1. Introduction

Semiconductor electronics are an essential element in today's digitized world. This has become common knowledge in 2022 since the U.S. and European governments have released billions of dollars in subsidies to strengthen and foster the semiconductor ecosystem in the U.S. [1] and European [2] chips acts. Essential semiconductor devices are integrated circuits (ICs). ICs are small semiconductor devices that are fundamental components in nearly everything in life and the economy.

A subgroup of ICs are **microcontrollers (MCUs)**. An **MCU** is an IC that combines different systems on one chip; also, the term **system-on-chip (SoC)** is used. An **MCU** combines the processor, memory, input/output (I/O) peripherals, and communication interfaces on a single chip. The field of applications is enormous. It ranges from tiny household and commercial applications to high-reliable technologies, like space applications and automotive technology. Depending on the application, **MCUs** must meet different quality, power consumption, performance, and cost requirements. **MCUs** used in the automotive industry have special needs regarding quality [3] and reliability [4]. This work focuses on **MCUs** for automotive applications; however, the method can be used for every **MCU**.

## 1.1. Motivation and Problem Statement

More than 100 of these **MCUs** are incorporated in modern vehicles [4] and are used in safety-critical steering and braking systems as well as in advanced driver assistance systems (ADAS). The **MCUs** must work under extreme temperature conditions from  $-40^{\circ}\text{C}$  up to  $150^{\circ}\text{C}$ . They are exposed to vibrations and harsh conditions. Moreover, the automotive **MCU** must ensure to work for up to 15 years [5]. A not correctly working **MCU** can cause tremendous damage and risk human lives. Therefore, much effort is invested in testing and validating automotive **MCUs** to satisfy quality requirements. The **MCU** test is essential for finding defects provoked by the manufacturing process, ensuring proper functionality, and meeting the required specifications.

The major challenge in testing automotive **MCUs** is to achieve *zero-defect quality* [6]. This means only **MCUs**, which are error-free, will be delivered to the customer. However, zero-defect quality is not easy to achieve for large and complex automotive **MCUs**. Therefore,

automotive **MCU** manufacturers invest much effort in testing all devices to deliver zero-defect quality.

There are many test stages along the manufacturing process, and modern **MCUs** have an extensive DFT infrastructure on board to support and enable the various tests. An essential test of the **MCUs** test procedure is performance screening. The performance of an **MCU** is the maximum achievable clock frequency at which the device can execute all use cases under worst-case conditions, denoted as  $F_{MAX}$ . Performance screening describes the test in which  $F_{MAX}$  is determined. Therefore, each device shipped to the customer must pass the performance screening by achieving the performance specified in the data sheet. The timing of the design in the **MCU** determines the  $F_{MAX}$ .

Since the technology node in **MCU** is shrinking, the timing variability becomes more sensitive to manufacturing, environment, and aging. This timing variability is called PVTA (process, voltage, temperature, aging) variation. The aging component is neglected in this work since it is an elaborate topic that is out of scope. Thus, **Process-Voltage-Temperature (PVT)** variability is addressed in this work. The timing of the **MCU** and  $F_{MAX}$  should be satisfied regardless of the PVT variability. This makes the performance screening even more challenging since the  $F_{MAX}$  should be guaranteed under worst-case conditions.

Figure 1.1 provides a simplified illustration of the performance in **MCU** manufacturing. The performance of manufactured **MCUs** can be approximated as Gaussian distribution colored in blue [7]. Thus, many of the devices are well centered around the mean. However, there are also devices with lower  $F_{MAX}$ , the so-called slow tail on the left side of the distribution, and devices with higher  $F_{MAX}$  on the right side, called the fast tail of the distribution. The reason for this distribution lies in the semiconductor manufacturing process itself. The aim is for many components within this Gaussian distribution to fulfill the requirement  $F_{MAX}$ , which in turn depends on the design border. This design border is the value that is required as a design specification. The exact position of the design border depends on many technical as well as financial factors.

The brute force approach would shift the design border (red) far to the left sides of the distribution - which means all **MCUs** are indeed fast. Unfortunately, that does not work since this would result in an **MCU** design that is not competitive due to area, design effort, and economic reasons. The design border in realistic scenarios is much closer to the distribution and can even cut into this distribution. The performance screening targets testing the performance near the design border.

However, the performance cannot be measured directly, and the entire multidimensional PVT space must be considered. In recent years, various methods for determining performance have been proposed. The methods range from structural tests to system-level-tests and indirect monitoring structures [8, 9, 10, 11, 12, 13, 14, 15, 16]. Due to the fact that a hundred

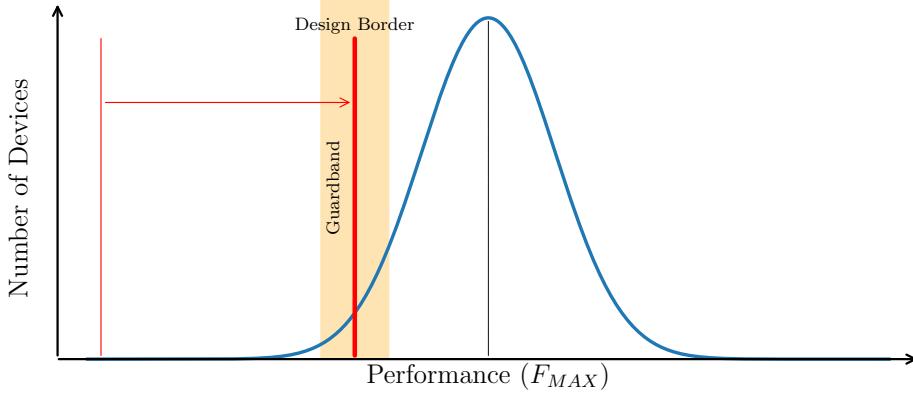


Figure 1.1.: Process window after tightening the limits.

percent accurate determination of the performance is not possible, it is an attempt to predict the performance as accurately as possible. What remains is an inaccuracy that must be priced into the performance screening. In order to safeguard the inaccuracy of the screening, a so-called Guardband (colored orange in Figure 1.1) is used. This margin is added to the performance screening to meet the high-quality requirements - better performance screening methods result in higher screening accuracy, and thus less Guardband is required.

In order to determine the performance for the performance screening, indirect monitors are often used. Such indirect monitor structures are [ring oscillators \(ROs\)](#). The oscillation frequency of such ROs is measured, and this measured frequency correlates with  $F_{MAX}$ . However, the [RO](#) structures usually have a significant area overhead, but in turn, have a high resolution and are easy and fast to measure [17]. Furthermore, the quality and accuracy of performance screening depend significantly on the design of such [RO](#) structures [18, 19, 20].

The *functional path ROs* [21] are a subtype of [ROs](#) that promise advantages in terms of area and accuracy since they use functional paths which are already on the design. However, the functional path ROs need further research on how to implement and control, which paths to select, and how to establish the functional path RO for performance screening on silicon in a large automotive [MCU](#). This work is an important contribution to these endeavors.

## 1.2. Related Work

Determining the performance of modern [SoCs](#) is challenging due to their complexity - especially for large [MCUs](#). Three significant approaches in  $F_{MAX}$  testing are widely used and visualized in Figure 1.2.

One way to test performance is to use structural at-speed test patterns, such as *Transition Delay Fault* [22] patterns or *Path Delay Fault* [23] patterns [8, 9]. However, such at-speed

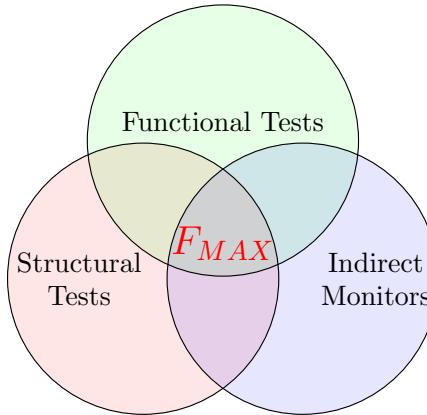


Figure 1.2.: Test approaches for  $F_{MAX}$  testing.

patterns can only provide a pass or fail categorization of the chip. Performance testing requires an accurate determination of the numerical performance value.

A straightforward approach is determining the **MCU**'s performance with a *Functional Test* on the **automatic test equipment (ATE)** [10]. Small critical code blocks as part of the **ATE** test program are executed, and the clock frequency is increased until the point of failure [11]. However, **ATE** systems usually have limited power integrity, making the exact performance prediction inaccurate [24]. As a result, *System-Level Test (SLT)* is gaining increasing attention [12, 13]. In contrast to the functional test on the **ATE**, the SLT is performed on an application-like board with software close to the device's final application. However, several thousand applications are conceivable. Therefore, all possible applications must be tested under different voltage and temperature conditions to determine the maximum performance of every device [24]- which is not an option due to the limited test budget. Thus, SLT is expensive in terms of test time and handling [25].

The third approach is to use indirect monitoring structures. There are many types of such indirect methods. However, they all used indirect metrics that allowed them to conclude performance [26, 27, 28, 29, 30, 31, 21].

*Razor FF* [26, 27] and *In-situ* slack monitors [28] are some types of indirect monitoring structures. Razor FFs are additional FFs that observe specific paths on the design and check if the timing is met. Also, in-situ slack monitors observe the remaining slack of a specific path and give an alarm if the slack margin disappears. Both approaches can be used for performance determination in small circuits where some known paths are timing critical.

However, for large **MCUs** in state-of-the-art CMOS technology, the timing of a design gets more complex [32], and it is challenging to define discrete time-critical paths. It may be that paths, which seem non-critical in the pre-silicon phase may become critical in the post-silicon phase due to process variations in manufacturing [33]. The degradation mechanism also

affects the performance [34]. Thus, it is essential to monitor numerous paths in the design. Consequently, many Razor FFs and in-situ monitors will add a considerable area – which should be prevented due to cost reasons.

Another indirect monitoring structure for performance screening is the use of **ring oscillators (ROs)**. Such ROs are divided into two basic variants - generic ROs and design-dependent ROs [29]. Generic ROs are made from precisely one kind of standard library cell, e.g., inverters or AND gates. Design-dependent ROs are synthesized according to the design of the chip. Several paths of the chip are being selected, synthesized, and designed as ROs. However, such RO structures occupy much area because all structures need to be placed on the chip; in contrast, *functional path ROs* can solve the disadvantage of the area requirement.

Wu et al. [30] proposed the basic concept of functional path ROs. A functional path RO uses existing paths in the design and creates an RO structure from such paths. In the basic concept, path ROs are used to test small circuits. Further development of functional path RO was proposed by Wang et al. [31, 21] - called *Path-RO*. Here, the main focus is on measuring specific path delays as precisely as possible, which is only accompanied by high area and effort.

However, efficient implementation is only one aspect of performance screening with functional path ROs. There are millions of functional paths on a large **MCU**, and the path selection directly affects the quality of performance monitoring. Many previous research was conducted to characterize timing effects [35, 36, 37, 33] in functional path ROs. Rangan et al. investigated the design of ROs in terms of PVT sensitivity [38]. Therefore, the accuracy of the performance screening strongly depends on the sensitivity of the performance monitors used. [39].

In summary, there is a lack of efficient and automatic implementation of functional path ROs. The implemented ROs should fit into industrial design flows and use the industrial test and control infrastructures like the DFT environment. Another open point is selecting the functional paths to catch the right paths for performance screening. There is also a lack of measurement data from silicon. Furthermore, last but not least, how good are the functional path ROs for performance screening?

### 1.2.1. SMON Benchmark Module in this Work

In order to have a baseline against which the functional path ROs can be compared, an SMON (Speed MONitor) module is used in this work. This SMON module, utilized as a benchmark, contains different types of ROs as performance monitors. The structure of the SMON module originates from [24], but the SMONs have been re-implemented in the newer 28 nm CMOS technology used for this research to serve as a valid baseline.

Inside the SMON module is a mix of generic and design-dependent ROs. The module is conceptualized as a compact block positioned on the **MCU**. The various generic ROs consist of inverter gates, NAND gates, and NOR gates from different cell libraries. The design-dependent ROs on the SMON module emulate functional paths from the design. The SMON module is designed to contain up to 255 different ROs. However, to serve as a benchmark, the SMONs contain 27 ROs that have been used in previous work ([24]).

### 1.3. Contribution of this Work

The main contribution of this work is in the area of functional path ROs for performance screening. In particular, the contribution can be settled along the development process of an **MCU** from the per-silicon design part through the manufacturing until the post-silicon evaluation of an **MCU** in state-of-the-art technology.

Parts of the approaches proposed in this work have been peer-reviewed and published in official scientific conferences proceedings [40, 41, 42, 43] and workshop proceedings [44, 45, 46, 47, 48]. In addition, parts of this work have been published in the **IEEE Transactions on Very Large Scale Integration (VLSI) Systems** in June 2023 [49]. In order to emphasize the novelty of this work, **two patents** are published and granted in two different countries. Another outstanding achievement of this work is that [41] was honored with the **Best Paper Award** at the IEEE European Test Symposium (ETS) 2022.

Besides this, there are considerable co-contributions in peer-reviewed conferences [24, 50] and a publication in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD) [51].

The following listing associates the core contributions with this work's publications and related sections in this thesis.

- Automatic and scalable implementation of functional path ROs in large automotive **MCU** designs is presented in Section 3.1 of which parts are published in [40, 44, 45].
- The development of advanced implementation methods is presented in Section 3.2, which emphasizes the so-called self-enabling approach and natural loops. The self-enabling approach was published in [41], and a DE and US patent [52] was granted. The natural loop approach was published in [47] and patented in DE (granted) and US (published) [53].
- A path selection methodology was developed and validated using analog simulation and sensitivity analysis of the circuitry, this is presented in Section 4, and parts were published in [42, 46].

- Functional path RO results are presented, derived from a large automotive **MCU**, and their benefits for performance screening are presented in Part **II**. Sections of this part were presented in [42, 43, 48] and [49].

## 1.4. Outline of this Dissertation

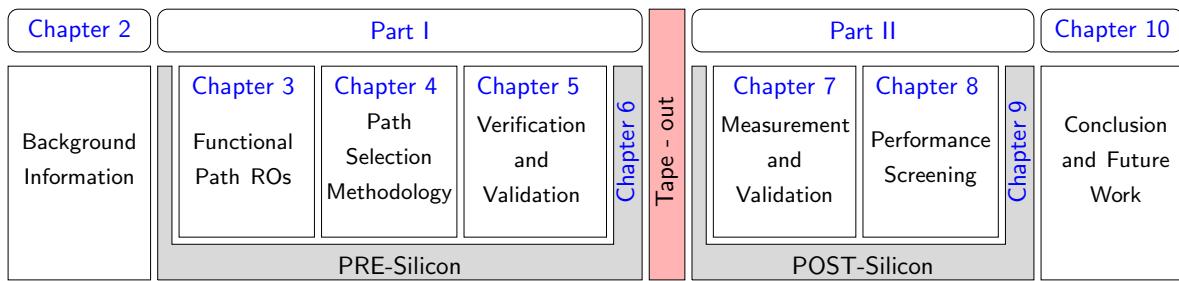


Figure 1.3.: Outline of the thesis.

The structure of this thesis is presented in Figure 1.3. Chapter 2 provides the background knowledge for this work. The industrial development flow of an **MCU** is recapped, and the timing and performance challenges are explained. Also, the fundamental testing knowledge is provided, and the most substantial Machine Learning (ML) approaches are explained.

The subsequent work is then divided into two parts: Part **I**: pre-silicon and Part **II**: post-silicon.

Part **I** presents the concept and implementation of the functional path ROs in Chapter 3. Chapter 4 presents the path selection methodology to find the best suitable functional paths for RO implementation. The pre-silicon verification and validation is explained in Chapter 5, followed by a comprehensive summary of Part **I**.

Part **II** presents the first measurement results and their validation in Chapter 7. Chapter 8 presents the use and results of functional path ROs for performance screening. A summary in Chapter 9 concludes Part **II**. The last Chapter – Chapter 10 – in this work concludes the thesis, including obstacles in an industrial context and further work.

## 2. Background

The background section provides an overview of the **microcontroller (MCU)** development and timing of digital circuits. Afterward, the testing of modern **MCUs** is reviewed, especially the performance screening. **Machine learning (ML)** use cases in **SoC** testing and the quality standards for automotive **MCUs** are also elaborated. The terms **SoC** and **MCU** are used as synonyms in this work and are not further distinguished.

### 2.1. Digital MCU Development Flow

In order to get an overview of the phases in which the proposed methods proceed, the industrial **MCU** design and test flow are presented. All methods in this work are suitable for industrial development flows. However, only a condensed overview of the **MCU** development flow is presented, as the detailed process is much more extensive [54]. The digital **MCU** development flow to develop a prototype **MCU** is shown in Figure 2.1.

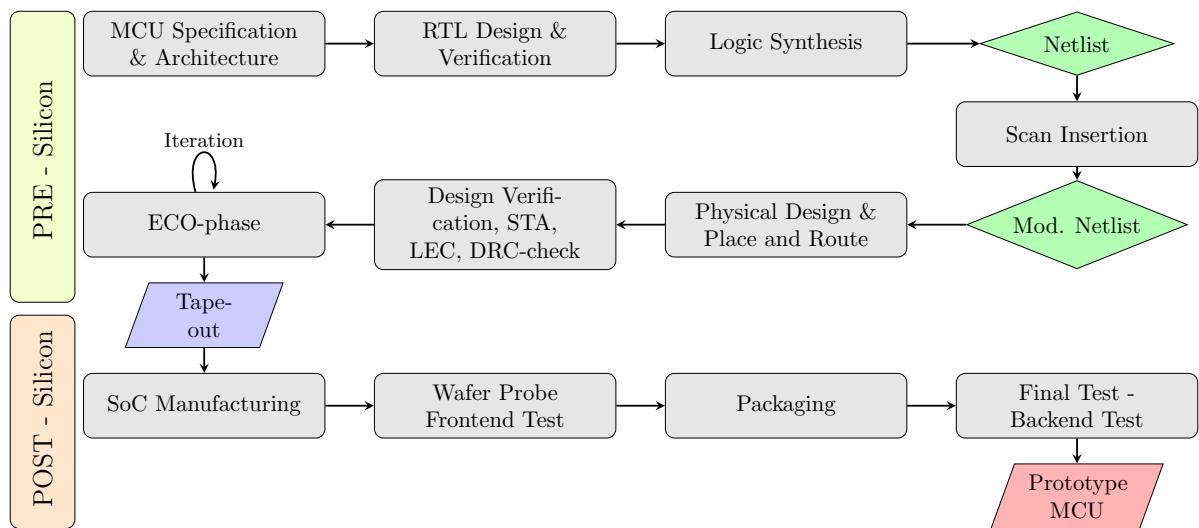


Figure 2.1.: Development flow of a prototype **MCU** divided into **Pre-Silicon** and **Post-Silicon**.

In general, the development process can be divided into the **Pre-Silicon (Pre-Si)** and **Post-Silicon (Post-Si)** phases. The arrival of the first **MCU** prototype indicates the transition from

---

## 2. Background

---

**Pre-Si to Post-Si.** The functional specification has to be defined in the initial phase, and the architectural chip elements, including all modules and submodules, are defined. The described functionality is translated into a hardware description language and a synthesizable **register-transfer level (RTL)** representation. With a proper **RTL** representation, the logic synthesis is started, and the abstracted **RTL** representation of the functionality is then translated into a logical gate-level design called *netlist*. The netlist of a design is a composition of logic gates and **flip-flops (FFs)** implementing a particular functionality previously described in **RTL**. The **RTL** representation and the netlist are constantly verified to check the desired functionality [55].

Once the netlist is verified, it is further modified with the **Design for Testability (DfT)** process (e.g., *scan insertion*), which results in a modified netlist. The **DfT** process enables proper testing of the manufactured chips; more details are explained in Section 2.4.1.

The modified netlist is then passed to the physical design. Floorplanning of the design, including power planning, clock tree synthesis, and place-and-route, is performed. The physical design is extensively verified and, if necessary, optimized.

An essential step in the design verification is ensuring the design's timing closure with the **static timing analysis (STA)**. All setup and hold times must be within the specifications under all allowed conditions. Plenty of other verification steps are also done, for example, the **Logic Equivalence Check (LEC)**, **Design Rule Check (DRC)**, IR droop is verified and cross-talk, and many more checks [55].

If there are issues with the functionality and timing of the design, an **engineering change order (ECO)** is usually used to solve the issue. An **ECO** is a process for fixing any design problems in a late design stage and is typically one of the last stages in the **Pre-Si** phase. The **ECO** can cause a minor modification in the netlist and the resulting change in the physical design. This **ECO** process is executed, and an incremental compilation run is performed. This means that only the parts of the design that are affected by the **ECO** process are changed. The rest of the design remains unchanged. Once the whole design is successfully verified, and all targets are met, the manufacturing of the **MCU** design is started, which is known as the *tape-out*. This is also the transition from **Pre-Si** to **Post-Si** phase.

The manufacturing process of **MCUs** in advanced **complementary metal oxide semiconductor (CMOS)** technology is a complex and elaborating process with plenty of chemical and physical steps on a silicon wafer. Multiple chips, also denoted as *dies*, are contained on a single wafer. After the wafer manufacturing, the **MCUs** are tested for the first time, called the **front-end (FE)** wafer test. The wafer is mounted on a chuck during the **front-end (FE)** wafer test. Afterward, the dies are separated by sawing the wafer into bare dies. The bare die, where the **FE** test is passed, is assembled in a package. The next step is the burn-in to detect early fails and ensure high reliability, followed by the final **back-end (BE)** test. The

finally produced **MCU** is extensively characterized, and it is validated, if the functionality is correct and all requirements are fulfilled. This is the terminus of the prototype development. If issues are identified, a redesign might be performed, or minor improvements are required. Finally, the **MCU** is transferred to mass production [56].

Note that the development process explained is a high-level overview; the entire **MCU** flow has much more detail that is neglected in this explanation.

## 2.2. Timing and Performance of digital Circuits

Modern **MCUs** consist of numerous architectural components. A large part consists of digital circuit elements, which is also the focus of this work. Besides that, there are also memories, analog circuit parts, and various interfaces on modern **MCUs**. Digital circuits are built on **CMOS** combinational and sequential circuit elements. All such circuit elements are part of a cell library containing **CMOS** circuit elements (**FFs**, Inverter, NOR - gates, AND - gates, ...) of different sizes, speeds, power, and other factors and denoted as *standard cells*. All **CMOS** elements are built using NMOS and PMOS transistors. The logic synthesis translates the **RTL** description into a netlist using such cell libraries.

The synchronous digital design uses edge-triggered **FFs** as sequential circuit elements. The **FFs** are triggered by a synchronous clock signal **CLK** distributed via the clock tree. Thus, there can be distinguished between clock paths and data paths. The clock tree distributes the clock signal over the chip, and data paths care for the chip's functionality. The data paths with combinational logic elements are arranged between the clock-triggered **FFs**. Thus, the data signal must propagate from the designated launch **FF** (**FF1**) through the combinatorial logic to the desired capture **FF** (**FF2**) in each clock cycle, as shown in Figure 2.2.

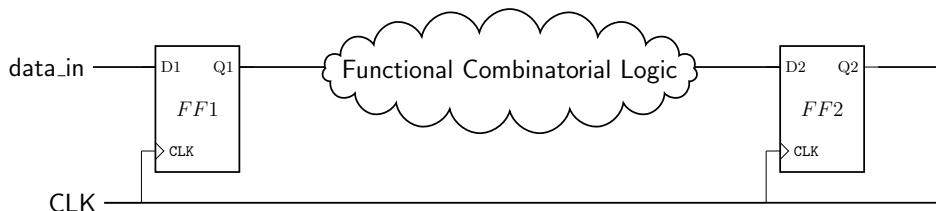


Figure 2.2.: Synchronous combinational logic path with launch and capture **FF**.

The clock-triggered **FFs** typically have 3 ports, 2 input ports and 1 output port. The port *D* is the input from the data path (*data\_in*), and the port *Q* is the output for the data path. The clock signal is connected to the port *CLK*. Once the *CLK* port detects a positive edge, the data is transferred from *D* to *Q*.

Thus, the data on *D1* in Figure 2.2 is latched from *D1* to *Q1* with a rising clock edge. The

## 2. Background

---

data then traverses the data path and the data should arrive at  $D2$  before the rising clock edge and then be latched from  $D2$  to  $Q2$  on another rising clock edge.

However, there is no instantaneous propagation from launch **FF** to capture **FF** due to individual delay time of the combinational logic elements of the data path. The data path delay is the sum of the delay of the individual library gates and the respective interconnects. In order to ensure proper functionality of the synchronous circuit, the timing between clock and circuit is essential. Therefore, certain timing constraints have to be met within the sequential circuits. Important timing constraints for the **FF** are the *setup* and *hold* time. The setup time  $t_{setup}$  is the minimal time previous to the positive clock edge in which the data input requires a stable signal. During the hold time  $t_{hold}$ , the data input has to remain stable after the positive clock edge. The relation between the minimum clock period  $T$  and the time constraints can be expressed as [57]

$$T \geq t_{c-q} + d_{Path_{MAX}} + t_{setup}, \quad (2.1)$$

and the hold time has to ensure such timing constraints

$$t_{c-q} + d_{Path_{MIN}} \geq t_{hold}. \quad (2.2)$$

Here, the clock-to-Q-delay  $t_{c-q}$  is the internal propagation delay required by the **FFs** to propagate the  $D$  to  $Q$  on a positive clock edge. For Equation 2.1, the worst-case propagation delay  $d_{Path_{MAX}}$  of the path must be considered. Whereas in Equation 2.2 the best case consideration  $d_{Path_{MIN}}$  of the path is taken into account.

Timing constraints must be met in terms of setup and hold time for a given clock period; otherwise, timing is violated. A valid timing of the circuit illustrated in Figure 2.2 is shown in Figure 2.3.

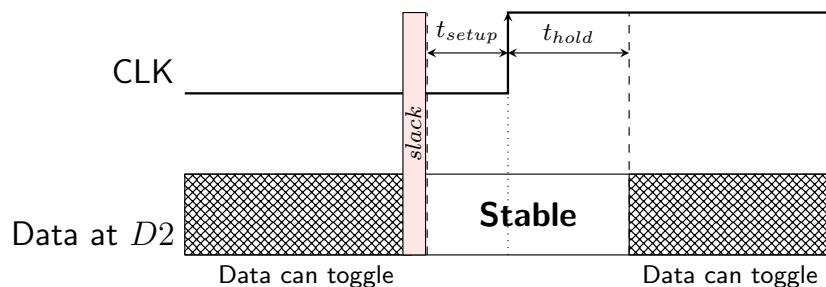


Figure 2.3.: Timing diagram of a capture **FF** in a synchronous digital circuit illustrating setup and hold time and slack.

The *slack* is also indicated in the Figure 2.3. The slack is the difference between the required arrival time where the  $t_{setup}$  is met and the actual arrival time. The setup timing is met when

the slack is positive (greater or equal to 0).

A synchronous **MCU** design consists of several millions of paths which all have to meet the timing given the clock period. Thus the performance  $F_{MAX}$  (maximum achievable clock frequency) of an **MCU** can be expressed as

$$F_{MAX} = \frac{1}{T}. \quad (2.3)$$

The shorter the clock period  $T$ , the higher the performance of an **MCU**. However, the timing constraints must be observed under all circumstances. The **Process-Voltage-Temperature (PVT)** variation plays an important role here.

## 2.3. PVT Variations

The **Process-Voltage-Temperature (PVT)** variation significantly impacts the timing of digital circuits, and it is becoming more critical in shrinking technology nodes [58, 59]. The digital circuitry has to ensure error-free working within the specified **PVT** range. The *process* variation is related to the variation in semiconductor manufacturing due to the complex process steps. The *voltage* variation impacts the operating voltage of the **CMOS** transistors, and the *temperature* is related to the die temperature, which also has a significant impact. In this section, **PVT** variations and their effects are explained. It also presents the methodology for handling the sources of variation in modern **MCUs**.

### 2.3.1. Process Variations

Process variation arises from the variability of process parameters during the complex manufacturing process of semiconductors. The process variation is caused by the limited control of such process parameters, which results in variability from the defined design target [60].

**MCUs** are manufactured by multiple chemical and mechanical processes based on a silicon wafer. A sequence of different process steps is executed called photolithography. Masks are used in the steps to build **CMOS** transistors on the plain silicon wafer. The hypothetical number of masks for building an inverter is six; however, the number and complexity of masks used for large **MCUs** can be enormous. The process steps, e.g., epitaxy, deposition, and implantation, are repeated until the **MCU** is manufactured. The metallization and interconnects are also part of the manufacturing process. Such parts connect the sources, drains, and gates accordingly and ensure power distribution [61].

The process variation impacts the film thickness, lateral dimensions, and doping concentration, resulting in variations in channel length and threshold voltage [62]. This impacts the

timing of the **MCU** and, therefore, the performance of each manufactured device [60]. The impact of the process variation can be classified into *inter-die* and *intra-die* variations.

### 2.3.1.1. Inter-die variation

Inter-die variations are process variations that affect all transistors on a die similarly, e.g., all transistors' channel length on a die might be too long related to the specified value. Those process variations are also called global process variations. The inter-die variation is further distinguished into **lot-to-lot (L2L)** variations, **wafer-to-wafer (W2W)** variations, and **die-to-die (D2D)** variations [61].

The wafers are processed in stacks called lots. There might be some maintenance steps from lot to lot, impacting the process variation. The same is also for **W2W** variation, which causes a different timing behavior from wafer to wafer. The **D2D** variation is visible on the wafer; e.g., a die on the wafer edge behaves differently from the die in the wafer center, mainly caused by **D2D** variations in manufacturing [61]. A wafer signature caused by **D2D** variation is shown in Figure 2.4, where the frequency of an **RO** is plotted for each die on the wafer.

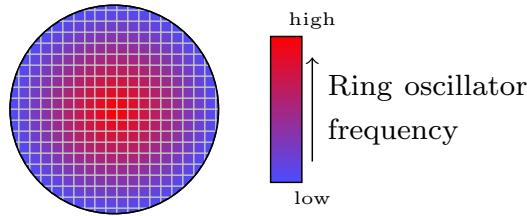


Figure 2.4.: Wafermap shows the **D2D** variations from the frequency of an RO.

The same RO is integrated at each die on the wafer. The dies on the wafer edge have a lower frequency of the RO than those in the center. Thus there is a radial **D2D** variation visible on the wafer [60]. Besides such centric radial **D2D** variation, several other wafer signatures are present in semiconductor manufacturing, which can also lead to defects on the wafer [63].

### 2.3.1.2. Intra-die variation

The intra-die variations affect various parts of the chip or transistors differently. They are also known as local variations or **within-die (WID)** variations. The **WID** variations were smaller in mature nodes than the **D2D** variations; however, they became more important in nanometer process nodes [61, 64]. Such **WID** variations can be categorized into two types: pure random variation and spatially correlated variations.

Pure random variations are local variations that are, by nature, randomly distributed across the chip with no recognizable pattern or signature. The main root cause of such variations

## 2. Background

---

are random dopant fluctuations (RDF) and line edge roughness (LER) [65].

The spatially correlated variations are also known as location-dependent variations. It shows that the parameter change of one device is correlated with the change of the same parameter of all other devices on the chip. The magnitude of the correlation between the parameters varies depending on their physical position on the chip. The main physical transistor parameters affected by spatially correlated variations are channel length ( $L$ ), channel width ( $W$ ), and oxide thickness ( $T_{ox}$ ) [60].

### 2.3.1.3. Interconnects

In addition to process variation in transistors, process variation in interconnects is also an important factor. The interconnects are wires that are used to connect the transistors, and they also have a significant contribution to the delay due to their resistance and capacitive components called *RC delay*. The process variations affect the line width and spacing, the metal and dielectric thickness, and the contact resistance. This variation, in turn, affects the RC delay. Also, long wires have significant resistance that dominates the RC delay of an interconnect. State-of-the-art **MCU** do have multiple layers of closely packed interconnects to cope with the design complexity on the **MCU**; therefore, the interconnects, and their RC delay become increasingly important [61].

### 2.3.2. Voltage Variations

Besides the process variation, the environmental operating conditions, such as voltage and temperature, significantly impact the timing of the circuitry and, therefore, its performance. The voltage variation is due to the supply voltage of the transistor. Since the manufactured nodes are rapidly decreasing, the **power delivery network (PDN)** and packaging follow at different paces. The supply voltage variation can be caused outside the chip due to the power supply and within the chip due to the package and interconnects. State-of-the-art **MCUs** operate in the sub-threshold region and are even more sensitive to voltage variation.

Voltage variations are categorized into two main categories: *IR drop* and *current derivative  $di/dt$  noise* [66]. The IR drop is also called voltage drop. It is caused mainly by the resistive components. That can be either off-chip (e.g., contact resistance, imperfect power supply) or on-chip caused by **PDN**. The IR drop is independent of the frequency and follows *Ohm's Law*. On the other hand, the parasitic inductance causes the noise of the current derivative. The switching activity of the transistors has the most significant influence on the noise of the current derivative. Strong load jumps can cause voltage undershoots or voltage overshoots [66].

Depending on the root cause of the voltage variation, the duration can be in the nanosecond

range for high-dynamic events until voltage variations in the microsecond range, depending on how fast the power supply can compensate for the variation. The voltage deviations impact the timing of the entire chip. The gate delay time is proportional to the voltage and follows the formula [67]

$$t_{gate} \propto \frac{V}{(V_{TH} - V)^\alpha}. \quad (2.4)$$

Where  $V$  ( $V_{DD} - V_{SS}$ ) is the supply voltage,  $V_{TH}$  is the threshold voltage of the transistor, and  $\alpha$  is a technology-dependent parameter. Thus, if the  $V$  is close to the  $V_{TH}$ , the voltage variation does have a major impact on the timing. There is a need to monitor the voltage precisely and react to occurring voltage variations immediately in designing a robust **PDN**, e.g., by using considerable supporting capacities. The tolerable voltage variation often found in the literature and on product data sheets is  $\pm 10\%$  of  $V$  [68, 69, 17, 61].

### 2.3.3. Temperature Variations

The second environmental condition which has a major impact on the timing behavior is the temperature. The temperature of the chips can be influenced by the ambient temperature or thermal hotspots on the chip itself. The ambient temperature depends on whether the chip is operated in different climatic regions (desert climate or polar regions) or how powerful the external thermal system (heating or cooling) is at the system level as well as on the application area (e.g., commercial, military) [61]. Thermal hotspots occur in high-switching active areas due to the power dissipated by the transistors. This also depends on how effective the thermal conduction of the chip can dissipate the heat from chip regions. Compared with the voltage variations, the temperature variations have a higher time constant in the range of milliseconds to seconds [69].

An increase in temperature elevates the delay of the interconnects due to their parasitic resistance. The temperature behavior of the gate propagation delay depends on the voltage, especially if the device is operated in the sub-threshold region due to weak carrier mobility [70]. This effect is called **inverse temperature dependence (IDT)**. The timing behavior and, therefore, the performance of a device depends on the dominance of the **IDT**. If long interconnect delays dominate the timing behavior of a device, the device will most likely be limited at high temperatures. If logic gates dominate the timing, cold temperatures are worse.

In conclusion, considering **PVT** variation in **MCUs** is crucial for reliable device timing and performance. Much effort is needed to ensure a proper function over the specified operation window. This includes that all hold and setup constraints are met. In order to achieve this, the timing behavior for each **PVT** case must be covered and verified during the design phase as well as validated in the **Post-Si** phase.

### 2.3.4. Methods to cope with PVT Variation

The **PVT** variations need to be considered during the design in the **Pre-Si** and the **Post-Si** phase since their timing influence is tremendous. Figure 2.5 comprehensively demonstrates the influence of the **PVT** variations with respect to the performance [71].

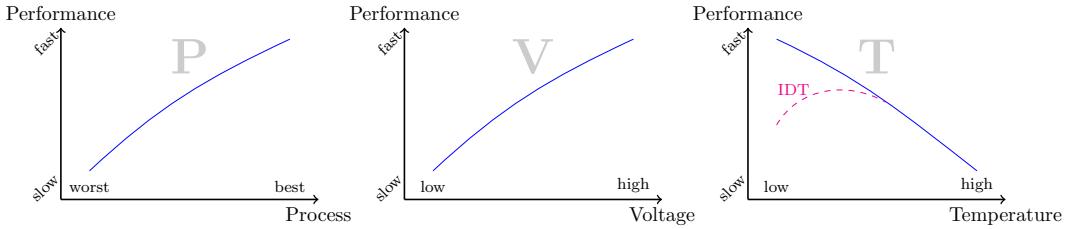


Figure 2.5.: Relation between **PVT** variations and performance.

The most common methodology to cope with the **PVT** variations in the **Pre-Si** design phase is performing *corner analysis* to investigate the specified **PVT** space with the help of powerful **electronic design automation (EDA)** tools. The **Post-Si** validation uses *corner lots* to explore the process range in manufacturing, and voltage and temperature are environmental components addressed with the test setup.

#### 2.3.4.1. Corner Analysis in Design Phase

Corner analysis, also known as corner case analysis or process corner analysis, is a technique used in design to evaluate the timing and performance of integrated circuits under different operating conditions. It involves simulating the circuit behavior across the **PVT** range in so-called corners. Especially, corner analysis aims to assess how the circuit behaves under the worst-case conditions.

In terms of process corners, the transistor manufacturing processes are clustered into *slow (S)*, *typical (T)* (also called *nominal*), and *fast (F)*. The **CMOS** technology uses two types of transistors: the nMOS and pMOS. The two transistor types are treated independently. That results in the process corners shown in Figure 2.6. The first character corresponds to the process corner of the nMOS, and the second character to the pMOS - as color-coded in Figure 2.6 [61].

Besides the transistor process corners, the interconnects are also considered with their parasitic RC components. The RC components are also simulated as slow, typical, and fast.

The environmental conditions - voltage and temperature - are also considered in the same way to be *S*, *T*, or *F*. The fast corresponds to high voltage and low temperature and slow corresponds to low voltage and high temperature, in between such conditions is the typical case [61]. Due to the **IDT**, the worst temperature case can change depending on the design,

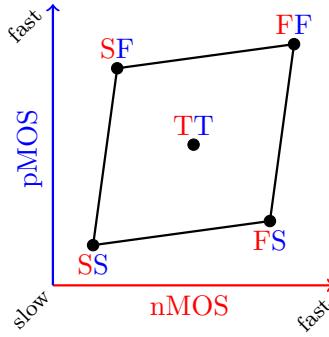


Figure 2.6.: Corner cases of a **CMOS** transistor.

as explained in Section 2.3.3.

In order to ensure a proper function of the design, a multi-corner analysis is required to ensure the correct timing behavior in each specified corner. The nomenclature of such a multi-corner is usually expressed in the three design corners (*S*, *T*, *F*) cases. Therefore an *SFTSS* corresponds to a *slow* nMOS, a *fast* pMOS, a *typical* RC, *slow* voltage (min), and *slow* temperature (hot). The design must be verified in every corner.

Timing verification using tools like **Simulation Program with Integrated Circuit Emphasis (SPICE)** is not feasible due to the complexity of large designs and limited computing power [72]. In addition, such tools are primarily intended for analog simulations at the transistor level. Therefore, a powerful method called **static timing analysis (STA)** is used to perform timing verification in digital circuits.

The **STA** is a method to investigate and verify the timing of the **MCU** without simulating the full design. The entire design is analyzed to determine if all timing constraints are met for every corner. The setup critical timing paths are typically suspected in the slow corners, whereas the fast corners face hold time violations. The term static refers to the circumstance that the design is analyzed in a static condition by not taking care of any input or output data - which is the case in performing a simulation. The **STA** is performed as one of the last steps before tape out to ensure the design's timing closure. **EDA** tools exist to accomplish an **STA** on the design and identify weaknesses. Especially the worst-case corners are focused on ensuring a robust design [71, 73].

However, the **STA** using the corner-based approach has limitations. Especially when **WID** variation becomes dominant, this can not be handled with the corner-based **STA** [74, 17]. There are methods like **statistical static timing analysis (SSTA)** [64] that manage the **WID** variations in designs - however, **SSTA** still needs to be consistently adopted across the industry [75, 32]. The typical approach to mitigate timing uncertainties caused by **WID** and noise is the use of additional margins. Those margins are on top added to the **STA** results,

adding pessimism and reducing the risk in the overall timing verification [71].

### 2.3.4.2. Corner Lots in Post-Silicon

In distinction to **Pre-Si** verification, the manufactured device must also be validated across the **PVT** corners in **Post-Si**. In order to cover the entire **PVT** space, corner lot wafers are manufactured. Such corner lot wafers are special flavors of wafer in which the manufacturing parameters are reflected. For example, the manufacturing parameters of all nMOS transistors on the wafer are tweaked to be slow. In contrast, all pMOS transistors in the wafer are manufactured to be fast - in other words, an *SF* corner lot wafer. The same parameter changes can be made for all manufacturing parameters that can be selectively influenced - these can be transistor parameters as well as interconnect parameters. The amplitude of the deviation can also be adjusted from the typical value, which is given in *sigma steps*.

The corner analysis of the environmental parameters voltage and temperature is easier to cover. The ambient temperature of the chip is adjusted using advanced temperature sources in the validation set-up. Even when the devices are on the wafer, such wafers are mounted on a chuck with very high-temperature stability. High-performance, high-precision power supplies can take over the power supply and allow the entire operating voltage range to be validated. The **PVT** validation of a device is part of the product characterization process [76].

### 2.3.5. Additional Safety Margins

Besides the **PVT** variation considered with the corner-based analysis, additional safety margins are required to ensure proper function. There are several reasons why those additional safety margins are required. Two reasons for additional margins in the timing verification are explained.

Modern **MCUs** operate with a clock frequency of several hundred megahertz. In operating mode with such high frequencies, electromagnetic cross-talk can occur, which impairs the signal integrity and, thus, the timing behavior. That electromagnetic cross-talk is, in most cases, an aggressor-victim scenario where the noise and emission of one particular region affect another region in which the timing error occurs [54].

The second added margin is the safety margin for aging. The **MCU** manufacturer has to ensure proper function in the delivery status and at the end of the life of the **MCU**. Several aging effects become more important in shrinking technology nodes, such as **negative bias temperature instability (NBTI)** and **hot carrier injection (HCI)**, which also affect the timing behavior of the circuit [34, 77, 78]. The margins required to *guardband* the aging over the lifetime depend strongly on the mission profile. In order to ensure a proper function independent from the mission profile, the added safety margins are intended to

cover the worst-case scenarios. This leads to the claim that the added safety margins are too pessimistic [79].

However, the additional safety margins in electromagnetic cross-talk and aging are mandatory if the **MCU** controls safety-critical applications in automotive.

## 2.4. Testing of MCUs

Since modern **MCUs** consist of more than hundreds of millions of transistors, the complex manufacturing process might lead to defects. Due to these defects, the **MCUs** might not be working properly. Thus, testing **MCUs** is essential in detecting such defects and ensuring an error-free operation. Efficient testing ensures functionality, timing, and performance, making it crucial for modern electronic devices. **Design for Testability (DfT)** is an essential method to enable test. The basic principle of **DfT** and its infrastructure and approaches are explained in this section.

### 2.4.1. Design for Testability (DfT)

**DfT** is a bunch of approaches used to make testing easier and more effective. **DfT** is adding further logic and components to the design to create an infrastructure for the test of the design. This infrastructure is inserted along the industrial design process, as Figure 2.1 explains with an important **DfT** step - the scan insertion.

A common **DfT** technique is the *scan test*; this includes *scan insertion* and *scan compression* and utilizing various fault models to detect defects efficiently. The following section (section 2.4.1.1) will describe this method because of the fundamental importance of this work. Another common technique in **DfT** are boundary scan and memory/logic **built-in self-test (BIST)**. Boundary Scan, or **Joint Test Action Group (JTAG)**, is a technology used to test and debug **MCUs** without direct access to their pins. Instead, a standardized interface called **JTAG** is used to communicate with the **MCUs**. **BIST** is an on-chip test infrastructure to detect defects without using external test equipment. There are approaches to testing digital **logic built-in self-test (LBIST)** and on-board **memory built-in self-test (MBIST)**, and those methods can also perform in-field tests [55].

#### 2.4.1.1. Scan Test

Scan test is a standard technique used in digital **MCUs** nowadays. This is a structural test method to detect defects in the circuit deterministically. In order to realize this, some scan infrastructure is necessary. This process is called scan insertion.

## 2. Background

---

**Scan Insertion** The scan insertion is done in two steps. First, the FFs and latches in the design are replaced with the so-called *scan FFs*. The scan FF is built with a **multiplexer (MUX)** and an ordinary FF; the scan FF is shown in Figure 2.7.

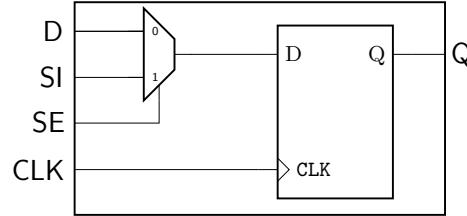


Figure 2.7.: Scan FF contains an ordinary FF and a MUX.

Besides the data input port (D) and data output port (Q), and the clock (CLK), the scan FF has two additional ports: the scan input port (SI, *scan\_in*) and the scan enable port (SE, *scan\_en*). The *scan\_en* controls whether the latched input data of the internal FF captures from D input - called functional (or mission) mode- or the data is captured via the *scan\_in* port - called scan mode.

The second step is the connection of the previously placed scan FF to the so-called *scan chains*. The circuitry connectivity after the scan insertion is schematically shown in Figure 2.8 for a small circuit, demonstrating three scan FFs connected in one scan chain.

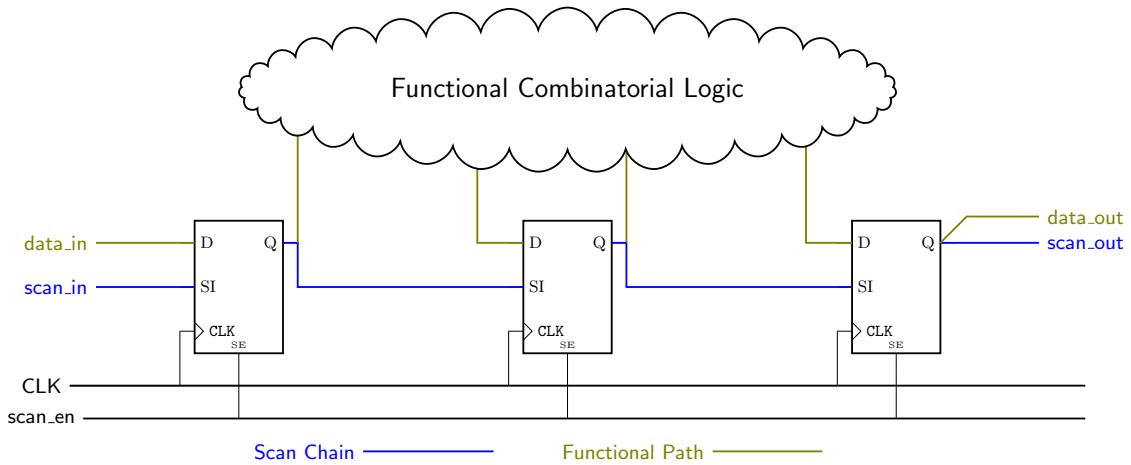


Figure 2.8.: Scan insertion replaces three FFs with scan FFs and connects them to a scan chain.

The scan chain is colored in blue and runs from the *scan\_in* through the *SI* of the first scan FF. From the *Q* of the first scan FF, the scan chain is connected to the *SI* of the next FF; this interconnection continues in this way. This is repeated until the last output of the scan FF within the scan chain, which is the *scan\_out* - the output of the scan chain.

Large MCUs contain a large number of scan chains. The *scan\_en* is valid for all scan chains.

## 2. Background

---

If the **MCU** is in scan mode (`scen_en` is high), the scan chains are sequentially loaded with 0 or 1 until all scan **FFs** have the desired value. After that load phase, the `scen_en` is switched off, and one or more functional at-speed pulses occur, called launch and capture. Once this is done, the `scen_en` turns on again, and captured values are unloaded from the scan chains. Such unload values are compared with the expected values. The **MCU** passes the test if every captured value equals the expected value. Such scan test procedure is shown in Figure 2.9.

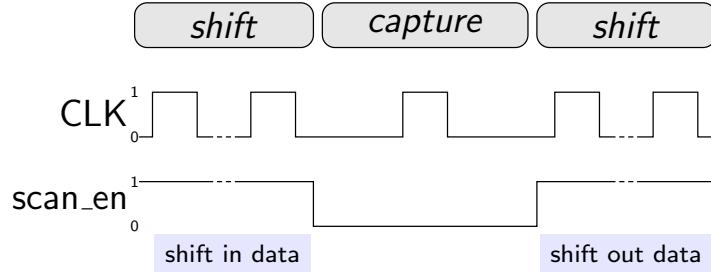


Figure 2.9.: Scan test procedure with a capture pulse.

The input values are called test/scan vectors or patterns. The scan test is repeated and consists of thousands of scan patterns. This provides controllability and observability for every scan **FF** during the scan test.

**Scan Compression** The scan chains become more extensive since more sequential scan **FFs** are included to cover the entire **MCU**. This would result in long scan chains and high effort to load and unload all scan chains with the scan pattern. Scan compression is an efficient method to reduce the length of the scan chain and subsequently reduce the test time by introducing a decompressor and compactor. Figure 2.10 shows the scan compression methodology.

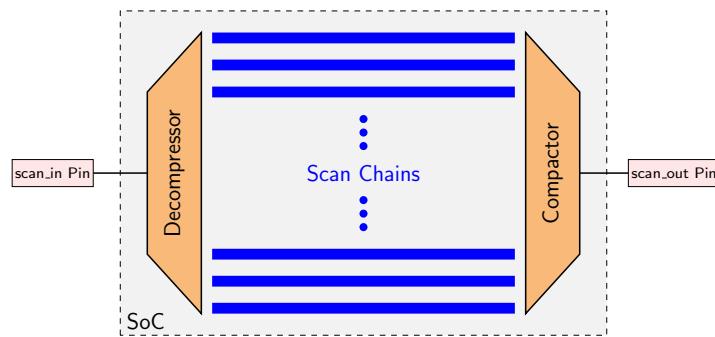


Figure 2.10.: Scan compression allows the parallel loading of scan chains with a decompressor and compactor through a single pin.

The scan chains are arranged between the decompressor and the compactor. In that way, the length of the scan chains is reduced. The scan chains are loaded and unloaded at high-speed

scan\_in and scan\_out ports. The shift speed of the scan chains remains nearly the same. However, the overall test time is reduced with this approach.

### 2.4.2. Fault Models

The test vectors/ scan patterns are not randomly chosen sequences of bits; instead, the test vectors are associated with fault models. Fault models are a formal abstraction where a fault is a logic description of the effect when a defect is present in the digital logic circuitry of the **MCU**. Dedicated test vectors are calculated using such fault models to test the digital logic for correct functionality. Such fault models aim to ensure defect-free circuitry after applying all variants of the dedicated fault model to all gates of the logic circuit. The metric to quantify the covered defect-free logic is called *fault coverage*. The fault coverage is calculated for each fault model, and several fault models are in use.

There are two classes of fault models: *static* fault models and *dynamic* fault models. Static fault models focus on static defects in the circuitry independent of any timing constraint. Commonly used static fault models are the Stuck-at-Fault model and the Bridging-Fault model [80, 81]. Such fault models are also used for IDDQ tests. Dynamic fault models, in turn, consider the circuit's timing behavior and are executed at speed. Widely used dynamic fault models are the Transition Fault, Gate Delay Fault, and Path Delay Fault Models. The Transition Fault model and Gate Delay Fault are aiming localized timing faults within the logic circuit. The path delay fault model considers the timing behavior of the entire path. The path delay fault model is used in this work and described in detail in the following.

**Path Delay Fault Model** The path delay fault model [82] aims to check if the path's timing is met within the clock cycle. In order to do this, the path delay fault model is executed at speed in the capture phase in contrast to the sequence shown in Figure 2.9. Therefore, two clock pulses are executed during the capture phase; the first is the launch pulse, and the second is the capture clock pulse. The test sequence is shown in Figure 2.11.

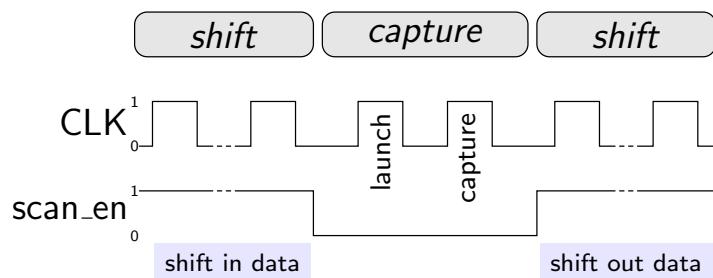


Figure 2.11.: Scan test using a path delay fault model.

During the shift in phase, the path to be tested is prepared, also known as path sensitization.

This means that all side inputs of the path need to be on a stable value. In addition, the side input needs to be on a non-controlling value. Once these requirements are fulfilled and the path is sensitized, the launch pulse is executed at the launch point; this is a controllable and observable point in the circuitry - a scan **FF**. The launched transition is then propagated through the sensitized path and is captured with the second clock pulse - the capture pulse - at the capture point, which is usually also a scan **FF**. The time between launch and capture clock pulse is at regular operation clock speed. The captured value is then shifted out, and the actual captured values are compared with expected values. If the expected values equal the actual values, the path delay test is successful. The path description to be tested must be provided for every path delay test.

The path delay fault patterns are distinguished in several modes based on sensitization criteria. The weakest path sensitization is functional sensitizable, which means that the path propagates the transition during the test, but the side inputs do not need to be stable. The non-robust mode ensures that the side inputs are only on a stable non-controlling value during the capture clock pulse. The tightest mode is the robust one, which requires stable non-controlling values at all side inputs during the launch and capture clock pulse [83, 84]. There is also the additional option of the hazard-free mode, which is even a little stricter than the robust mode. In this mode, an attempt is made to prevent glitches and reconvergence in the path [68]. Dependent on the sensitization mode chosen, it becomes more challenging to calculate a path delay test pattern that fulfills all sensitization requirements.

In addition to the sensitization mode, the clock handling can also be adjusted, the **launch-off-shift (LOS)** and **launch-off-capture (LOC)** mode [85, 86]. The **LOC** method was implicitly introduced in Figure 2.11. Such a mode allows a clear distinction between shift and capture - the shift phase cares for path sensitization, and the two at-speed clock pulses are executed in the capture phase. Instead, **LOS** squeezes the launch clock pulse into the end of the shift phase. Both approaches have their advantages and disadvantages [68]. Due to strict separation in shift and capture, the **LOC** will be the essential mode in this work.

There are many constraints in the calculation of scan patterns for large **MCUs**, so **EDA** tools are used, especially the **automatic test pattern generation (ATPG)**.

#### 2.4.2.1. Automatic Test Pattern Generation

**ATPG** is a methodology to generate test patterns used in the industrial environment by considering the fault models mentioned in Section 2.4.2. An **ATPG** tool is a software tool that generates test patterns for a given **MCU** design. The **ATPG** tool requires the netlist of the design with the already placed scan infrastructure. Based on the fault model determined, the **ATPG** tool calculates scan patterns to achieve high fault coverage for a wide range of potential faults in the circuit. The resulting scan patterns are in an ASCII file format called

waveform generation logic (WGL) or standard test interface language (STIL). The scan pattern in **WGL/STIL** format is required to test an **MCU** in an industrial test environment using **automatic test equipment (ATE)**.

### 2.4.3. Automatic Test Equipment

The **automatic test equipment (ATE)** is the hardware instrument on which the **DUT** is mounted during the test procedure. The **ATE** provides power to the **DUT** and applies all test patterns. The **ATE** itself is a complex real-time system that can leverage many **DfT** methodologies. The scan patterns in **WGL** format are read into the **ATE**. Regarding the execution of the test, the **ATE** generates the stimuli for the **scan\_in** port and loads the scan chains. The clock signal is also controlled via the **ATE**, as well as the comparison of the **scan\_out** values with the expected values is checked on the **ATE**. Such **MCU** testing can be done on the wafer level where the pads on the die are connected with needles from a probe card to the **ATE** or with package **MCU** which is then mounted on a socket through the **ATE**. With the **ATE**, fast, precise, and automated execution is enabled [87, 55].

## 2.5. Performance testing

In this section, the performance testing is explained. The performance of an **MCU** is the maximal achievable clock frequency of the device under worst case conditions. The performance of an **MCU** is also denoted as  $F_{MAX}$ . The approach to test the performance is called *performance screening*, in which the  $F_{MAX}$  of every device is checked. The term *speed binning* is also used for this approach which suggests an instantaneous sorting in categories according to the speed of the devices.

The performance testing of large modern **MCUs** is challenging due to the complex design and **PVT** variations. Structural dynamic scan patterns (see Section 2.4.1.1 - transition fault pattern, path delay fault pattern) are not suitable for precise performance testing. This has two reasons, the scan patterns provide only a pass/fail criterion, and second is not possible to determine the unique performance limiting paths. Many near-critical timing paths are in modern **MCU** designs, resulting in a timing wall. As a result, it is practically impossible to determine the path that is causing the performance limitation on large **MCUs**.

Therefore, other approaches are necessary to determine the performance of the chip. The following sections present two commonly used methods which are central pillars of this work.

Note that the performance testing does not aim to detect devices with physical defects (e.g., shorts, opens); structural scan tests are used for this purpose.

### 2.5.1. Performance Monitors

One way to determine the performance of a chip is to use indirect performance monitors [26, 27, 28, 29, 30, 31, 21]. Such performance monitors are **ring oscillators** (ROs). An RO is an odd number of  $N$  inverting logic gates connected serially, forming a closed loop. The basic structure of an RO using inverter gates is shown in Figure 2.12.

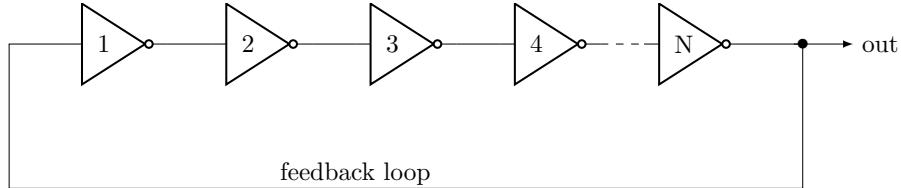


Figure 2.12.: Basic principle of an RO using inverter gates.

Each logic gate in the RO and the interconnects in between add delay to the overall timing of the RO. A positive or negative edge is launched at the start of the RO once the power supply is switched on, and the signal propagates through the logic gates in the RO. Due to the odd number of logic gates and, therefore, the implicit inverting behavior of the RO, the positive/negative edge becomes inverted at the end of the RO. Then the inverted edge is propagated through the feedback loop, triggering the next negative/positive edge. This results in a continuous oscillation where the frequency of the RO is the critical metric that is measured on the output.

The oscillation frequency  $f$  of an RO is expressed as

$$f = \frac{1}{2T_{RO}}, \quad (2.5)$$

where  $T_{RO}$  is the delay time of the RO.  $T_{RO}$  is calculated as follows,

$$T_{RO} = T_{Gate} \cdot N + T_{Loop}. \quad (2.6)$$

$T_{Gate}$  is the propagation delay for each of the  $N$  gates in the RO, and  $T_{Loop}$  is the interconnect delay. The timing of the gates used in the circuit behaves similarly to those used for the ROs. This is why such ROs can be used as indirect performance monitors.

In order to utilize ROs for performance testing, more than the basic RO structure shown in Figure 2.12 is needed. Thus there are various RO designs proposed and analyzed in literature. A fundamental distinction can be made between the two types of ROs: *generic ROs* and *design-dependent ROs* [29].

Generic ROs consist of a homogeneous logic gate type, e.g., inverter or NAND gate. Standard gate libraries used in large MCUs designs typically consist of several standard logic

---

## 2. *Background*

---

gates in different driver strengths and further variations in the number of inputs/outputs and threshold voltages. Building one **RO** out of each used logic gate in the cell libraries would result in hundreds of generic **ROs**. Another way is to leverage the design information of the chip and build so-called design-dependent **ROs**. Such **ROs** aim to mimic the design of the chip. In order to do this, several methods are used, from straightforward path replicas to sophisticated synthesis algorithms considering the entire **PVT** space [38]. In the end, the performance test's accuracy and quality highly depend on the sensitivity of the performance monitors used [39, 88].

Also, the value of  $N$  has a significant impact. The visibility of **D2D** and **WID** process variation within an **RO** depends on the number of logic gates  $N$  in the **RO**. Especially for smaller technology nodes ( $<40\text{ nm}$ ), the **WID** variability is only visible with short **ROs** containing less than 10 gates. In comparison, the **D2D** variation is independent of  $N$  [88, 89]. Such **ROs** follow the process variation, usually Gaussian distributed [90]. In addition,  $N$  also affects the oscillation frequency, which is a considerable limitation for the frequency measuring gear and the resulting accuracy, and each additional gate contributes to the leakage current, which is also a critical variable in **MCU** requirements [87].

### 2.5.2. Functional Testing

Another fundamental approach in performance testing is the execution of functional tests. A functional test is named due to the fact that the functional test cases are executed on the **MCU**. Such functional test cases simulate different workloads or functional test cases of the **MCU** to find defects and complement structural testing [10].

The functional test is distinguished into two subgroups. The first method is the traditional functional testing where small code pieces are uploaded to the **MCU** on the **automatic test equipment (ATE)** and executed. The second method is the **system-level test (SLT)**. The **SLT** is a time-consuming test where the **MCU** is mounted in an application-like board, and several test cases are executed. Such test cases can be customer application test cases or reusing verification stimuli [13]. **SLTs** require lots of test time and are challenging in high volume production [25].

In contrast to the structural scan test, traditional functional testing and **SLT** do not have straightforward coverage metrics, making it difficult to determine the testing quality [13].

In order to use the functional test for performance testing, there is a particular procedure. The clock frequency (execution frequency) of the **DUT** is executed at a low frequency in an infinite loop. Then step by step, the clock frequency is increased. This is continued until the maximal execution frequency without a failure. Using that approach, the performance of a device is determined given the particular functional test case [11].

## 2.6. Machine Learning in IC Testing

### 2.6.1. Machine Learning Basics

The term *machine learning* was introduced by Arthur Samuel in 1959 [91]. **machine learning** (**ML**) describes a subfield of *artificial intelligence*. **ML** facilitates computers (machines) to learn from existing data and predict the outcome of non-seen data without explicitly programming the computer. **ML** uses mathematical and statistical methods to identify the provided data's patterns, relations, or similarities. The field of **ML** has been rapidly increasing in the last decades. Meanwhile, **ML**-based techniques can be found in numerous applications and industrial scenarios.

The scope of **ML** is to find a function  $f$  that describes the relation between the input data  $x$  and the output data  $y$ :  $f : x \mapsto y$ . Since the function  $f$  is unknown and shall be learned by **ML**, the **ML** calculates an approximation function  $\hat{f}$  using the provided dataset  $\mathbb{X}$  -  $\hat{f}$  is also known as the **ML** model. Thus the input data  $x$  and output data  $y$  are vectors. Each input  $x_i$  is called a *feature*, and the output  $y_i$  is called a *label*. In order to create an **ML** model, the provided dataset is split into a *training set*  $\mathbb{S}$  and a *test set*  $\mathbb{T}$  (or validation set). The training set is used to train the **ML** model, which means finding a suitable approximation function that maps the features of  $\mathbb{S}$  to the labels. Once the training is done and the **ML** model exists, the **ML** model is validated. The **ML** model validation feeds unseen features from the test set  $\mathbb{T}$  into the **ML** model, and  $\hat{y}_j$  is calculated. Then the  $\hat{y}_j$  and known  $y_j$  deviation from the test set is compared. The **ML** model is scored depending on how well the predicted data and the known data fit. Scoring metrics are, for example, the **mean absolute error (MAE)** and the **root mean square error (RMSE)**.

**ML** can be divided into different subfields as shown in Figure 2.13. The subfields used in this work are *supervised learning* and *unsupervised learning*.

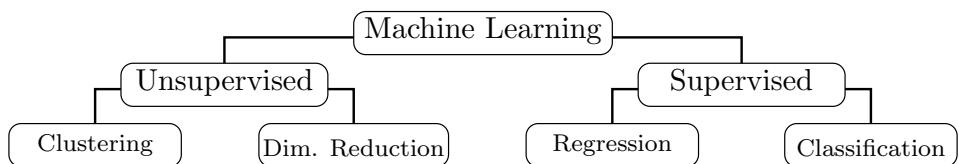


Figure 2.13.: **ML** overview of approaches in this work.

**Supervised ML** is the learning approach in which the features are mapped to the labels, and the **ML** model is considered a black box. Such a supervised learning algorithm aims to calculate a model that maps the input to the output data. Using the trained **ML** model, it is possible to predict the outcome of non-seen data using only the input data. The supervised **ML** can be categorized into *classification* and *regression* [92].

## 2. *Background*

---

The classification-based **ML** classifies the labels into discrete categories or classes. Therefore the labels should be in a categorial data format. A straightforward example is the classification into TRUE or FALSE based on the given features. The feature data type does not have to be categorical.

In contrast, regression-based **ML** provides continuous numerical values for the output variable. They are often used to make precise numerical predictions.

**Unsupervised ML** is applied when only features are available for the training, and the labels are unknown. Thus it is not supervised by the labels - therefore, unsupervised. This type of learning intends to find patterns and statistical dependence in the features without prior knowledge of the outcome. Two often used unsupervised methodologies in **ML** are *clustering* and *dimensionality reduction* [92].

Clustering describes the process of dividing a dataset with features into groups or clusters with the same or similar characteristics. The clustering algorithm uses distance-based, density-based, or hierarchical approaches to cluster the dataset into distinct groups [93, 94, 95].

On the other hand, dimensionality reduction intends to transform (or filter) the features into a reduced feature set. This should be done by keeping valuable information from the dataset. An often-used approach in dimensionality reduction is **principal component analysis (PCA)** which transforms the feature space into a reduced feature space by considering the eigenvalues [96].

Dependent on the **ML** approach, there are different scoring and error metrics explained once used in the later sections.

### 2.6.2. Machine Learning in Testing

Modern **MCUs** have become larger and more complex over recent years. This results in higher effort in testing to ensure the same or even higher quality. **ML** has become a well-established method for making testing more efficient and manageable in the testing of **MCUs** [97, 98].

However, once **ML** is applied to real-world problems, it can help and harm. One of the most crucial things in **ML** applied to test is the choice of the training set. On the one hand, it shall identify outliers that are a risk for proper training. On the other hand, the trained **ML** model shall be robust and able to generalize. Thus, **ML** helps to determine devices in pass and fail and many other areas in testing. This work uses **ML** to manage the difficult task of performance screening in automotive **MCUs**.

## 2.7. Automotive Quality

Automotive MCUs means that such MCUs are used in automotive applications that are often responsible for safety-critical systems. Such functional safety systems can be braking, steering, airbag systems, and numerous other automotive applications. Also, functional safety is essential for advanced driver-assistance systems (ADAS) and autonomous vehicles. The guidelines and standards for electronic components in functional safety automotive environments are described in ISO 26262 [99]. In addition, there are qualification specifications and requirements defined in AEC Component Technical Committee [5] agreed upon and defined by a large automotive community. These guidelines and requirements show how important it is to deliver high-quality automotive MCUs.

The quality and reliability of automotive MCUs are specified in defective parts per million (DPPM). In literature, the term PPM and the notation in defective parts per billion (DPPB) can be found. The critical metric remains the number of defective devices that successfully pass the test. Here, the devices are related to several ground truths (millions, billions). However, no severe rule exists for DPPM rates in the automotive industry [100]. Nevertheless, there is a solid strive to ensure zero defect quality, which means 0 DPPM accepted [101]. One of the most essential things to ensure zero defect quality is high qualitative testing [100, 102, 6].

The outcome of semiconductor testing is either the device passes all tests, or it fails the testing (one or more tests). On the other hand, there is the chance that the testing result is not correct. Either the test indicates that the device is pass whereas the actual device should fail or vice versa. This circumstance can be visualized in the confusion matrix shown in Figure 2.14. Such a metric can be used for all classification problems and ML classification.

		Actual Device	
		Negative	Positive
Predicted	Negative not working	True Negative (TN)	False Negative (FN)
	Positive working	False Positive (FP)	True Positive (TP)

Figure 2.14.: Confusion matrix of a classification problem.

The confusion matrix have four potential outcomes. The true positive (TP) and true negative (TN) is clear - the test (predicted) result equals the actual state of the device. A false negative

## 2. Background

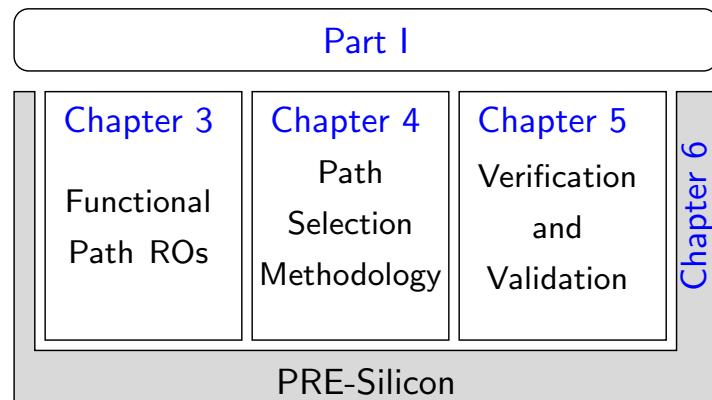
---

(FN) that is tested fail but is actually pass will cause yield loss. Yield describes the proportion between devices tested as non-defective and the total number of tested devices. In order to make it clear: FN are fault-free devices that are thrown away due to the test result - the yield decreases. However, concerning automotive quality, the false positive (FP) devices are the most critical, also known as escapes, and harm the quality. In order to handle and manage this issue to guarantee automotive quality, the six sigma criterion can be deployed.

### 2.7.1. The Six Sigma Guardband

The six sigma approach [103, 104] is a quality enhancement methodology and was invented by Bill Smith in the 1980s. This approach aims to deliver high-quality products with as low defect rates as possible. This approach is applied to various quality management approaches to reach the ultimate quality goal. The six sigma approach assumes a Normal Distribution of the data under investigation. Thus, the mean is the center of the distribution, and one  $\sigma$  is the average deviation distance from the mean value. 68% of the distribution area is between  $\pm 1\sigma$  of the mean value. Therefore, with a  $\pm 6\sigma$  range, there are at least 99.99966% in the distribution area. This corresponds in a statistical term with a defect rate of at least 3.4 **defective parts per million (DPPM)** - known as the *Six Sigma Level* [104]. The six sigma level of 3.4 **DPPM** assumes that a  $\pm 1.5\sigma$  shift of the Normal Distribution around the mean value is present. The shift margin is considered a batch-to-batch variation (**D2D** process variation) of the production concerning the high-quality level. If the process is perfectly centered, the resulting six sigma approach shows only a theoretical value of two defective parts per billion [103].

Hence, translating this into the testing environment, a margin of up to six sigma is needed to ensure automotive quality with minimal **DPPM** rates. The six sigma approach can be applied to ensure specific **DPPM** rates in performance testing, e.g., performance screening.



Outline of Part I.

## Part I.

# PRE-Silicon

# 3. Functional Path Ring Oscillators

## 3.1. Basic Concept

The *functional path RO* approach uses functional combinational logic paths within the **MCU**. The functional combinational logic paths run directly from register to register or from memory to memory and are typically used in the functional application of the **MCU**. Therefore, the behavior of the **MCU** is represented by the functional paths. A functional path in the design is denoted as  $p_i$ .

Two significant changes in the circuitry are required to create a functional path **RO** out of an ordinary functional path, as shown in Figure 3.1. A **multiplexer (MUX)** is inserted at the start point of the combinational logic path, and a *feedback loop* is inserted connecting the end point of the path with the start point of the path. An **RO** requires inverting behavior by default (see. Section 2.5.1); the same requirements apply to the functional path **RO**. If the functional path behaves non-inverting, an inverter can be placed along the feedback loop. The **MUX** can be switched between functional mode (0) and oscillation mode (1); the latter can only be activated for test purposes. The control infrastructure maintains the control pin of the **MUX** (*enable*). Also, the oscillation frequency (*observe*) for further processing is connected with the control infrastructure. For further details regarding the control infrastructure, see Section 3.4.

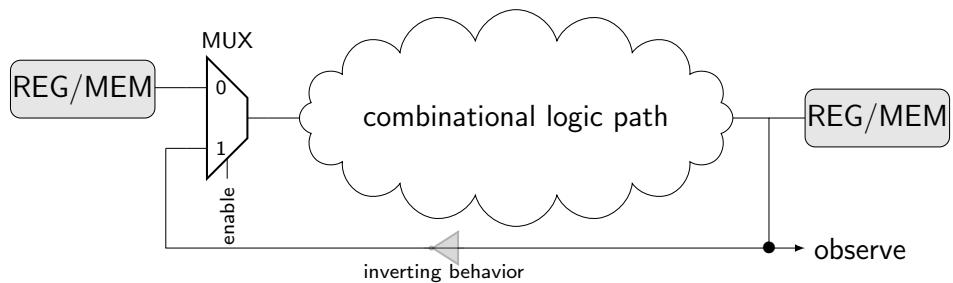


Figure 3.1.: Example of a functional path **RO**. Adapted from [40] © IEEE 2021.

However, the path by itself will not oscillate unless all supporting logic into the path is constrained to propagate the oscillating values along the path. Therefore the functional combinational logic path has to be *sensitized* for a stable oscillation. The sensitization guarantees

### 3. Functional Path Ring Oscillators

---

that all side inputs must be on a non-controlling stable value to ensure a stable oscillation of the functional path **RO**. A path sensitization with the respective side inputs is described in Figure 3.2.

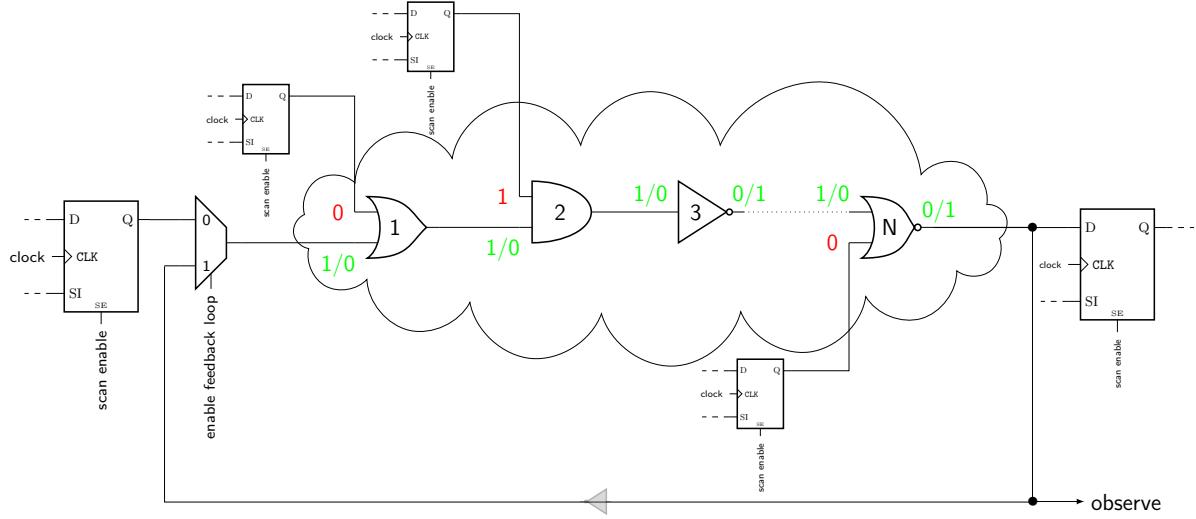


Figure 3.2.: Path sensitization of a functional path **RO** with the scan architecture. Adapted from [40] © IEEE 2021.

When the **MUX** changes from functional mode to oscillation mode and the path is sensitized, the functional path **RO** oscillates by default due to its inverting behavior.

Path sensitization can be accomplished by adding supporting logic to constrain all side inputs. Such an approach leads to many additional logic gates to the design and, therefore, much area overhead, which must be prevented in the competitive **MCU** market. The preferred solution is to use the existing **DfT** environment placed by scan insertion. A commercial **ATPG** tool is used to compute scan patterns based on an existing scan architecture that supports all logical assignments.

Commercial **ATPG** tools support many **DfT** fault models. Fault models are classified into static and dynamic fault models. Static fault models, e.g., Stuck-at and IDDQ, can not handle the sensitization automatically. Each side input of a gate along the **path under test** (**PUT**) has to be constrained with the non-controlling value. Such constraining has to be done manually in static fault models. Thus, much effort is required, and the **ATPG** tool has difficulties handling the constraints, as it has to calculate a scan pattern (e.g., stuck-at), taking into account all constraints of the side inputs. This, however, is not possible in many cases.

Dynamic fault models can support sensitization. In particular, the **ATPG** tool can perform dedicated path sensitization using the path delay fault model. The **PUT** is passed to the **ATPG** tool, and the tool attempts to sensitize the path with a scan pattern in path delay mode. This eliminates the need for manual handling of boundary conditions. The calculated path

delay pattern ensures the oscillation of the functional path **RO**. The **ATPG** tool in path delay mode has different levels of sensitization. The robust detection provides the most powerful path sensitization, independent of a clock event. In addition, the **LOC** method [85] is needed to generate the path delay pattern. This is because a fully sequential procedure is required to sensitize and measure the functional path **ROs**.

Approaching the sensitization problem with **ATPG** has the disadvantage of limited path delay efficiency of commercial **ATPG** tools, which means that only a limited number of paths can be sensitized using a suitable robust detection algorithm. Also, the **LOC** method reinforces such limitations due to the sequential **ATPG** algorithm of the **ATPG** tool in the **LOC** method [86]. However, the lack of path delay efficiency is not a significant limitation for functional path **ROs**. In large automotive **MCUs**, many paths can be sensitized with the **ATPG** tool in path delay mode.

The path delay fault model using the **LOC** method has a fixed test procedure, including clock activity (see Figure 2.11). The resulting path delay scan pattern must be modified in some way to be suitable for the functional path **RO** approach. A typical path delay pattern has three phases, the *shift-in phase*, the *at-speed phase* (for the launch and capture pulse), and the *shift-out phase* [86]. An additional phase is required when applying such a pattern for the functional paths **ROs**. The additional phase is the **RO** frequency *measurement phase*, which is inserted after the shift-in phase. The four phases of the functional path **RO** pattern are shown in Figure 3.3.

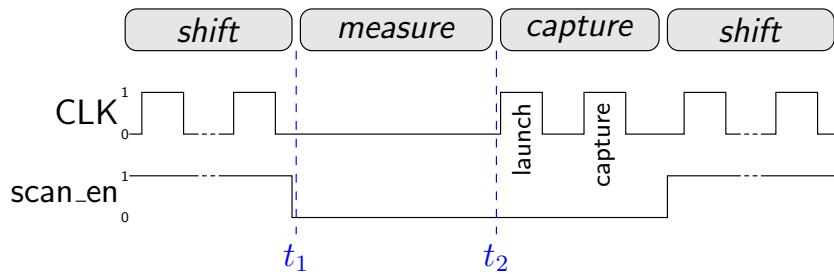


Figure 3.3.: The test sequence during the scan test pattern. Adapted from [40] © IEEE 2021.

The **PUT** is sensitized during the shift-in phase. At  $t_1$  the **MUX** is switched from functional mode to oscillation mode, and the functional path **RO** starts to oscillate by itself. The oscillation frequency is measured in the measurement phase from  $t_1$  to  $t_2$ . During the measurement phase, the **ATE** determines the oscillation frequency of the **RO**. The frequency value is stored in the test database for further processing. The at-speed launch and capture phase is irrelevant to the functionality of the functional path **ROs**.

The integration of the functional path **RO** measurement into the context of the **ATPG** scan pattern enables a fast measurement. The functional path **ROs** can then be activated with the

scan pattern and thus easily implemented in the industrial design flow.

The basic implementation presented above is called *Option 0*. All functional paths  $p_i$  that can be sensitized with a path delay pattern  $v_{p_i}$  are in the set  $P$ . The set  $P$  contains  $n$  functional paths, which can be represented in this way:  $P = \{p_1, \dots, p_n\}$ . All paths in  $P$  are eligible for the Option 0.

However, the main disadvantage of the Option 0 is the routing overhead, especially for the **MUX**'s enable signal and the observe signal used to measure the frequency. Both signals (enable and observe) have to be routed across the **MCU** for each **RO** individually. Accordingly, there will be many routing paths added in a large automotive **MCU**, especially when the functional path **ROs** are spatially distributed across the chip. Therefore, the upcoming section focuses on advanced implementation concepts to make the functional path **ROs** more efficient in terms of implementation.

## 3.2. Advanced Implementation Concepts

The functional path **RO** approach saves a lot of chip area compared with conventional **RO** approaches and enables a promising monitor structure by utilizing the **DFT** scan environment. However, the automotive **MCU** market is competitive, and every additional gate or routing line impacts the margin in the subsequent mass production of such **MCUs**. Therefore, this section focuses on concepts to implement the functional path **ROs** more efficiently to (i) minimize the impact on the functional circuitry and (ii) require as few additional gates and routing lines as possible. In doing so, the natural properties of the circuit are exploited by an intelligent combination and selection of functional paths.

### 3.2.1. Circuitry-Wise Optimization

The functional paths are distributed across the **MCU**. The idea of circuit-wise optimization is to find functional paths that inherently cause low routing overhead if they are implemented as **ROs**. Each functional path **RO** runs from the launch **FF** to the capture **FF**; both **FFs** are scan controlled. In state-of-the-art large **MCUs**, many such **FFs** are multi-bit **FFs**. This means they have numerous in- and output pins using the same synchronous clock signal. Consequently, there is a high likelihood that some functional paths share the same **FF** using different pins.

There are four types of path topologies identified that indicate lower routing effort. The goal is to find functional paths which share the same **FFs**. The four types are shown in Figure 3.4.

The launch / capture **FF** of a path  $p_{i/j}$  is denoted as  $LFF_{p_{i/j}} / CFF_{p_{i/j}}$ . The first type, Type 1 (Figure 3.4a) analyzes each path  $p_i$ . If the  $LFF_{p_i}$  is equal to the  $CFF_{p_i}$ , it can be assumed that

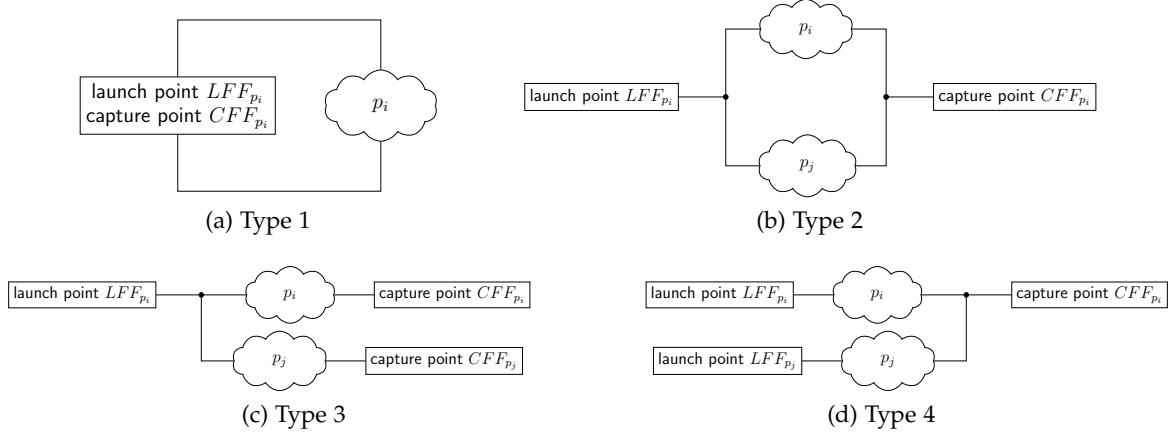


Figure 3.4.: Four path topologies of functional paths.

the path is self-contained, which means that the path is a cycle. In this case, routing for the feedback loop is not necessary.

Figure 3.4b presents Type 2 the parallel paths, which checks if two paths ( $p_i$  and  $p_j$ ) share the same  $LFF_{p_i}$  and  $LFF_{p_j}$ . In that case, they can share the feedback loop because only one RO is activated during functional path RO measurement. Therefore, only one feedback loop is necessary for path topologies in Type 2. Furthermore, also only one routing line for the observe signal is needed.

Type 3 (Figure 3.4c) and Type 4 (Figure 3.4d) are less restricted scenarios of Type 2. Instead of simultaneously checking for the same  $LFF_{p_i}$  and  $CFF_{p_i}$ , Type 3 only checks for the same launch FF, and Type 4 checks for the same capture point. The advantages are that only a single enable signal or a single observe signal is required.

Each path is investigated regarding similarities of Type 1, 2, 3, 4. As described in Section 3.1 all functional paths must be sensitizable with a path delay pattern. Those functional paths, which are sensitizable and have the properties from Type 1, 2, 3, 4, are in the subsets  $P_{Type1,Type2,Type3,Type4} \subseteq P$ .

Another way to use topological properties is the concept of *Natural Loops*. Instead of looking for common FFs of the functional paths, natural loops look at the course of the paths across the chip.

### 3.2.2. Natural Loops

The concept of natural loops is to use two or more functional paths and connect the end-point of one path to the start-point of the following path to create a functional path RO. The goal is to identify paths that can be connected with a natural loop in a way that the additional

routing needed for the feedback loop becomes minimal. Figure 3.5 shows an example of two natural-looped paths.

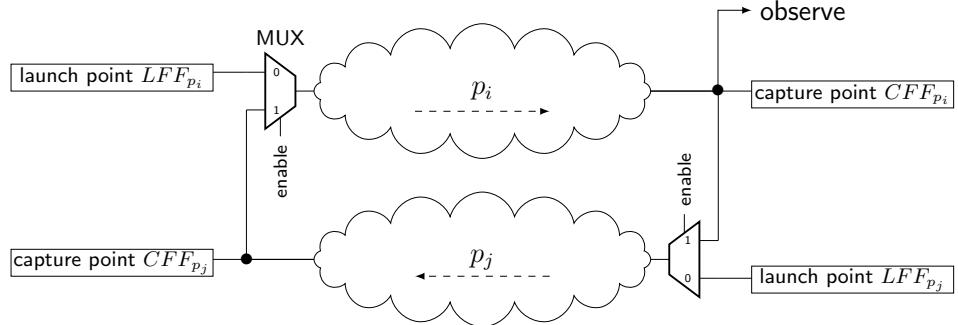


Figure 3.5.: Basic concept of the natural loop approach.

Path  $p_i$  is used as a forward line, Path  $p_j$  is in the opposite direction and is used as a return line. This eliminates the need for a large part of the feedback loop and the signal buffers of the feedback loop. Thus, paths with long feedback loops are the primary candidates for this approach. Note that the inverting behavior of the paths within the natural loops must be ensured by additional inverters or a suitable selection of the paths.

The natural loop functional path RO functionality in Figure 3.5 can be described as follows. The circuitry is in functional operation mode as long as the MUX switches are OFF (0). Once the MUXes are switched ON (1) and both paths are sensitized, the oscillation will start through both paths.

A prerequisite for this concept is that the ATPG tool can sensitize the connected paths with one path delay pattern. A modern ATPG tool tries to sensitize as many paths as possible with as few patterns as possible. Thus, one path delay pattern can be used to sensitize many paths. The patterns used for the natural loop approach must be able to sensitize at least two paths.

All coordinates of each cell are known for the paths which are sensitized by a pattern. Thus, the start- and end-points of each path are known. Also, the physical length of the path can be calculated, given those coordinates. The coordinates of the start- and end-point and the physical path length are fed into an optimization algorithm. Based on the coordinate and length information, the algorithm decides whether the paths should be combined into a natural loop or remain independent functional path ROs.

A criterion of the algorithm to select the natural loops is the overall path length and the expected length of the feedback loop. If both lengths are below a certain threshold value, the paths are not considered as natural loops. The threshold value needs to be set individually for an MCU design. Another criterion is the distance from the end-point of one path to the start-point of another path. If there are paths that run in the opposite direction and have their start- and end- points nearby, they are excellent candidates for the natural loop approach.

Paths that have such properties are usually unidirectional buses. These buses run over a long distance, are in opposite directions and arranged in parallel.

Whereas Section 3.2 focuses on concepts using topological properties of the functional path to reduce the routing and additional buffers, the following section presents the *self-enabling* approach.

### 3.3. Self Enabling

The self-enabling approach targets reducing the routing of the enable and observe signal lines for the functional path ROs. The prerequisite of the functional paths for such an approach is that they have to be in  $P$  - which implies the functional paths must be sensitizable with an appropriate path delay pattern. The implementations of the functional path RO published so far [40, 31, 21] require a separate enable signal that activates the oscillation. This enable signal must be generated and routed individually for each RO, which causes considerable effort, especially when implementing many functional path ROs.

The *self-enabling* approach eliminates the need for the individual enable signal for the functional path ROs. The self-enabling approach activates the RO using the appropriate circuits and DFT methodology. The basic principle of a self-enabling functional path RO is shown in Figure 3.6, where the select pin of the MUX is connected to the corresponding launch FF  $LFF_{p_i}$  of the path  $p_i$ . Thus, the MUX is activated by the corresponding  $LFF_{p_i}$ . In other words, the functional path RO enables itself.

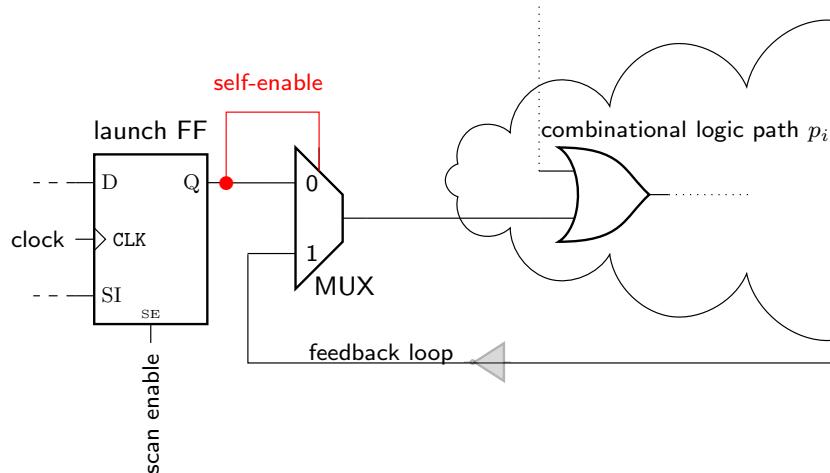


Figure 3.6.: The basic principle of the self-enabling approach of a functional path RO. Adapted from [41] © IEEE 2022.

Each path  $p_i$  is sensitized with a corresponding path delay scan pattern  $v_{p_i}$ . When the scan

pattern  $v_{p_i}$  is applied, all side inputs of  $p_i$  are at a stable non-controlling value. Furthermore, it is assumed that all **FFs** within the circuit are scan **FFs**. Consequently, all **FFs** are controllable with the **DfT** scan environment.

The launch **FF** of path  $p_i$  is called  $LFF_{p_i}$ .  $LFF_{p_i}$  enables the corresponding path  $p_i$ . An **ATPG** tool is used to control  $LFF_{p_i}$ . In commercial **ATPG** tools, scan **FFs** can be restricted to a specific value  $\{0, 1, X\}$  during the scan pattern generation. Those restrictions are called **ATPG constraints**. The **ATPG** constraints for a specific path  $p_i$  are  $c_{p_i}$ . This particular  $c_{p_i}$  is a tuple of  $\{0, 1\}$  constraints for all  $LFF_{p_i}$  that launch paths in  $P$ .

The  $LFF_{p_i}$  of  $p_i$  must be constrained to 1, while all other launch **FFs** of the remaining paths must be constrained to 0. The constraint ensures that only the selected path is activated.

The constraint can be formulated as follows:

$$c_{p_i} := (c_{p_{i_1}}, c_{p_{i_2}}, \dots, c_{p_{i_n}})$$

$$\text{with } c_{p_{ij}} := \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise.} \end{cases}$$

Suppose  $P$  consists of  $n = 100$  paths  $P = \{p_1, \dots, p_{100}\}$ ; when checking self-enabling for a particular path, e.g.,  $p_3$ , the launch **FF**  $LFF_{p_3}$  must be restricted with **ATPG** constraint 1. The other 99 paths  $\{p_i | p_i \in P \setminus \{p_3\}, i = \{1, \dots, 100\}\}$  with the corresponding launch **FFs** must be constrained with **ATPG** constraint 0. Thus,  $c_{p_3}$  must look like the following:

$$c_{p_3} = (0, 0, 1, 0, 0, \dots, 0).$$

The constraint tuple  $c_{p_3}$  is allocated to the **ATPG** tool before the path delay pattern generation. The tool attempts to generate a new path delay pattern  $\widetilde{v}_{p_3}$  given the **ATPG** constraints. If the tool can generate a suitable scan pattern  $\widetilde{v}_{p_3}$  for the path  $p_3$  given  $c_{p_3}$ ,  $p_3$  is included in the new subset  $\widetilde{P}$  containing all paths that can be used with the self-enabling approach.

In case the **ATPG** tool cannot generate a pattern for a path considering the **ATPG** constraints, the path is rejected. The check is performed for all paths in  $P$ . If a path is rejected, it will automatically end up in  $\widetilde{P}$ .

The implementation of the basic concept in a large **MCU** in the automotive design flow requires a standardized approach. Therefore, a new library gate, the so-called **RO-MUX**, is introduced. This is necessary in order to implement all self-enabling functional path **ROs** automatically and uniformly. The **RO-MUX** is shown in Figure 3.7.

The **RO-MUX** itself has seven ports, an *IN* and *OUT* port, a *local enable*, *general enable*, *feedback*, *feedback*, and *observe* port. Included in the **RO-MUX** are a 2-to-1 **MUX**, 2 AND gates and an inverter. Note that the **RO-MUX** is unbundled into the individual gates after

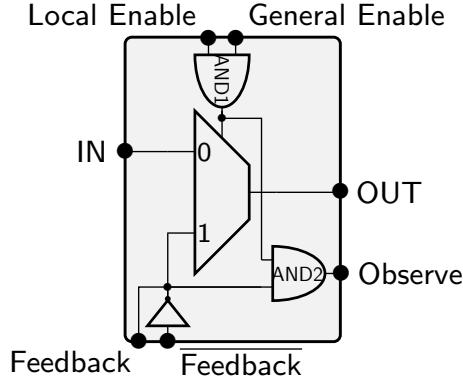


Figure 3.7.: The library gate called RO-MUX. Adapted from [49] © IEEE 2023.

the implementation. This unbundling prevents the generation of new error states in more complex gates that may be difficult to test from DFT perspective.

The RO-MUX is implemented between the launch  $LFF_{p_i}$  and the first gate of the path  $p_i$  to create a functional path RO. The  $IN$  port is connected to the  $LFF_{p_i}$  and the  $OUT$  port is connected to the first gate of  $p_i$ . If  $p_i$  itself is inverting, the endpoint of the path is connected to the *feedback* port; if not, the *feedback* port is chosen. The *local enable* is connected to the output of  $LFF_{p_i}$ .

Furthermore, the activation of the functional path ROs during functional mode of the MCU must be prevented. A *general enable* signal to unlock the oscillation mode, similar to the scan enable signal, ensures this. This signal is set by a protected bit and can be further gated in order to provide *freedom from interference*, which is necessary for safety-critical applications [99]. One signal is used for all functional path ROs on the chip.

The AND1 gate inside of the RO-MUX ensures that the RO-MUX is enabled if and only if *local enable* and *general enable* are active. Then, the MUX switches from functional mode (0) to oscillation mode (1), and the RO oscillates. Parallelly, the AND1 output controls also the observe signal via the upper input of AND2.

If the MUX is in oscillation mode, the upper input of the AND2 gate is on a non-controlling value and, therefore, transparent. Thus the oscillation frequency is passed from the feedback port directly to the observe port. The AND2 prevents the observe signal from uncontrolled toggling during the functional mode.

The observe signal, on which the frequency of the RO is measured, is spatially compacted with an XOR-tree. The AND2 in each RO-MUX ensures only one RO toggles during functional path RO test mode, which also reduces the cross talk and switching activity. The compaction of the observe signal by an XOR-tree can be seen in Figure 3.8.

The implementation of the XOR tree can easily be done as part of the implementation

### 3. Functional Path Ring Oscillators

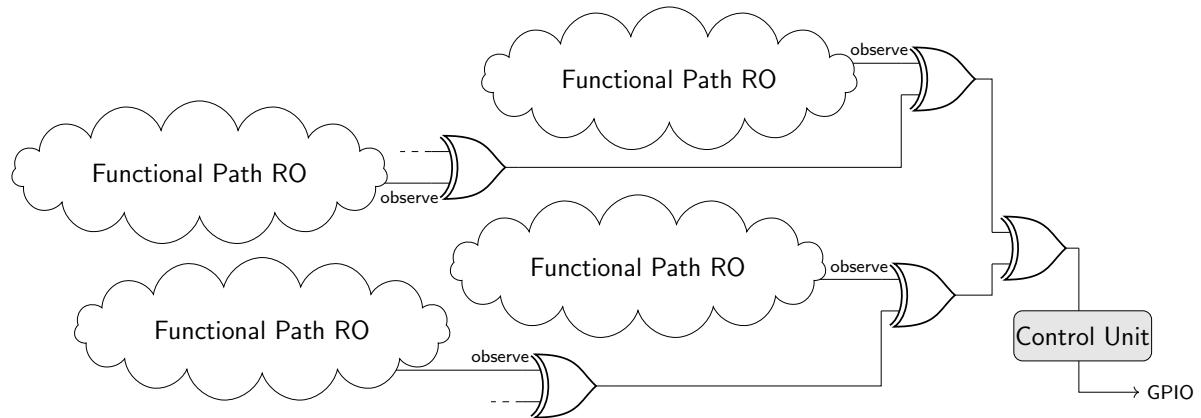


Figure 3.8.: XOR-tree for compacting the observe signals and forwarding it to a GPIO.  
 Adapted from [49] © IEEE 2023.

process of the functional path **ROs**. Thus, the XOR-tree is implemented efficiently and accurately by the **EDA** tools as a tailored solution.

Two options are proposed for implementing the concept of self-enabling- *Option 1* and *Option 2*.

### 3.3.1. Option 1 - the Direct Self-Enabling

Option 1 is also denoted as direct self-enabling because the local enable signal is connected to the launch **FF** of the corresponding path. The enabling of the path is accomplished with **ATPG** constraints, and the functional path has to be in set  $\tilde{P}$ . The detailed implementation of Option 1, is shown in Figure 3.9.

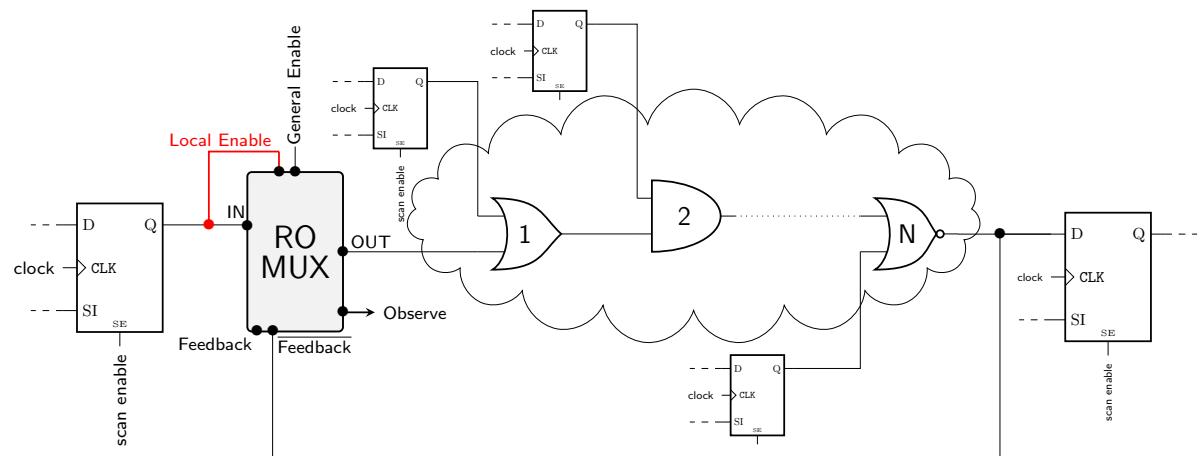


Figure 3.9.: Detailed implementation of Option 1. Adapted from [49] © IEEE 2023.

The implementation in Option 1 requires ATPG constraints. The ATPG constraints are

assigned prior to pattern generation. Therefore, the **ATPG** tool must consider these **ATPG** constraints for pattern generation, which can lead to limitations. If the **ATPG** tool has too many **ATPG** constraints, it may have difficulty generating enough path delay patterns because they constrain the **ATPG** tool too tightly. However, the limitation of the **ATPG** tool depends strongly on the **DFT** environment.

In addition, *Option 2* is proposed, which does not have the drawback of the **ATPG** constraints.

### 3.3.2. Option 2 - the Indirect Self-Enabling

The main advantage is that *Option 2* no longer uses the **ATPG** constraint. For this purpose, the local enable of the RO-MUX is no longer connected to the corresponding launch **FF**. However, a new combinational logic gate called *unlock gate* is introduced. The output of the unlock gate controls the local enable signal. The inputs of the unlock gate are connected to surrounding scan **FFs**. The general enable signal and all other circuitry functions are treated as in *Option 1*. The suitable paths for this option are in  $\hat{P} = P \setminus \tilde{P}$ . An elementary implementation is shown in Figure 3.10 where the unlock gate contains an AND gate connected to two surrounding **FFs**.

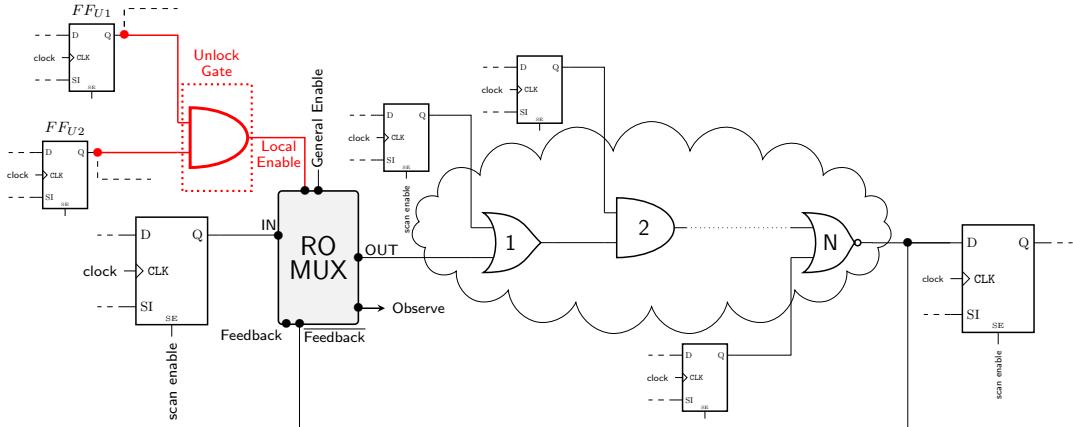


Figure 3.10.: Option 2 - the indirect self-enabling controlled by three surrounding scan **FFs**.

The scan pattern  $v_{p_i}$ , which sensitizes  $p_i$ , is deterministic and the values of all scan **FFs** are known for every  $v_{p_i}$ . If the functional path is sensitized with  $v_{p_i}$ , all scan **FFs** on the chip remain in a steady state since no clock pulse is triggered for the time being. Furthermore, there exists exactly only one pattern  $v_{p_i}$  that sensitizes  $p_i$ , thus, the combination of the assigned values of all scan **FFs** is unique.

In order to enable the RO-MUX with the local enable, the output of the unlock gate has to be 1. In the example from Figure 3.10, the two **FFs** (**FF<sub>U1</sub>** and **FF<sub>U2</sub>**) must have a 1 value, if  $v_{p_i}$

is applied. If  $FF_{U1}$  and  $FF_{U2}$  have a high value, the local enable is high, given that the unlock gate contains a two input AND gate. The  $FF_{U1}$  and  $FF_{U2}$  are two arbitrary scan FFs that are near the RO-MUX and have, by default, a high value, if  $v_{p_i}$  is assigned to the circuit. The fact that the FFs for unlock gate are selected with the knowledge that all scan FF values are known allows getting rid of the ATPG constraints. Therefore, the ATPG tool is not restricted by constraints during the pattern generation. The number of unlock FFs is not limited and any combination is allowed; only the unlock gate is designed accordingly.

The unlock gate consists of standard logic cells that are adapted to the corresponding use case for every path  $p_i \in \hat{P}$ . The combinational logic of the unlock gate and its complexity depend strongly on the design, the scan environment, and the number of ROs to be implemented. In the elementary case (Figure 3.10), the combinational logic within the unlock gate is an AND gate. However, the combinational logic could be even more complex. Thus, the level of protection against incidental enablement corresponds directly to the effort spent on the combinational logic of the unlock gate.

Theoretically, any sensitizable functional path ( $P$ ) can be implemented in *Option 2*; however, *Option 2* is much more complex and sophisticated. Thus it is a backup solution, if the ATPG tool struggles with the constraining for *Option 1*.

### 3.4. Control Infrastructure

Efficient implementation of the functional path RO is one factor; the second factor is ensuring an adequate control infrastructure. The control infrastructure is responsible for enabling the functional path ROs and forwarding the oscillation frequency to a general purpose input/output (GPIO) pad or counter structure. Furthermore, the control infrastructure should be scalable and must handle all implementation options.

There are three options for implementing the functional path RO on the design. Option 0 needs a distinct enable and observe signal (see Section 3.1). Option 1 and Option 2 are the self-enabling approaches that reduce the routing effort but need ATPG constraints (see Section 3.3.1), or the unlock gate (see Section 3.3.2). The properties of each option with its advantages and disadvantages are stated in Table 3.1.

Thus, all options have advantages for different use cases, so the control infrastructure should be suitable for all three options. Finally, the functional path RO oscillation frequency is the desired output independent of how the functional path ROs are implemented.

In order to demonstrate the control infrastructure with its functionality, first, a simplistic solution is presented that allows control of 8 functional path ROs implemented in Option 0. Such control infrastructure is shown in Figure 3.11.

The control infrastructure in Figure 3.11 consists of a scan-controllable register with three

### 3. Functional Path Ring Oscillators

Table 3.1.: Properties of Option 0, Option 1, and Option 2.

	Traditional enable		Self-enable	
	Option 0	Option 1	Option 2	Option 2
suitable path set	$P, \hat{P}$	$\tilde{P}$	$P, \hat{P}$	$P, \hat{P}$
RO-MUX	✗	✓	✓	✓
local enable signal	control infrastructure	Launch FF	Nearby FFs	
general enable signal	✗	control infrastructure	control infrastructure	
observe signal	control infrastructure	XOR- Tree	XOR- Tree	
Pro	easy to implement	reduced routing effort	high flexibility	
Cons	high routing effort	ATPG constraints	high complexity	

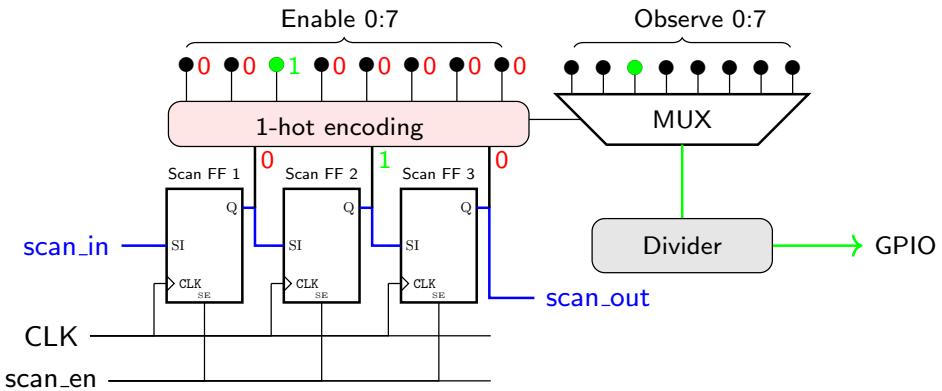


Figure 3.11.: Basic control infrastructure for the functional path ROs with 8 ports.

scan FFs. The scan FFs are connected in series and are part of a scan chain. Therefore, they can be controlled via the ATPG tool. The three scan FFs are generating a 3-bit binary coding. Subsequently, the 3-bit binary code of the scan FFs is translated into a one-hot-encoded selection bus. Thus, the 3-bit binary signal is converted to an 8-one-hot coded selection bus. The one-hot-encoded selection bus is responsible for enabling the respective MUX of the eight implemented functional path ROs. The same one-hot-encoded selection bus also maintains the control infrastructure's internal MUX, compressing the observe signal with the oscillation frequency from the selected functional path RO. The selected observe signal is then divided by a static internal divider stage and forwarded to the GPIO pin. The divider stage is necessary to divide the oscillation signal to a suitable frequency in which the GPIO can process it.

The third functional path RO is enabled in Figure 3.11. For this, the three scan FFs (FF 1, FF 2, FF 3) are constrained with the values 0, 1, 0. Thus the third functional path RO is activated due to the one-hot-encoding. The active ports are colored green in Figure 3.11. The constraining of the scan FF is accomplished in the path delay pattern generation for

### 3. Functional Path Ring Oscillators

the respective functional path **RO**. Thus, the control infrastructure's information on which functional path **RO** is active and maintained is contained in the respective scan pattern.

However, the presented example can facilitate only eight functional path **ROs** of Option 0. Therefore a more generalized solution is developed. In order to combine these three options, a *hybrid concept* is introduced. This hybrid concept is a control infrastructure that can handle all options, the traditional implementation (Option 0) as well as the self-enabling approaches (Option 1 and Option 2). The hybrid control infrastructure is shown in Figure 3.12.

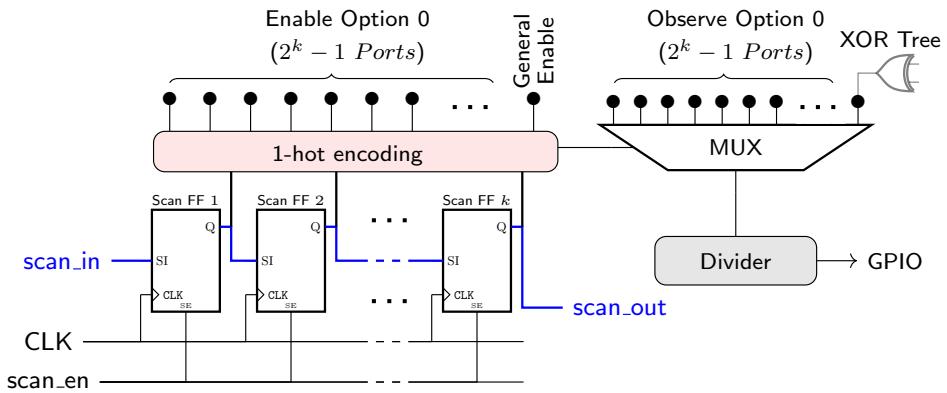


Figure 3.12.: The control infrastructure for the functional path ROs. Adapted from [49]  
© IEEE 2023.

The hybrid control infrastructure consists of a scan controllable register with  $k$  scan **FFs** and a one-hot encoded selection bus. This results in  $2^k$  one-hot encoded ports. Such ports control the enable signals for the functional path **ROs**, in particular for Option 0. In total  $2^k - 1$  enable and observe pins are available to be used for the functional path **ROs** implemented in Option 0. The general enable signal for the functional path **ROs** in Option 1 and Option 2 is generated on the last pin. Moreover, on the respective last observe pin, the root of the XOR-tree is connected.

Such a hybrid solution of the control unit is a scalable and efficient methodology to combine the implementation of Options 0, 1 and 2. If only Option 1 (or 2) is needed,  $k$  is set to 1, and the general enable signal and the XOR-tree are connected. An essential advantage is that the infrastructure is controllable with scan pattern, which means all settings are integrated into the scan pattern itself, which is very efficient in terms of test time.

The substantial advantage of the scan-controlled infrastructure is that the scan pattern that sensitizes one particular functional path **RO** also contains the information for the control infrastructure. There is no need for additional settings of test structures or a specific test mode; all that is needed is managed within a scan pattern.

### 3.5. Implementation Flow

The concept and technique of functional path **ROs** have been described in the preceding sections; this section focuses on implementing such structures on silicon in the industrial design flow.

The implementation of the functional path **ROs** in the design is done in two stages. First, the control infrastructure is implemented, and in the second stage, the functional path **ROs** are implemented. The control infrastructure is implemented in the **register-transfer level (RTL)** description in the early design phase. The scan-controlled **FFs** in the control infrastructure are part of the scan chains. Inserting them later would drastically affect the overall scan environment, which should be avoided. The enable and observe ports remain open and are not connected. The wiring of these ports will be done later.

The general requirements for the **MCU** to be developed are also defined in the early design phase. This means it is known which area, performance, and time specifications the **MCU** should have, and the control infrastructure can be adapted accordingly. The hybrid solution of the control infrastructure allows, for example, the use of a certain number of functional paths **RO** in Option 0 - this is predefined by the number of enable/observe ports. At the same time, the self-enabling approach allows the implementation of an additional, scalable number of functional path **ROs** in Option 1 or 2. Therefore the final number of implementable functional path **ROs** is very flexible and independent of the defined size of the control infrastructure.

**Example:** The control infrastructure consists of 5 scan **FFs** (32 one-hot encoded ports). Thus, 31 functional path **ROs** can be implemented in Option 0, and port number 32 is for the general enable signal for Option 1 or 2. This allows any scalable number of functional path **ROs** to be implemented in Option 1/2.

After the control infrastructure is described in **RTL**, the industrial design flow of an **MCU** continues until a late design phase - here, the second stage of the implementation starts.

The second stage is the implementation of the functional path **ROs** via an **engineering change order (ECO)**. Such implementation flow is shown in Figure 3.13.

The **ECO** implementation process starts in a late design phase after synthesis and place-and-route have been performed. At this stage, the paths of the design do not change considerably, which is an essential prerequisite for a smooth implementation process.

A **static timing analysis (STA)** tool analyzes the netlist and reports functional paths from the design. Such **STA** report contains all physical design information (launch **FF**, capture **FF**, all gates, coordinates of all gates, etc.) of the paths with the respective timing information.

The **STA** path report with all functional paths from the design is then pre-filtered and pre-processed. In the pre-filtering process, some functional paths that are unsuitable for the **RO** implementation are discarded. Such discarded paths are, for example, false paths. These

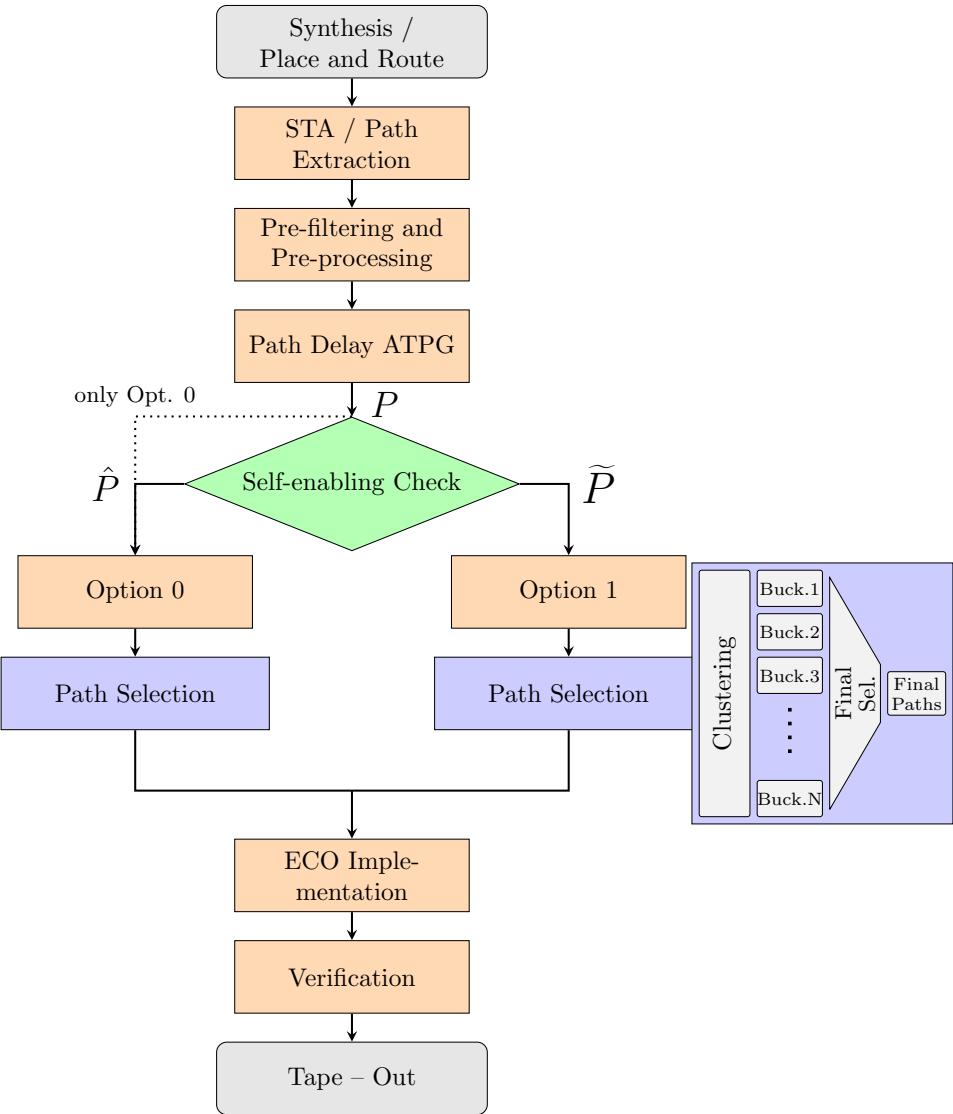


Figure 3.13.: Implementation flow of functional path RO. Adapted from [49] © IEEE 2023.

are paths that are not valid in functional mode but are included in the **STA** report. Other discarded functional paths have, for example, a short delay time. That would lead to a high oscillation frequency when implemented as **RO**. The measurement test setup can accurately measure only a specific frequency window of the oscillation frequency. Thus, all paths of which **RO** frequencies are outside the expected measurement window are discarded.

After that, the pre-filtered **STA** report is pre-processed. The path report is parsed and key properties of the paths are extracted, called *path characteristics*. For example, path characteristics from a path report with physical design data contain physical path length, cumulative cell driver strength, cell count, and other characteristics that are primarily used in the selection of the functional paths as **ROs** in the later implementation process.

The remaining functional paths after pre-filtering are passed to a commercial **ATPG** tool which is in path delay mode. The **ATPG** tool attempts to sensitize each functional path with a robust path delay pattern. Only a subset of the initially passed path list can be robustly sensitized. These functional paths then constitute the set  $P$ .

The following step is the self-enabling check, which checks the possible implementation of the functional paths in  $P$ . Paths that can be implemented with Option 1 are in the subset  $\tilde{P}$ , and paths that can be implemented with Option 0 or Option 2 are in  $\hat{P}$ .

However, the self-enabling check can be skipped if Option 0 (or Option 2) is the desired implementation method (dotted line in Figure 3.13). In this case, all paths in  $P$  are automatically assigned to the set  $\hat{P}$ .

Afterwards the selection process starts. The path selection process (see Section 4) works independently of the underlying set - whether  $\tilde{P}$  or  $\hat{P}$ . Finally, the selected functional paths and the options (Option 0, 1, or 2) used for the implementation are then provided to the physical implementation.

An **ECO** accomplishes the physical implementation, and an incremental compile run creates the functional path **ROs** in the netlist. The **ECO** script needs the start and end points of the path to create the feedback loop and the **MUX** or RO-MUX placement and, the option for implementation. For the implementation in Option 0 (or Option 2), the ports to be used in the control infrastructure for the observe and enable signal are needed. For implementation in Option 1, the RO-MUX is placed and connected, and the XOR-tree is built. Option 2 also requires the information for the unlock gate and its wiring. During the implementation of the functional path **ROs** all design guidelines within the **EDA** tool are considered; for example, additional signal buffers are placed on long routing lines to ensure a certain slew rate of the signals.

The entire information is then included in the **ECO** command script and the incremental compile run is executed. As a result, the netlist is modified accordingly and the selected functional path **ROs** are implemented in the design.

All **ECO** commands can be scripted in a **tool command language (TCL)** sequence to automate the implementation process. Thanks to this implementation process, the functional path **ROs** are placed in the industrial design environment with a push button solution. However, before the tape-out of the **MCU** is launched, the implemented **ROs** must be verified for their function and the correctness of the implementation, which is explained in Section 5.

Before coming to that, the results of this chapter are shown in the following.

## 3.6. Results

In order to evaluate the presented methodology of functional path ROs, the functional path ROs are implemented for this purpose in the context of the development of a new generation automotive **MCU**.

This section is divided into four subsections. The first subsection provides some background information on the automotive **MCU** test chip. Subsection 3.6.2 presents the general advantages of functional path ROs in terms of area and leakage. The following subsection provides a proof of concept for the advanced implementation concepts. The routing benefits of the self-enabling approach are presented in Subsection 3.6.4.

### 3.6.1. The Automotive Microcontroller Test Chip

The functional path ROs are implemented on a large automotive **MCU** in advanced **CMOS** technology. In order to cope with the complexity of such a large **MCU**, the **MCU** is divided into several modules with a hierarchical scan infrastructure [105]. Three dedicated modules were selected as test modules to implement the functional path RO. All modules of the **MCU** have their own scan infrastructure. The three selected modules were chosen to be fundamentally different in their size as well as their functionalities. Some basic information about the three selected modules can be seen in Table 3.2.

Table 3.2.: Basic information of the **MCU** modules.

Module	Scan FFs	Scan chains	Area	
A	414 803	5247	Large	
B	204 565	3072	Medium	
C	135 165	1452	Small	

The three modules are arranged in descending order according to their area and the size of the scan environment. Module A is the most extensive module, and Module C is the smallest. Module A and Module C contain combinatorial logic and have integrated CPUs,

while Module B has much integrated memory. The shape of the different modules also varies from oblong rectangular to square, which impacts the routing.

The functional path RO approach is benchmarked against traditional RO structures such as the **SMON** module (see Section 1.2.1).

### 3.6.2. Implementation Benefits

In order to demonstrate the implementation of functional path ROs, two functional paths were selected from the design and implemented as functional path ROs. In this section, Option 0 is used to illustrate the basic implementation concept of functional path ROs. One path (*Path a*) is a long path with a relatively large number of cells. The cumulative driver strength is also high. In contrast, *Path b* is very short, has a small number of cells, and the driver strength is approximately one-tenth compared with *Path a*. The path characteristics are shown in Table 3.3.

Table 3.3.: Path characteristics of *Path a* and *Path b*.

Demo Path	Cell Count	Cumulative Driver Strength	Path Length [μm]
<i>a</i>	27	2014	1459
<i>b</i>	12	220	90

The two paths were obtained from the set  $\hat{P}$  after pre-filtering and self-enabling check, as explained in Section 3.5. The implementation of paths *a* and *b* can be seen on the floor plan in Figure 3.14 and Figure 3.15.

The functional paths are marked in yellow in the floor plans. The **MUX** is located at the start point of each path, and the feedback loop (red) connects the end point of the path to the start point. The enable signal (light red) is routed from the control infrastructure to the **MUX** at the start point. The observation signal (red) is also connected to the feedback loop at an arbitrary location along the feedback loop and forwarded to the control infrastructure.

As can be seen in Figure 3.15, the routing from the functional path to the control infrastructure is one major drawback of the Option 0 implementation. Some signal buffers must be placed for such long routing lines, especially for the enable and observe signals. The signal buffers can cause a significant amount of cell contribution in modern **CMOS** technologies [106].

The timing impact on the functional path due to **MUX** insertion ranges from 2 % to 3 % delay increase for all considered functional path ROs. By selecting non-critical timing paths for **RO** integration of the functional path, the increased delay has no actual impact. Pre-

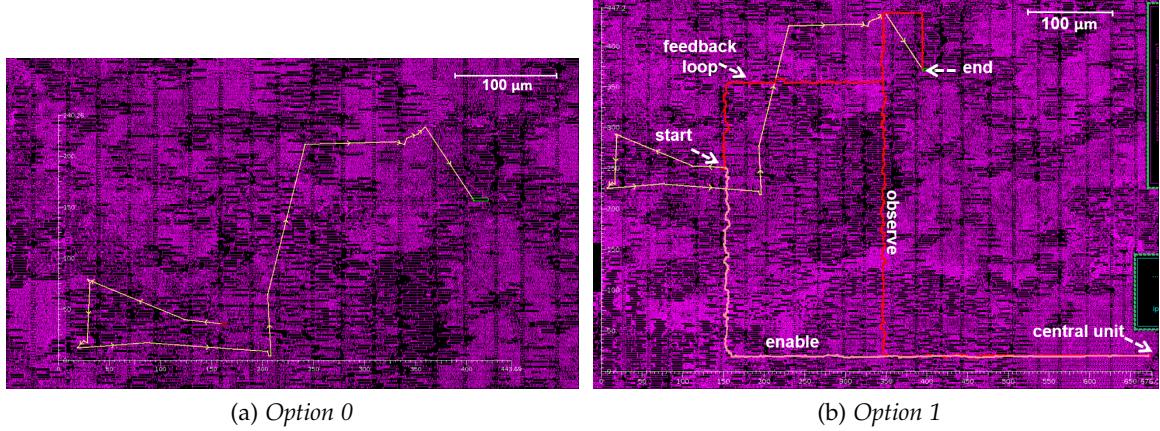


Figure 3.14.: Implementation on layout based on *Path a* after **ECO**. Adapted from [40] © IEEE 2021.

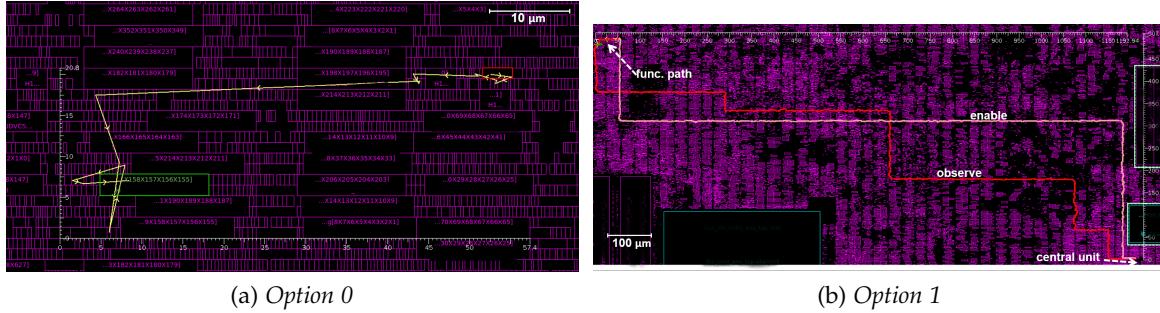


Figure 3.15.: Implementation on layout based on *Path b* after **ECO**. Adapted from [40] © IEEE 2021.

filtering in the implementation flow ensures that only those functional paths pass through the flow for implementation that are within a particular timing window to avoid generating timing violations by the later **MUX** insertion. The insertion of buffers on the feedback line also influences the **RO**'s absolute oscillation frequency. However, only the relative frequencies are needed to correlate the **RO** frequencies with the device performance.

The main advantage of the functional path ROs will be evident when comparing the cell area saved with functional path RO to conventional on-chip **RO** structures e.g. the **SMON** module (see Section 1.2.1).

The experiment was executed on Module C with a chip area of approximately 1.5 mm<sup>2</sup> and a leakage power consumption of 175 mW.

The impact of the physical implementation via the ECO on the chip is minimal. An additional **MUX** has an area impact of 1.2e–6 % in relation to the chip area under consideration.

The area impact of one buffer is 0.37e–6 %. The number of buffers to be placed depends on

the length of the feedback loop and the connection line to the observe signal. The longer the signal line, the more buffers must be used to ensure the corresponding transition slew rates. *Path a* has a long feedback loop that requires some buffers for transition edge slew rates, whereas *Path b* is compact and needs only one buffer for the decoupling from the functional path. The **EDA** tool by itself will care for sufficient slew rates and the buffering after **ECO** according to the design rules.

The control infrastructure is scalable to the desired number of functional path ROs to be needed. The estimated area overhead and leakage increase for the considered chip area is shown in Table 3.4.

Table 3.4.: Estimated area overhead and leakage increase of the control infrastructure.

No of ROs	Scan Bits	Area [%]	Leakage [%]
8	3	1.0e-5	0.18e-5
32	5	3.4e-5	0.63e-5
128	7	13.3e-5	2.4e-5
512	9	52.0e-5	9.5e-5

Assume there are 128 **ROs** to be implemented. This number is either implemented as functional path ROs or conventional **ROs** via an **SMON** module. Thus the **SMON** module is tailored to 128 contained **SMONs**, and the percentage amount of required area is calculated. The insertion effort of the **MUX** for the functional path ROs is also estimated. In addition, four buffers per functional path RO are inserted by default. This is a practical number based on trial implementation runs on the design under investigation; some **ROs** need only one, some **ROs** need more buffering, and the central control unit is also considered. The estimated area and leakage consumption in Module C is shown in Table 3.5.

Table 3.5.: Area overhead and leakage increase of 128 functional path ROs in comparison to an equal sized **SMON** module.

RO-Type	Area [%]	Leakage [%]
Functional Path ROs	0.091	0.39e-3
SMON module	2.4	10.1e-3
Savings	96.2	96.1

Thus, over 96 % of the area and leakage can be saved by implementing functional path ROs instead of using traditional approaches. Most of this area and leakage advantage is due to the fact that all the functional cells of the **RO** are already present in the design. Only the control infrastructure needs to be added.

### 3.6.3. Proof of concept of the Advanced Implementation Concepts

Concerning the advanced implementation methods, a proof of concept is being conducted to investigate whether and how many paths can be used for the methods from Section 3.2.

#### 3.6.3.1. Path Analysis Approach

The path analysis approach aims to find functional paths that have, by default, benefits if implemented as functional path ROs. For that, the design is checked to see if any functional paths can be allocated in four types (see Section 3.2.1).

An STA path report is extracted from each of the three modules of the MCU design and examined for the included path types, the results of which are shown in Table 3.6.

Table 3.6.: Path analysis analysing the structure of the functional paths.

	Module A	Module B	Module C
Pre-filtered Paths from STA report	27 374	7594	12 687
Type 1 - Self-contained paths	4	13	13
Type 2 - Parallel paths	0	0	0
Type 3 - same launch point	26 774	7425	12 553
Type 4 - same capture point	0	0	0
Independent paths	600	169	134
Unique launch FFs	1863	575	518
Unique capture FFs	27 374	7594	12 687
<hr/>			
After ATPG path sensitization			
All sensitizable paths ( $P$ )	2096	699	3860
Type 1 - Self-contained paths ( $P_{Type1}$ )	1	0	0
Type 2 - Parallel paths ( $P_{Type2}$ )	0	0	0
Type 3 - same launch point ( $P_{Type3}$ )	1970	619	3789
Type 4 - same capture point ( $P_{Type4}$ )	0	0	0
Independent paths	125	80	71
Unique launch FFs	306	220	193
Unique capture FFs	2096	699	3860

For example, if Module A is considered, most functional paths use multibit FFs, as can be seen from the number of unique starts FFs of 1863. Thus, a substantial proportion of paths start with the same launch FFs. However, all paths (27 374) use unique capture FFs. On the other hand, 600 paths are completely independent, which means those paths that use neither any shared start FFs nor the capture FFs. The same magnitudes of the number of paths related to the analyzed path types are also observed after path sensitization with the

ATPG tool. In the end, 1970 functional path ROs are of Type 3 and can be implemented as functional path ROs. If the three modules are compared, Type 3 is the only possibility to use the path analysis. Finally, such an analysis is highly dependent on the actual design.

### 3.6.3.2. Proof of Concept - Natural Loops

As explained in Section 3.2.2, the natural loops are the second approach of advanced implementation concepts. Also, a proof of concept is conducted on the three modules for such an approach.

The same pre-filtered STA report is used as in the section before, and the ATPG tool is used to sensitize the functional path ROs with a path delay pattern. One path delay pattern can sensitize multiple functional paths; the results are shown in Table 3.7.

Table 3.7.: ATPG results reveal the number of patterns necessary for path sensitization.

Module	Input Paths	Sensitizable Paths	Patterns
A	27 374	2096	529
B	7594	699	178
C	12 687	3860	1012

The input paths are the paths that are fed to the ATPG tool. The tool can only sensitize a subset of the paths. The number of sensitizable paths varies from below 10 % to 30 % in Module C. The number of generated patterns also fluctuates among the modules. Such numbers depend on the design as well as the scan environment.

Consequently, there are enough patterns that sensitize more than one path. Module A is chosen for a detailed consideration. Figure 3.16 shows how many paths are sensitized with one pattern in the three modules.

Accordingly, over 470 patterns can sensitize at least two paths. The same behavior can be seen in Module B and Module C.

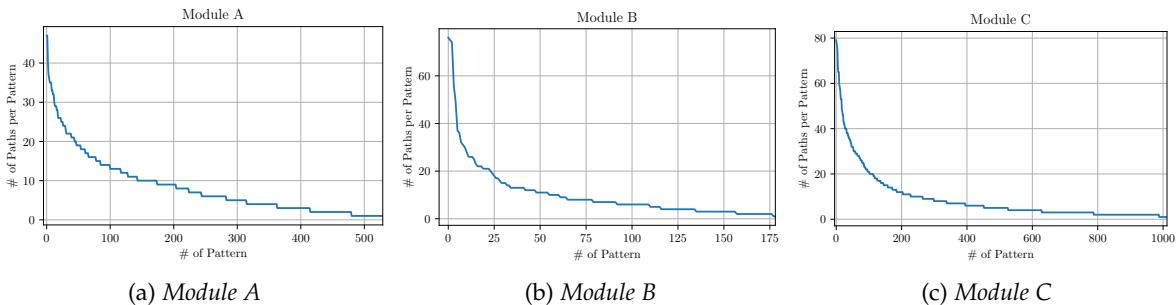


Figure 3.16.: Sensitizable paths per pattern for three modules.

### 3. Functional Path Ring Oscillators

In Figure 3.17a, two paths that are promising candidates for the natural loop approach are shown on the design in Module A. The paths run in opposite directions and can be implemented as a natural loop functional path RO. Figure 3.17b reveals how many paths are sensitizable with a particular pattern in Module B and how the paths are spatially distributed over the whole module.

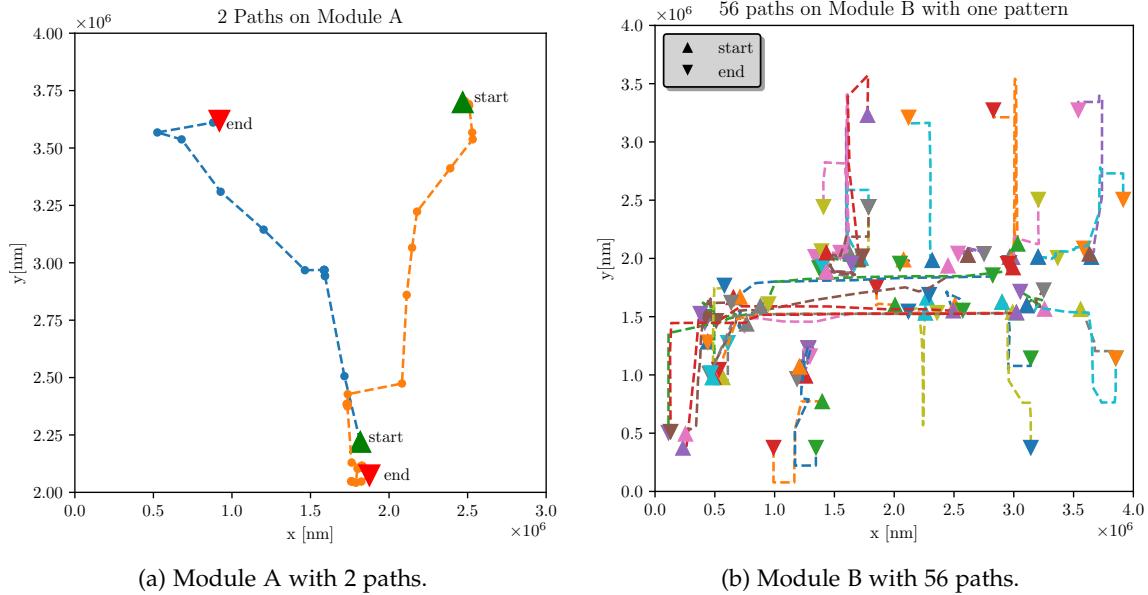


Figure 3.17.: Candidates for natural loops.

The focus of this section was to provide a proof of concept; the actual savings and routing are strongly dependent on where and which functional paths are chosen. Therefore, no attempt was made to quantify the methodology with exact numbers, as these are limited to the present design and cannot be generalized.

The following section presents the results of the self-enabling approach.

#### 3.6.4. Routing Benefits of the Self-enabling Approach

The advantage in terms of routing reduction is investigated among the three modules in this section. For that, the implementation in Option 0 (basic concept) and Option 1 (direct self-enabling) are compared. In addition, an estimation is given for a feasibility study for implementing Option 2.

### 3.6.4.1. Self-enabling Option 1

First, the self-enabling check is applied to determine how many paths are suitable for implementation in Option 1 - which requires [ATPG](#) constraints. Second, routing reduction estimation is performed to compare the implementations of Options 0 and 1.

An [STA](#) report is requested in a late design stage from each of the three modules. Then, the implementation flow starts and determines how many paths can be implemented in Option 0 and Option 1. The results are shown in Table 3.8.

Table 3.8.: Number of paths suitable for functional path RO implementation.

Set	Module A		Module B		Module C	
	Count	Percent	Count	Percent	Count	Percent
STA Input	27374	100 %	7594	100 %	12687	100 %
$P$	2096	7.66 %	699	9.20 %	3850	30.35 %
$\tilde{P}$	1362	4.98 %	698	9.19 %	3284	25.88 %
$\hat{P}$	734	2.68 %	1	0.01 %	566	4.46 %

The first row indicates the number of functional paths from the [STA](#). After the pre-filtering and the [ATPG](#) tool run in path delay mode, a subset of the functional paths can be sensitized. In the modules A and B, the [ATPG](#) tool can sensitize less than 10 %, while in Module C, the [ATPG](#) tool can sensitize over 30 %. However, for large automotive [MCUs](#), there are still enough paths left for [RO](#) implementation. The last two lines in Table 3.8 show the subset of functional paths that are suitable for the self-enabling approach ( $\tilde{P}$ ) and those that are only suitable for implementation in Option 0 ( $\hat{P}$ ). It can be observed that a large number of the paths in  $P$  are also suitable for the self-enabling approach. This shows that Option 1 is a feasible implementation approach for large [MCUs](#).

The number of paths in the subset for Option 1 basically depends on how the [ATPG](#) tool can handle the constraints; this, in turn, depends strongly on the [DFT](#) scan environment, the chip size, and the scan compression used. For the automotive [MCU](#) under investigation, Option 1 is the means of choice for routing critical modules.

The reduction in routing is then estimated to quantify the benefits between Option 0 and Option 1 in terms of routing. For estimating the routing effort, the *Manhattan Distance* is used, which is the sum of the absolute distance of the Cartesian coordinates. This distance is assumed to correspond to the worst-case scenario for routing.

There are eight paths selected from each of the three modules. The functional paths are chosen from the subsets  $\tilde{P}$  and implemented in Option 0 and Option 1. All paths are selected to be spatially distributed over each module. The schematic floorplans of the implementation in Option 0 are shown in Figure 3.18. The same functional paths are implemented in Option 1

in Figure 3.19

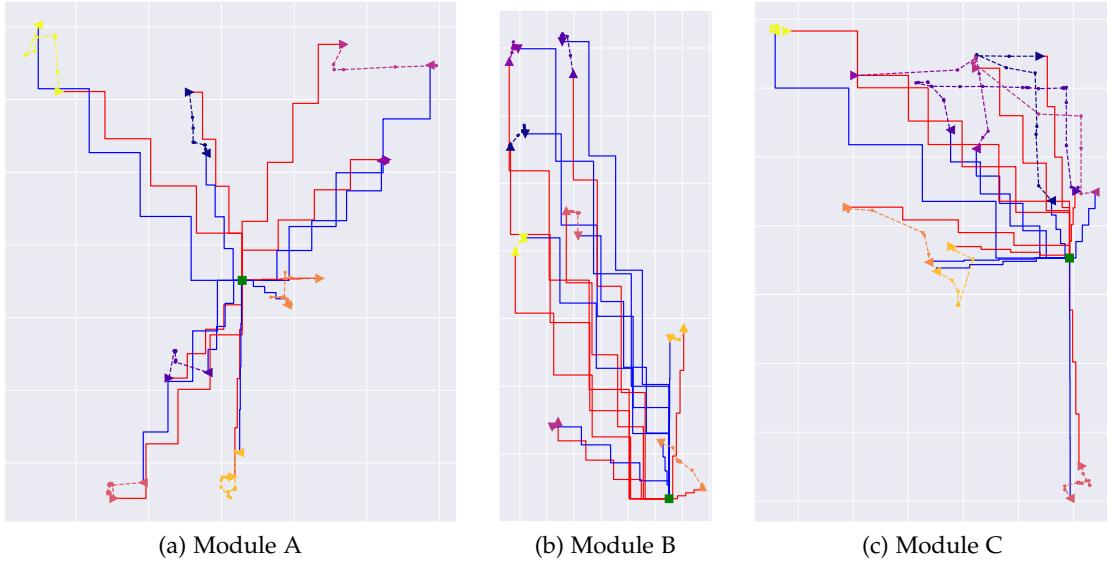


Figure 3.18.: Routing visualization of 8 sample paths implemented in Option 0.

In Figure 3.18 the enable and observe signals of the functional path ROs are routed to the respective control infrastructure. Each of the signals is routed individually. In contrast, the self-enabling approach Option 1 is used in Figure 3.19. As a result, the control signal becomes obsolete. In order to provide a fair comparison of the routing overhead, the XOR tree is also summarized at the control infrastructure. In this example, routing can be reduced by close to 70 % in Module A and C and over 77 % in Module B.

Routing reduction was simulated for all three modules with a larger number of functional path ROs in Option 1 compared with Option 0. The paths were chosen to ensure spatially distribution on the chip. The results are shown in Table 3.9.

Table 3.9.: Routing reduction of the RO implementation for *Option 1*.

Impl. ROs	Module A Reduction [%]	Module B Reduction [%]	Module C Reduction [%]
8	69.67	77.55	69.25
50	77.52	85.98	79.39
100	78.20	87.43	80.47
150	79.49	88.19	79.46

The routing reduction is more than 70 % across modules, in some cases even up to 80 %. Module B achieves the best results. The chip shape has a non-negligible influence in the

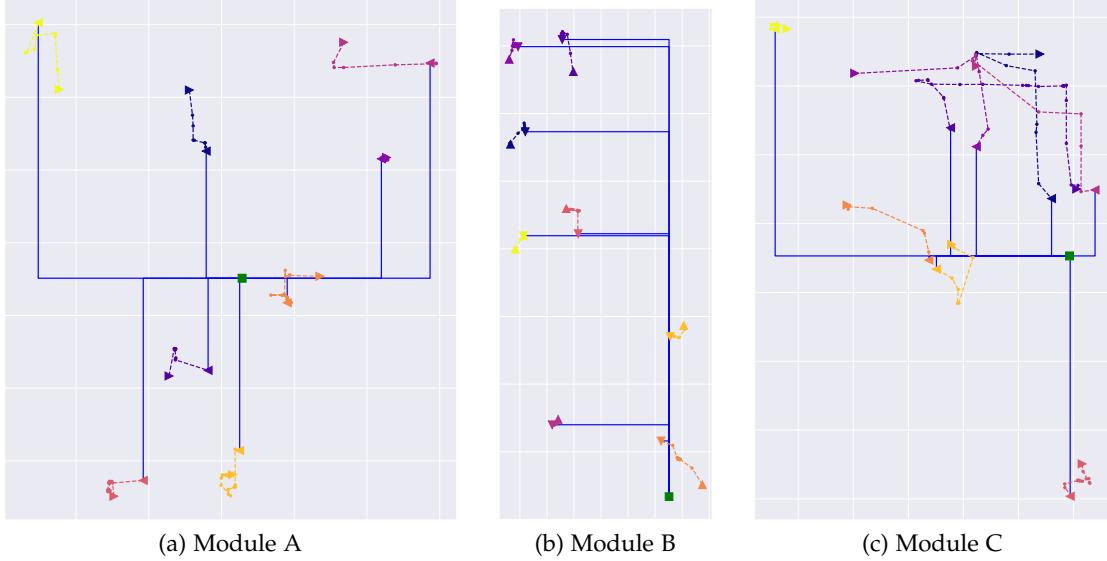


Figure 3.19.: Routing visualization of 8 sample paths implemented in Option 1.

analysis. Particularly in Module C, which had a stretched rectangular shape, the routing reduction declines again (1 percent point) in the case of 150 implemented paths. It is assumed that modern EDA tools can further reduce the routing overhead by optimizing the XOR tree for the observe signal.

The general enable signal was neglected in the previous routing simulation in Table 3.9. In order to estimate the routing overhead of the general enable signal, the dimensions of respective modules were considered. The general enable signal must expand over the entire module if all ROs are distributed over the entire module. This is simulated by calculating the diagonal of each module and adding three factors (1x, 1.5x, 2x) to the original routing reduction. For each factor, a separate calculation is done to see the impact of the general enable signal. The resulting overall routing reduction and the deviation from the previous estimate are shown in Table 3.10.

For a few implemented functional paths ROs (for example, 8 ROs), the general enable signal has a non-negligible influence. This is true for all three factors, especially for the twofold factor of the diagonal. However, if a more significant number of functional path ROs are implemented, the influence becomes negligible and is around one percent for 150 implemented ROs. Thus, the general enable signal is negligible for a high number of ROs, indicating the method's scalability.

As a result of the routing reduction due to the self-enabling approach, the cell utilization can be increased, thus reducing the overall area of the chip. In addition, a large part of the signal buffers required for long routing distances can be eliminated, saving area and leakage.

Table 3.10.: Routing reduction of the RO implementation for *Option 1* including the general enable signal.

Impl. ROs	1x diagonal			1.5x diagonal			2x diagonal		
	Mod. A	Mod. B	Mod. C	Mod. A	Mod. B	Mod. C	Mod. A	Mod. B	Mod. C
8	52.13	67.63	51.41	43.54	62.74	42.45	34.95	57.84	33.48
50	74.83	84.14	76.93	73.49	83.22	75.70	72.15	82.30	74.47
100	76.95	86.48	79.36	76.32	86.01	78.80	75.70	85.54	78.25
150	78.70	87.57	78.78	78.30	87.26	78.43	77.91	86.95	78.09
deviation				deviation			deviation		
8	17.18	9.79	17.92	25.77	14.69	26.88	34.36	19.59	35.85
50	2.69	1.84	2.46	4.03	2.76	3.69	5.37	3.68	4.92
100	1.25	0.95	1.11	1.87	1.42	1.67	2.50	1.90	2.22
150	0.79	0.62	0.69	1.19	0.93	1.03	1.59	1.24	1.38

Nevertheless, quantifiable numbers are difficult to determine, as this depends heavily on the design.

This section has revealed how powerful the self-enabling approach is using Option 1. Due to the high number of paths that can be implemented in Option 1, the implementation of Option 2 is obsolete for this large **MCU**. However, a short investigation is done for Option 2.

#### 3.6.4.2. Self-enabling Option 2

The previous section has shown that  $\tilde{P}$  contains relatively few paths (only one in Module B) compared with  $\hat{P}$  (see Table 3.8). Therefore this experiment is conducted to subset  $P$ . If Option 2 is implemented, the unlock gate is placed and connected to nearby scan **FFs**. The design is analyzed to clarify how many scan **FFs** are nearby the launch point of the functional path. The amount of functional paths with a scan **FF** closer to  $10\mu m$  is shown in Table 3.11.

 Table 3.11.: Paths with nearby **FFs** per module.

	Module A	Module B	Module C
$P$	2096	699	3860
$\geq 10\mu m$	1396	684	3458
$< 10\mu m$	700	15	402
$10\mu m$ w.r.t. diag.	0.22	0.28	0.40

Exactly 700 functional paths in Module A have **FFs** closer than  $10\mu m$  from the respective launch point. However, in Module B there are only 15 paths that have very close scan **FFs**

### 3. Functional Path Ring Oscillators

---

around them. The  $10\mu m$  is about 0.2 % - 0.4 % percent of the module diameter. Thus, it shows up also here that already, with three considered modules, no generally valid statement can be made concerning the general advantage of an option concerning another implementation option. Option 2 is feasible since adjacent FFs can be found, but as far as the actual routing savings are concerned, no statement can be made. As a general rule for the three modules of the large automotive MCU, Option 1 is the preferred solution, and Option 0 is suggested as a backup.

## 4. Path Selection Methodology

### 4.1. Selection Flow

This chapter deals with the selection of functional paths and introduces a methodology that uses physical design data in order to determine functional paths for performance screening.

An essential step in the implementation of the functional path RO is the selection of the appropriate paths that contribute to the performance prediction. This path selection is independent of the implementation option (Option 0, 1, or 2), since only the functional path itself is important. Path selection aims to find paths that represent the performance of the entire chip by trying to select a diverse set of paths on the **MCU**. The physical-aware path selection approach is suitable for any implementation concept and is revealed in Figure 4.1.

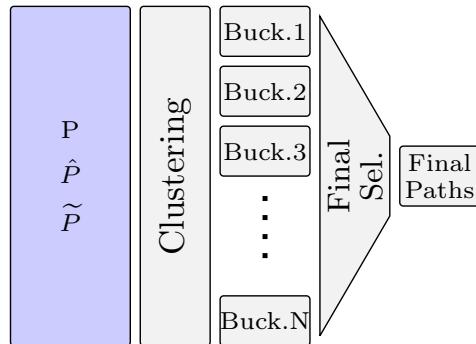


Figure 4.1.: Physical-aware functional path selection flow.

Path selection is independent of whether it is performed on the set  $P$  (for Option 0),  $\tilde{P}$  (for Option 1) or set  $\hat{P}$  (for Option 2). Therefore, assume that the set  $\mathcal{P}$  is a set of  $m$  sensitizable functional paths  $\mathcal{P} = \{p_1, \dots, p_m\}$  that could be a set from  $P$ ,  $\tilde{P}$ , or  $\hat{P}$ . For each of the functional paths in  $\mathcal{P}$  some characteristics  $ch_i$  are extracted during the preprocessing. The characteristics can be described by an  $l$ -dimensional vector  $ch_i = (ch_{i1}, ch_{i2}, \dots, ch_{il})^\top$  where  $ch_{ij}$  denotes the value of the  $j$ -th characteristic of the path  $p_i$ .

Characteristics extracted from a physical design data path report include, for example, cell count ( $ch_{i1}$ ), accumulated cell driver strength ( $ch_{i2}$ ), physical path length ( $ch_{i3}$ ), and other characteristics ( $ch_{i4}, \dots, ch_{il}$ ). Characteristics are defined using the designers' knowledge and previously published research [36, 37, 38, 88, 17].

#### 4. Path Selection Methodology

---

The *cell count* (characteristic  $ch_{i1}$ ) is the number of individual cells within the path. The cell count is calculated as follows

$$ch_{i1} = \sum_{k=1}^n \mathbb{1}_{g_k \in p_i} \quad (4.1)$$

where  $g_k$  represents the logic cell of the the design.

This characteristic is beneficial to cope with process variations. The process variation, in general, can be divided into **die-to-die (D2D)** and **within-die (WID)** variations (see Section 2.3.1). Whereas the **D2D** variation affects all die transistors, identically, the **WID** variation affects each transistor individually. Paths with low cell count are more sensitive to **WID** variation, whereas paths with high cell count balance the **WID** variation and are more sensitive to **D2D** variation [88]. Thus, both cases are needed to cope with the process variation in detail. In modern **CMOS** technologies, the **WID** becomes more prominent than in mature technologies [17].

The *accumulated cell driver strength*  $ch_{i2}$  is the sum of the strengths of each cell on the path, and can be expressed as

$$ch_{i2} = \sum_{k=1}^n P_{g_k \in p_i}. \quad (4.2)$$

Each logic cell has a certain driver strength  $P_{g_k}$ . The driver strength depends mainly on the load to be driven by the cell, i.e. how many other logic cells are being supplied by the output of the driving cell, which is also called fan-out. The higher the driver strength is, the more fan-out the cell has.

Considering multiple driver strengths is advantageous to monitor nonlinearities within the cell characteristics. Only some operating conditions are characterized in the cell characterization of the standard cell library. If an operating condition is needed between the characterized conditions, the cell behavior is interpolated based on the available data. Thus, potential nonlinearities are not captured. Considering paths with different driver strengths can thus reveal behaviors that are not evident in the characterization [37].

The *physical path length*  $ch_{i3}$  is the sum of the Euclidean distance from cell to cell within the path. The **STA** report contains all the coordinates of the cells along the path; therefore, the distance can be calculated as follows

$$ch_{i3} = \sum_{k=1}^n d_{g_k \in p_i}. \quad (4.3)$$

The  $d_{g_k}$  represents the Euclidean distance between two cells of the path. However, the calculated Euclidean distance does not exactly match the length of the routing lines on the chip. Nevertheless, this approach is well suited for path selection, as a mixture of short,

medium, and long paths should be covered. In addition, the cell coordinates and path length can be used to cover a reasonable spatial distribution of observed paths on the chip and to detect process variations.

The resulting matrix looks as follows

$$\begin{aligned} p_1 &\hat{=} ch_{11} \ ch_{12} \ \dots \ ch_{1l} \\ p_2 &\hat{=} ch_{21} \ ch_{22} \ \dots \ ch_{2l} \\ \vdots &\hat{=} \vdots \ \vdots \ \ddots \ \vdots \\ p_m &\hat{=} ch_{m1} \ ch_{m2} \ \dots \ ch_{ml}. \end{aligned}$$

In order to reduce the complexity and merge similar paths, the functional paths are clustered into so-called *buckets* according to the chosen path characteristics. Each functional path is assigned to a particular bucket. In total, there are  $N$  buckets  $\mathcal{B} = \{B_1, \dots, B_N\}$ .

There are several algorithms for this unsupervised clustering problem. The K-means algorithm is used, which is a distance-based unsupervised methodology [93]. The algorithm aims to group paths with similar path characteristics. This is a multivariate optimization problem.

The number of buckets is determined with the elbow criterion [94]. The K-means algorithm starts with only one bucket, and the [within-cluster-sum-of-squares \(WCSS\)](#) is calculated as a measure of within-cluster variance. In the case of  $N=1$ , the [WCSS](#) is the highest. The algorithm run is repeated by increasing the number of buckets, and the [WCSS](#) starts to decrease. The relation between the [WCSS](#) and the number of buckets results in a graph with an elbow shape. From a certain number of buckets, the [WCSS](#) drop is no longer as steep as before - such a point is called the elbow point. The number of buckets at this elbow point is the *K-means* clustering best choice.

Other cluster algorithms can also work fine, e.g., the density-based *DBSCAN* [95]; that depends mainly on the dataset and the algorithm's settings. For the distance-based clustering (e.g. K-means), it is essential to standardize the characteristics set previous to the clustering [107].

Each bucket contains paths that appear similar according to the specified characteristics. The main target of the cluster approach is not to have sharp disjunct buckets, which would be impossible due to the high correlation among the characteristics. The intention is instead to cluster structurally different paths; therefore, the specific choice of a particular algorithm (distance-based, density-based, or hierarchical-based) to perform the clustering has only a minor influence on the results.

The final selection of paths to be implemented as [ROs](#) in the design is made after clustering. There are three different ways of handling this.

One way is to randomly pick a path from each of the  $N$  buckets and select a final subset for the implementation. Such an approach is straightforward, and a subset of paths has, by

#### 4. Path Selection Methodology

---

default, a diverse mixture of path types because the paths are out of the different buckets.

The second way of selecting the final subset is a more sophisticated approach that involves the coordinates of the paths. This ensures that the ROs are spatially distributed, in addition to being from different buckets. For example, some paths from the same bucket are picked, but the paths are spatially distributed across the chip.

The third way takes a step back to clustering. The results of the elbow method are ignored, and the number of buckets is chosen according to how many functional path ROs are to be implemented. If it is known in advance that 32 ROs are to be implemented, for example, the number of buckets for clustering is also set to 32.

The final selection approach depends on the buckets' granularity and the number of functional path ROs to be implemented. Finally, the selected paths are then handed to the ECO flow, and the functional path ROs are implemented on the netlist.

## 4.2. Results

The path selection is executed on each of the three modules. The used set of the functional paths to demonstrate the path selection and clustering is  $\tilde{P}$ .

The characteristics are already extracted during the pre-processing. There are 10 characteristics extracted from each path. Thus, the matrix of Module A has the dimensionality of  $10 \times 1362$ , because the set contains 1362 paths (see Table 3.8).

On each of the three modules, the K-means cluster algorithm is started to group the paths in individual buckets. In order to determine the suitable number of buckets, the elbow method is used. The results are shown in Figure 4.2.

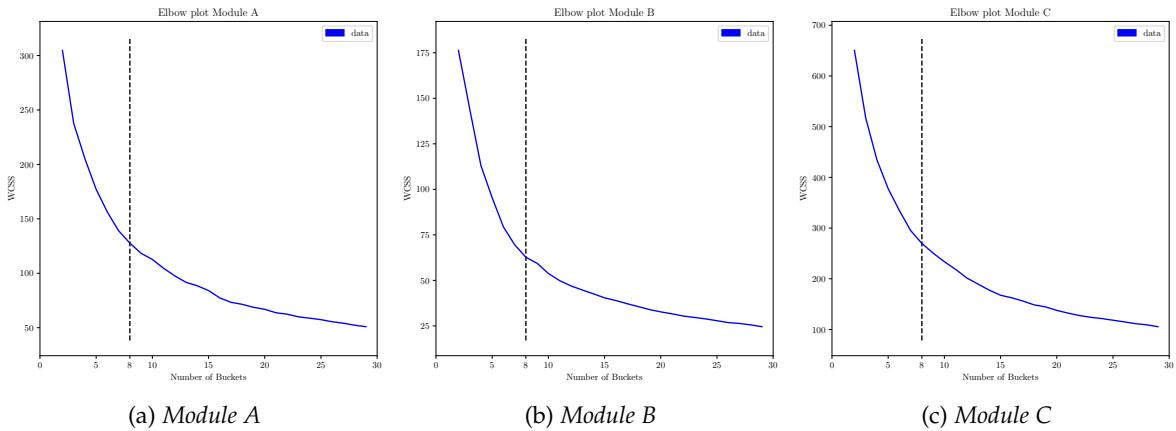


Figure 4.2.: Elbow plots of the three modules.

The elbow diagrams show how the WCSS behaves as the number of buckets increases. The

#### 4. Path Selection Methodology

sweet spot is marked in the diagrams with the dashed line, as the recommended number of buckets for clustering should be chosen. Coincidentally, in all three modules, the number derived from the graph is 8. Thus,  $N$  is set to 8, and K-means clustering is performed on every module.

The functional paths of each module are grouped into 8 buckets. The number of functional paths per bucket varies a lot, as shown in Table 4.1.

Table 4.1.: Number of paths per bucket after the K-means clustering.

Bucket	Module A		Module B		Module C	
	Count	%	Count	%	Count	%
0	386	28.3	186	26.7	949	29.0
1	179	13.1	148	21.2	615	18.8
2	160	11.7	95	13.6	549	16.7
3	153	11.2	81	11.6	546	16.6
4	149	10.9	73	10.3	184	5.6
5	117	8.6	46	6.6	182	5.5
6	111	8.1	35	5.0	160	4.8
7	107	7.9	34	4.9	99	3.0
total	1362	100	698	100	3284	100

The three modules' largest buckets contain almost 30 % of the respective functional paths. In contrast, some buckets contain just 3 % of the respective paths, for example, in Module C.

In order to present the contribution of the characteristics,  $(ch_{i1})$ ,  $(ch_{i2})$ , and  $(ch_{i3})$  are plotted for the different buckets in Module B. The plots are shown in Figure 4.3.

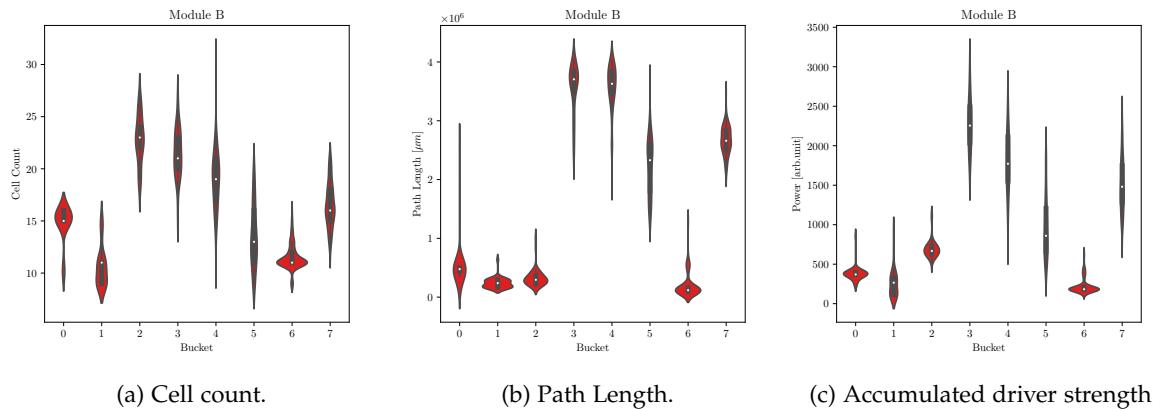


Figure 4.3.: Violin plots presenting the distribution of paths on Module B according to the defined path characteristics.

The violin diagrams show that the individual buckets differ. On the X-axis, the 8 buckets are plotted. In Figure 4.3a, the number of cells in the different 8 buckets is plotted on the Y-axis, and the length of the path is plotted in Figure 4.3b. In Figure 4.3c the accumulated driver strength is plotted on the Y-axis. Looking at buckets 2 and 3, both have nearly identical cell counts. Bucket 2, however, contains very short paths, while bucket 3 contains long paths also with respect to the driver strength; the two buckets differ significantly. Such differences can be observed for all characteristics.

From each of the buckets, the final paths can be selected and implemented as ROs via the ECO. There are three selection methods proposed in the previous section.

#### 4.2.1. The final Selection for the Test Chip

The presented selection flow is part of the overall implementation flow, as shown in Figure 3.13. Up to 8 functional path ROs shall be implemented on the three Modules. Option 0 is chosen as the implementation option. The implementation flow is started, and the path selection is executed. The final selected - and implemented - functional paths are shown in Table 4.2.

The implemented functional path ROs show a large diversity regarding the defined characteristics. The 22 functional path ROs are implemented in the design, and the MCU is fabricated.

The entire process is fully automated and has a low turn-around time. Less than 24 h elapse from the generation of STA reports to the incremental compilation run. Thus, a smooth design process can be guaranteed, and the tape-out is not delayed.

---

4. Path Selection Methodology

---

Table 4.2.: Final selected functional path to be implemented as an RO.

RO	Module	Bucket	Cell Count	Driver Strength	Path Length [μm]
0	A	A-3	16	560	1145
1	A	A-7	9	315	478
2	A	A-5	26	1322	1383
3	A	A-3	20	898	900
4	A	A-6	12	527	1007
5	A	A-0	35	1759	844
6	A	A-1	19	627	467
7	A	A-4	30	989	345
8	B	B-3	20	2030	2726
9	B	B-0	19	1089	905
10	B	B-2	26	2970	3847
11	B	B-3	21	2320	3749
12	B	B-7	9	140	142
13	B	B-1	20	1200	846
14	B	B-6	20	510	565
15	C	C-3	19	468	301
16	C	C-2	9	470	1280
17	C	C-0	11	307	475
18	C	C-5	27	1965	1251
19	C	C-6	10	370	348
20	C	C-7	16	318	147
21	C	C-1	31	2620	1062

---

## 5. PRE-Silicon Verification and Validation

This section presents the last steps in the [Pre-Silicon](#) part. The functional path ROs are selected and implemented via an [ECO](#). Figure 5.1 presents an overview of this section and at which point in time the described methodology fits in the [MCU](#) development flow.

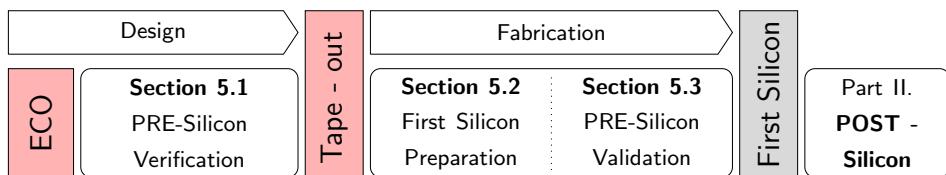


Figure 5.1.: Overview of the Section [Pre-Silicon](#) verification and validation.

### 5.1. PRE-Silicon Verification of the Functional Path ROs

The flow described below is one way of verifying the functional paths. Still, it is also possible to create a different verification flow compatible with the given design flow.

The [ECO](#) is executed in an incremental compilation run and the functional path ROs have been implemented in the design. This compilation run results in a new netlist that contains the [ECO](#) changes. This new netlist must be verified to ensure that the compile run does not affect the functionality of the entire [MCU](#). In addition, it is necessary to check that the functional path ROs are implemented and working correctly.

The [Logic Equivalence Check \(LEC\)](#) is used to check the functionality of the [MCU](#). The [LEC](#) compares the given netlist with the [RTL](#) representation of the *Golden Reference*, the latest [RTL](#) design. The netlist must be logically equivalent to the *Golden Reference RTL* design throughout the physical design flow. A successful [LEC](#) is essential for the incremental compilation runs associated with [ECOs](#) performed late in the design phase [55]. A successful [LEC](#) ensures that the implemented functional path ROs do not influence the functionality of the [MCU](#).

Once the [LEC](#) is successful, the verification of the functional path ROs starts. The implemented functional path ROs are checked by gate-level timing simulation. For such gate-level simulation, the netlist after incremental compile run is needed, and the functional path RO path delay patterns  $v_{p_i}$  are required for the simulation. The netlist must contain the timing

information for the oscillation to work properly; otherwise, no oscillation occurs because the feedback loop is a short circuit. One such netlist format is the back-annotated [standard delay format \(SDF\)](#). That [SDF](#) file allows accurate timing simulation.

The test sequence of a functional path RO with the respective modified path delay pattern is shown schematically in Figure 3.3. The gate-level simulation with the [SDF](#) file and the functional path RO pattern follows this test sequence. In the measurement phase, the actual oscillation frequency occurs due to the timing back-annotation of the [SDF](#) File. Thus, the simulation has two aspects: (i) the functional behavior of the functional path ROs is verified, and (ii) the expected oscillation frequency is simulated and stored as initial silicon limits for production test.

### 5.1.1. Functional Verification

Figure 5.2 shows a section of the gate-level simulation. The end of the shift-in phase and the start of the measurement phase are shown.

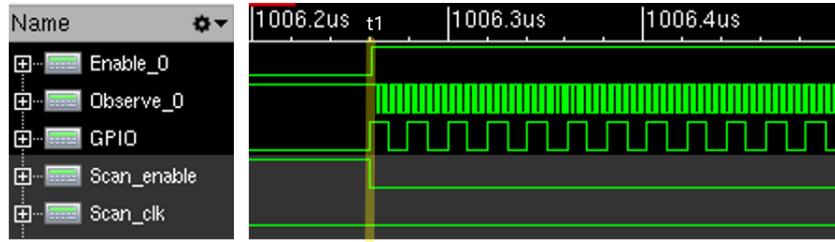


Figure 5.2.: Digital simulation snapshot at the beginning of the oscillation. Adapted from [49]  
 © IEEE 2023.

At the beginning of the simulation window shown, all signals are stationary, the shift-in phase is completed, and the functional path RO is sensitized. The upper signal shows the enable signal that triggers the oscillation. Below that is the observe signal, which is stabilized to allow the RO to oscillate once activated. The next signal shows the [GPIO](#) pin as measured by the [ATE](#). The scan enable is on a high state since the [MCU](#) is in the shift-in phase. The clock signal is shown for the sake of completeness.

At time  $t_1$ , the measurement phase begins, and oscillation is activated. The scan enable is deactivated, and the enable signal switches the [MUX](#) instantaneously to the oscillation mode. Thus, the observe signal follows the oscillation frequency of the functional path RO. A divider is inserted between the observe signal and the signal from the [GPIO](#). Thus, the divided oscillation signal is received at the [GPIO](#) pin. The first few oscillation periods cannot be used to determine the oscillation frequency due to the initial random state of the divider and transient effects. At  $t_1$ , the divider has an instantaneous high state, and the first pulses

## 5. PRE-Silicon Verification and Validation

---

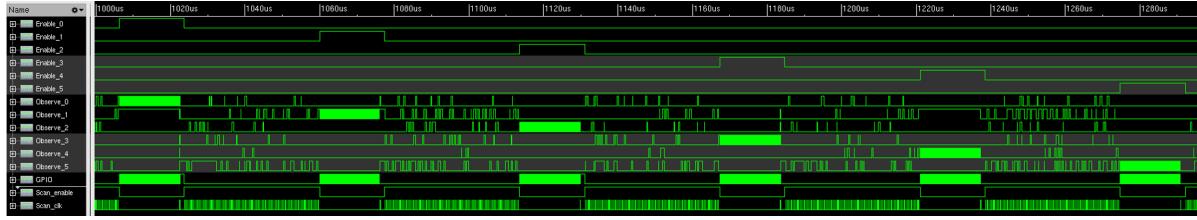


Figure 5.3.: Digital simulation sequence with 6 functional path ROs.

are absorbed until the divider's shift register operates properly. However, such transient effects can be masked, and the frequency at the [GPIO](#) pin is measured after a few hundred nanoseconds. The oscillation itself is independent of any clock event.

Verifying each functional path RO by an individual simulation run is inefficient. Therefore, the functional path RO patterns are merged into one large pattern. That large pattern is used for the verification and in the production test. The functional path ROs are sequentially activated and measured. A snapshot of the gate-level simulation is shown in Figure 5.3, where six functional path ROs are sequentially activated.

The first six signals are the enable signals for the functional path ROs; below them are the observe signals. The [GPIO](#) pin, the scan enable, and the clock signal are unique to the chip and are therefore used for all functional path ROs.

In the simulation run, the six functional path ROs are activated sequentially. The simulation starts with the shift-in phase to sensitize the first functional path RO. The clock signal has high activity in the shift-in phase; thus, all scan-FFs are loaded with the respective values. Then the oscillation starts, and the frequency is measured. At the end of the oscillation, two small clock pulses can be seen. These clock pulses are the launch and capture at-speed pulses from the path delay pattern. After the at-speed pulses, the shift-out phase starts. Since the individual patterns are merged, the first pattern's end is the start of the second pattern. Thus after the shift-out phase, there is a smooth transition into the shift-in phase of the second pattern. After the shift-in phase of the second pattern, the second functional path RO oscillates and is measured. This procedure is continued until all functional path ROs are tested. The verification is successful when all functional path ROs inserted via the [ECO](#) have been tested and oscillated. Thus, the functional path ROs are correctly implemented, and the design is released for tape-out.

### 5.1.2. Test-limit Extraction

As mentioned, the gate-level simulation has an additional benefit. The measured oscillation frequencies of the functional path ROs within the simulation run are stored for the production test. Thus, the stored simulation values act as the first test limits for the production.

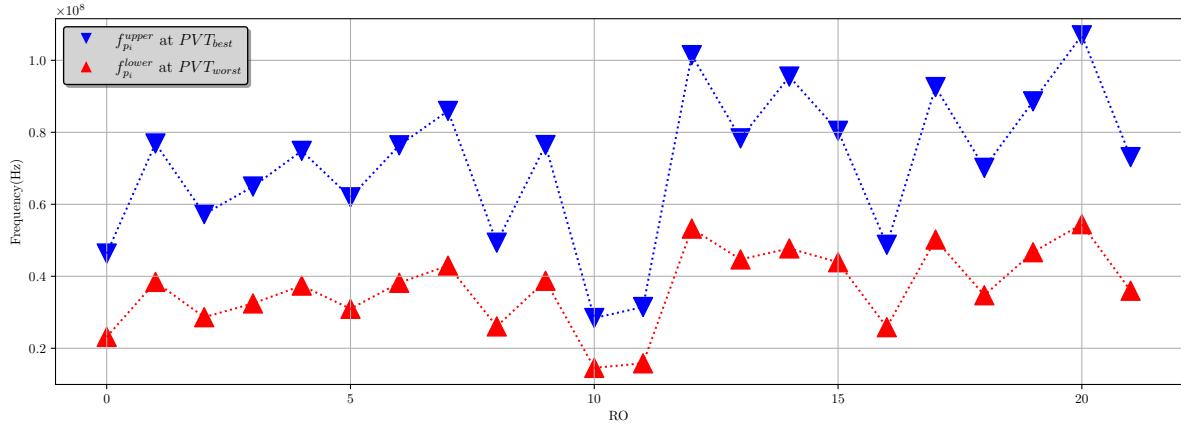


Figure 5.4.: Limits from the digital simulation using the worst and best case SDF.

In order to be more accurate, there is more than one SDF file. This is because the timing of the MCU depends on the PVT conditions. There is one SDF file for each PVT condition. Thus, each simulation run with the different SDF files gives a different result because the timing behavior in each PVT condition is different. The worst-case PVT ( $PVT_{worst}$ ) and best-case PVT ( $PVT_{best}$ ) conditions are used to determine the lower  $f_{p_i}^{lower}$  and upper  $f_{p_i}^{upper}$  test limits for each functional path RO. The best case represents a fast MCU under the best voltage and temperature conditions, resulting in the highest achievable oscillation frequency. Slow MCUs under the worst voltage and temperature conditions result in a low oscillation frequency. Two simulation runs are started with two different SDF files - the worst-case and best-case conditions, and the oscillation frequencies are stored as lower and upper test limits; the results are shown in Figure 5.4.

The 22 implemented functional path ROs are plotted on the X-axis, and the simulated frequencies  $f_{p_i}^{upper}$  and  $f_{p_i}^{lower}$  for each RO are plotted on the Y-axis. The expected frequency range varies from 10 MHz at  $f_{p_{10}}^{lower}$  up to 110 MHz at  $f_{p_{20}}^{upper}$ . The lower  $f_{p_i}^{lower}$  and upper  $f_{p_i}^{upper}$  bounds are considered during performance measurement on silicon.

## 5.2. SI Bring-up Preparation

### 5.2.1. Changes previous to Tape-out

As described in Section 2.1, where the overall digital MCU development flow is explained, the ECO phase is the last opportunity for changes in the design. Parallel to the functional path RO implementation in the ECO phase, there are plenty of checks and last-minute fixes during this phase - especially for large designs. For example, timing fixes, design rule checks, and IR issues are fixed [54].

In productive design flow, such **ECO** runs are grouped and executed together, and some iteration **ECO** runs are needed if some parts of a design need special treatment. Such large automotive **MCUs** are developed with a large team of engineers where everyone is responsible for only a tiny design part or issue. Therefore, as a matter of fact, implementing the functional path ROs as the very last step before tape-out is difficult. As a result, the design might change at the last minute when the functional path RO implementation is already done.

In order to check the influence of implementing the functional path ROs before the tape out of the **MCU**, the associated changes are reviewed. The functional path ROs are implemented in the **ECO** phase, where also other **ECOs** are executed. Therefore, the extracted path characteristics are compared pre- and post-tape-out. The clustering and path selection were executed based on the path characteristics of the pre-tape-out databases. The post-tape-out data represents the actual state of the chip, including changes made to the design in the late **ECOs**.

The logical composition of the gates in the path was protected after implementation, so everything remained the same in the basic logical composition of the path; otherwise, it might not be possible to sensitize the path. However, it is allowed to insert buffers, make cell swaps (concerning driver strength and transistor type), or make other optimizations that do not change the logical behavior.

The graphical representation of 3 path characteristics is shown in Figure 5.5, and Table 5.1 shows the mean and median deviation values of overall selected paths from pre- to post-tape-out.

Table 5.1.: Deviation of the implemented functional path ROs pre- and post-tape-out.

	Cell count $ch_{i1}$	Path length $ch_{i2}$	Driver strength $ch_{i3}$
Mean	2.22 %	2.68 %	6.48 %
Median	0.00 %	0.18 %	4.45 %

Only a minor impact can be seen regarding cell count in Figure 5.5a and path length in Figure 5.5b. Also, the mean and median average values indicate that there are only minor deviations. The slightly higher mean value also indicates some dominant paths (e.g., RO 20 in Figure 5.5a) because the mean value is more outlier sensitive than the median value. Impact due to driver strength (Figure 5.5c) is slightly higher. Increasing driver strength indicates that timing issues have been fixed in the **ECO** runs. Either buffers have been inserted with contributions now to the driver strength of the path, or logic cells have been replaced by cells with a higher strength that can drive higher loads.

However, complete destruction of the paths pre- and post-tape-out cannot be observed - even for the other characteristics. The original path selection is kept after all further **ECOs** are

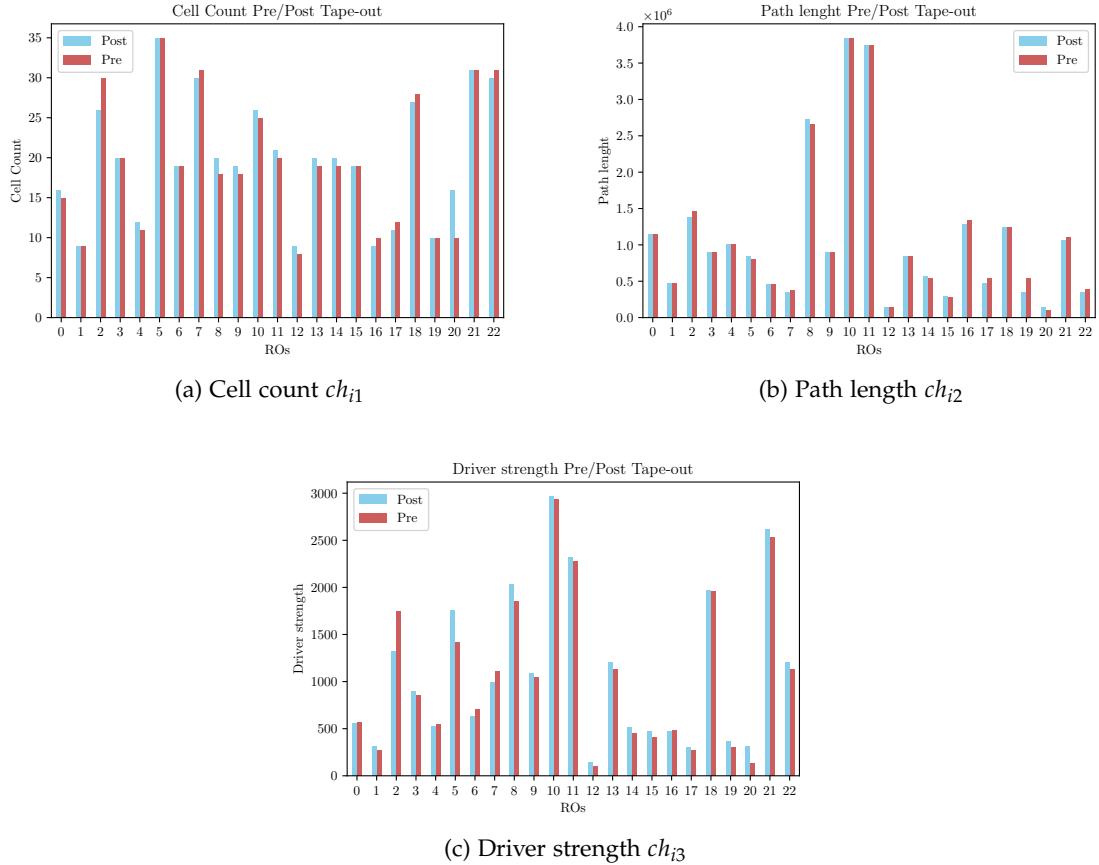


Figure 5.5.: Path characteristic pre- and post-tape-out.

performed. A negligible deviation can also be seen in the individual buckets.

### 5.2.2. Simulation with Functional Patterns

A part of the **MCU** development flow is the IR drop analysis. Static and Dynamic IR drop is investigated to verify the **MCU**'s **power delivery network** (PDN) and ensure uniform distribution of VDD and VSS on the chip. A tool such as Ansys RedHawk-SC [108] is used for such investigations and is equipped with a graphical interface that shows the switching activities and the IR drop on the floor plan. Therefore, it would be of interest to know whether the selected and implemented functional path ROs are located in areas of significant IR drop. For example, Wang et al. [21] had reported an IR drop that affected the functional path RO results of the chip. The authors reported a static IR drop of more than 10 %.

In contrast, the static IR drop of the **MCU** design in this work is minimal due to the excellent connection to the power grid and the high power integrity of the package. Therefore

the simulated static IR drop can be neglected.

However, the dynamic IR drop strongly depends on the use case and is simulated with some use cases. Two use cases and the dynamic IR drop and toggling activity can be seen in Figure 5.6.

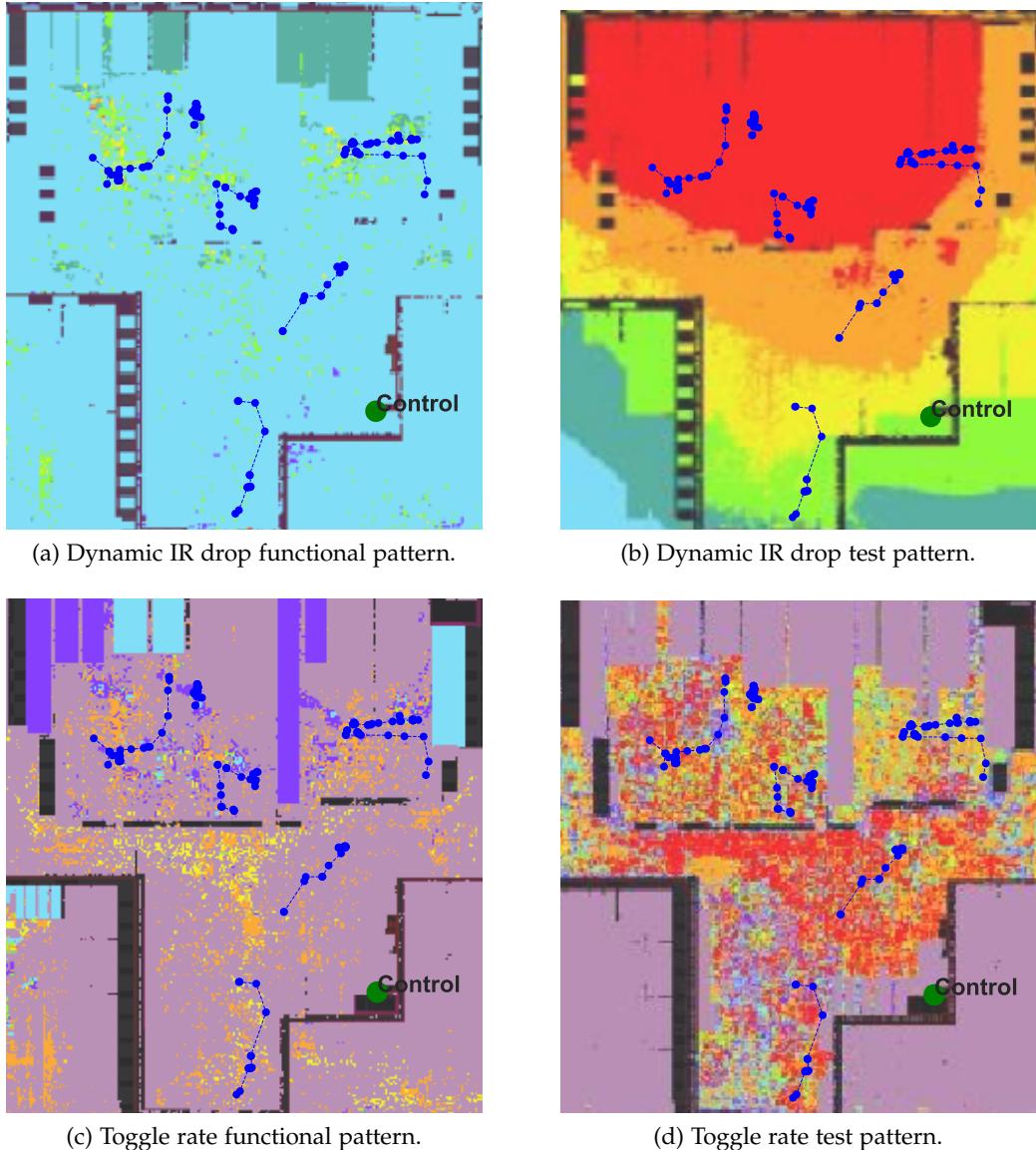


Figure 5.6.: Ansys RedHawk-SC simulation on Module A including the visualization of the implemented functional path ROs.

Figure 5.6a shows the dynamic IR drop on Module A in a highly active time domain of a functional customer-oriented test pattern, and in Figure 5.6c, the corresponding toggle activity during the simulated time domain is shown. A toggling activity can be seen across

the module, whereas the dynamic IR drop occurs only in the module's upper region. At least one of the functional path ROs was implemented in the region with some dynamic IR drop. However, if a different customer use case is simulated, the dynamic IR drop may change and occur at other locations on the chip. Therefore, harvesting all regions in detail with functional path ROs in advance in functional use cases is challenging. However, the IR drop that occurs in the design is in the low single-digit percentage range, which is not worth mentioning for a robust design.

On the other hand, in Figure 5.6b and Figure 5.6d, one of the most aggressive test patterns is simulated. The toggling activity covers a wide area, and a dynamic voltage drop occurs in the module. The functional path ROs are implemented in the areas with higher IR drop and the areas with lower IR drop. Here, too, the IR drop is in the single-digit percentage range.

The simulation showed an IR drop, but it is not noticeable (static) or very small (dynamic). In order to monitor possible weak areas in the design concerning the [PDN](#), it is good to have as many functional path ROs as possible and distributed over a wide area.

### 5.3. PRE- Silicon Validation and Improvement

In this section, the [Pre-Silicon](#) validation is done based on analog simulations, focusing on validating and improving the path selection flow. Section 4.1 describes the path selection flow as a heuristic flow with a low turn-around time that fits perfectly into the industrial design flow. However, path selection must be validated, and a heuristic is an imperfect process that strives to improve. The time between the tape-out and the first silicon usually takes weeks to months. Thus, this [Pre-Si](#) validation flow is proposed to get early feedback from the selection flow. Then, there is no need to wait until the first silicon is available.

The validation of the selection flow can be done either by simulating the paths or with the measurements on silicon. As mentioned, if the measurement approach on silicon is chosen, there is a certain amount of downtime until the sample measurements are available for a statistically relevant basis. In addition, many corner lots and measurement conditions are required to cover the entire [PVT](#) range, which is difficult to reach in the tightly scheduled engineering phase and with limited resources. The standard test program usually covers only some discrete voltage or temperature conditions, for example, and not a continuous sweep of the whole range. Therefore, the simulation-based approach is favorable in that case. Thus, we can validate the physical design-based path selection and improve the selection flow for future derivatives and redesigns in the same technology.

### 5.3.1. Model Generation

After tape-out, the design is fixed and will not be changed. At this point, model generation begins. Model generation is used to create the most accurate circuit representation possible in a simulatable SPICE (Simulation Program with Integrated Circuit Emphasis) [72] model.

The **MCU** design contains various gates and components, all of which have a unique designation. The designation of all components is equal to the designation in the **STA** path report used in Section 3.5 and Section 4.1. The model extraction extracts only the components from the design that are part of the functional path **STA** report. The goal of the extraction is to obtain a simulatable analog SPICE model from the functional path RO, including the parasitic elements that act on the functional path RO in the chip. Three components are necessary to start the model extraction. One is the chip design representation, including the parasitic components, the so-called **standard parasitic exchange format (SPEF)** file; second, the path information file containing all functional path ROs; and third, the standard cell library file containing the description of the gates used in the design. The process of the model extraction can be seen in Figure 5.7.

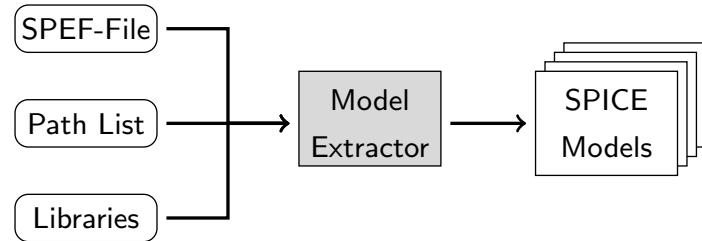


Figure 5.7.: SPICE Model extractor with data in- and output. Adapted from [42] © IEEE 2022.

The **SPEF** file contains the parasitic resistance and capacitance, which must be considered to mimic the exact behavior of the functional path ROs on the chip. All functional paths are listed in the path information file, which are implemented as functional path ROs. The path information file also contains the gates in the feedback loop. In some cases, an inverter is placed to ensure the path's inversion, or some buffers are added during the implementation to provide a reasonable slew rate of the oscillating signal. In addition, the path list can also contain sensitizable functional paths without the **RO** implementation. The library file has comprehensive information of all gates in the design, including the boolean function, characterization data, and the analog transistor model. These three files are passed to the model extractor, and the output are analog SPICE models of the input path lists. Exactly one SPICE model is created per entry in the path list. The side inputs of the path are terminated with the non-controlling value, so the extracted path is sensitized by default. The parasitic elements are also included in the SPICE models.

As mentioned, the path list can contain either functional path ROs or sensitizable functional paths. The model extractor recognizes the difference and decides what kind of SPICE model to be extracted. Figure 5.8 shows a sketch of the SPICE model after extraction from a functional path (5.8a) and from a functional path RO (5.8b).

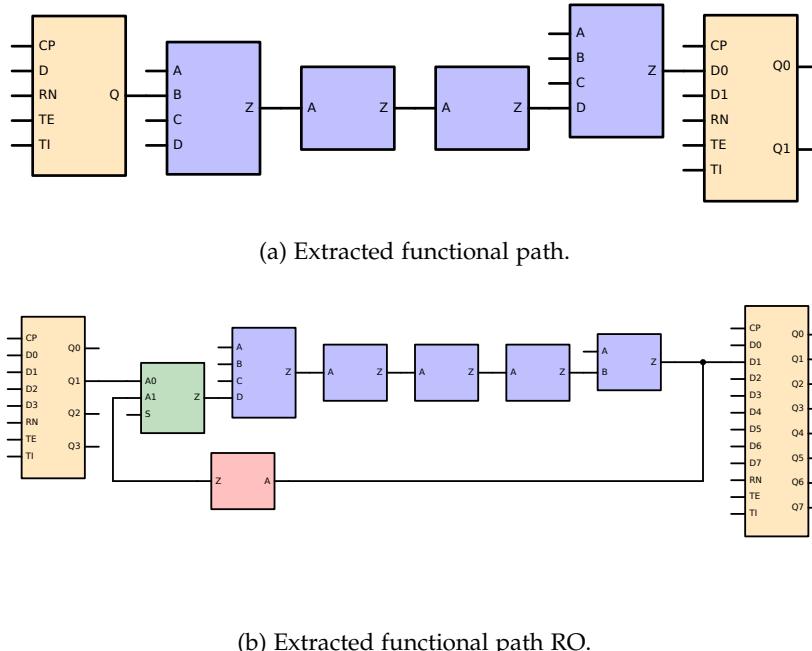


Figure 5.8.: Two alternatives of the extracted analog SPICE models. Adapted from [42] © IEEE 2022.

The generation of the SPICE model from the [SPEF](#) file ensures realistic behavior, since the paths on the chip are exposed to parasitic effects. Therefore, the extracted SPICE models are very close to reality.

The extracted SPICE models are used to investigate the sensitivities of the functional path ROs. Thus, the heuristic path selection process is validated with the sensitivity analysis of the SPICE models. A second use case of the SPICE models is to perform model-hardware correlation once the silicon arrives from manufacturing which is not further investigated.

### 5.3.2. Sensitivity Analysis of the Functional path ROs

The extracted SPICE models are analyzed with respect to their sensitivity under different PVT conditions. The resulting output parameters are the delay time  $d_{p_i}$  of a path or the frequency  $f_{p_i}$  of the functional path RO. The delay time of the sensitizable functional paths (Figure 5.8a) is further distinguished for the falling ( $d_{p_i}^{fall}$ ) and rising ( $d_{p_i}^{rise}$ ) edge. In the case of a functional path RO (Figure 5.8b), the oscillation frequency is additionally determined, which

results from the superposition of the rising and falling case  $f_{p_i} = \frac{1}{d_{p_i}^{fall} + d_{p_i}^{rise}}$ . Sensitivity analysis investigates the sensitivity of such delay/frequency values at different **PVT** conditions. Thus, it can be indicated which paths react particularly sensitively to certain **PVT** parameters.

The investigation of **PVT** conditions is divided into the individual process, voltage, and temperature components and considered separately. Thus, a path's delay/frequency sensitivities are determined by varying one of the three separate components (P – V – T) and keeping everything else the same.

The voltage and temperature sensitivity of the SPICE models are investigated with a transient analysis. Here, the voltage and temperature conditions are varied in discrete steps over the specified operating range provided in the datasheet of the MCU. Thus a sensitivity around the nominal operating point can be calculated in terms of voltage and temperature sensitivities of the functional path ROs. Such analysis does not consider local dynamic voltage droops or temperature effects of gates within the analyzed path. However, these simplifications are sufficient to give a general order of magnitude of the sensitivity of the analyzed paths. The investigation of IR drop was shown in Section 5.2.2.

The voltage and temperature analysis is done with the transient analysis on a SPICE simulator; the process sensitivity is analyzed with the MunEDA WiCkeD™ Tool Suite [109]. The WiCkeD tool performs the sensitivity of the process parameters with the SPICE simulation in the background. The WiCkeD tool investigates all design parameters (geometry of transistors and passive elements) under a given statistical process variation and mismatch based on the provided voltage and temperature condition [110]. The sensitivity of the output parameters (delay/frequency) is analyzed, given the global process variation of the design parameters. The sensitivities of the output parameters are given for each simulated functional path RO. A sensitivity analysis with respect to process parameters using the WiCkeD tool consumes less set-up effort and simulation time than an extensive Monte-Carlo simulation.

The sensitivity analysis described above with respect to the **PVT** conditions provides an initial indication of how good the path selection flow is. In addition, the sensitivity analysis of the functional paths can be used to improve the selection flow further.

### 5.3.3. Improvement of the Heuristic Selection

The heuristic process of selecting paths based on physical design data, explained in Section 4.1, may result in some functional paths that are sensitive to specific parameters not being selected as functional path ROs. This deficiency in path selection is that the defined characteristics do not adequately cover specific parameters that affect performance.

The analysis focuses on the process parameters in functional paths (see Figure 5.8a). The aim is to find lacking process parameter sensitivities and try to define an additional or

improved selection characteristic. However, the process parameters consist of a large number of individual parameters. Some individual process parameters dominate the sensitivity analysis of a functional path. Different individual process parameters may be dominant if other functional path ROs are considered.

Thus a bunch of randomly selected sensitizable functional paths is fed into the model extractor, and SPICE models are generated, as in Figure 5.8a. The resulting SPICE models are analyzed for their delay sensitivity concerning the process parameters. Assuming that this sensitivity analysis results in certain path types with a very high sensitivity to precisely one or more process parameters, they can be analyzed with respect to the physical design characteristics and new selection features can be defined. In this way, new features can be added to the path selection process, and the heuristic selection process can be continuously improved.

#### 5.3.4. PRE- Silicon Validation Results

In order to show the quality of the path selection process for the 22 implemented functional path ROs, the SPICE models are extracted using the methodology proposed in Section 5.3.1. Thus, 22 SPICE models of the functional path ROs (see Figure 5.8) are generated, and the PVT sensitivities are investigated to validate the selection process based on physical design data.

The voltage and temperature sensitivities of the selected paths are calculated. The normalized sensitivity of voltage and temperature with respect to the RO frequency  $f_{pi}$  is shown in Figure 5.9. The normalization defines the lowest occurring sensitivity of a path as 0 % and the highest sensitivity as 100 %.

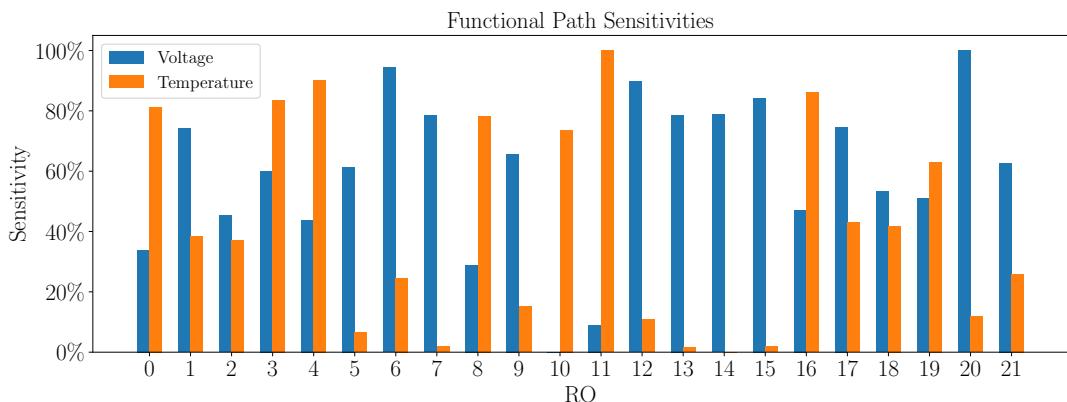


Figure 5.9.: Voltage and temperature sensitivity of the functional path ROs. Adapted from [42] © IEEE 2022.

Some paths are more sensitive to voltage, others to temperature. It is noticeable that ROs

with a long functional path length are more susceptible to temperature (see RO 10 & 11). Thus, the path selection process finds a very heterogeneous set of ROs that differ in their sensitivity to the voltage and temperature parameters, which is the goal.

The selected functional path ROs are also investigated regarding their sensitivities in terms of process parameters. The technology used has almost 100 individual process parameters; their change will be analyzed for frequency sensitivity. Figure 5.10 shows that all process parameters are covered with the selected 22 functional path ROs.

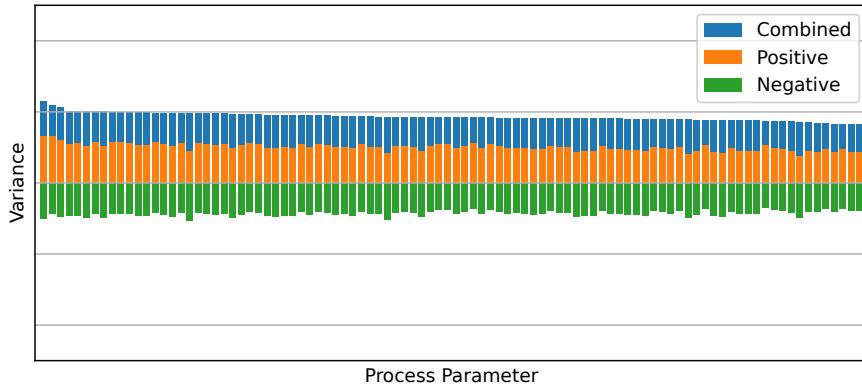


Figure 5.10.: Variance of the frequency sensitivity with respect to the individual process parameters of all selected functional paths ROs. Adapted from [42] © IEEE 2022.

All individual process parameters are listed on the X-axis, and the variance of the sensitivity to frequency is shown on the Y-axis. A distinction can be made between positive frequency changes and negative frequency changes. Some paths react with a positive frequency change to certain parameters, others with a negative one. The absolute value is shown in blue and results from the sum of the positive and negative frequency change.

By looking at each functional path RO individually, some paths are sensitive to the process parameter as a whole, and others are not. The variance of the frequency sensitivity of the selected functional paths ROs with respect to all process parameters is shown in Figure 5.11. The portions of the individual process parameters have been cumulated.

Thus, some paths are more sensitive to the process than others. By combining the results, the heuristic path selection process finds paths with different frequency sensitivities with respect to the whole PVT conditions, which was also the intention of the methodology. Note that the absolute frequency change varies significantly within PVT conditions. Much of the frequency change occurs due to the variation of voltage and temperature - voltage, in particular, has an enormous impact. The effects of process parameters on frequency sensitivity are the smallest over the entire PVT range. Therefore, the oscillation frequency must be measured with high precision to resolve the sensitivity of all PVT parameters.

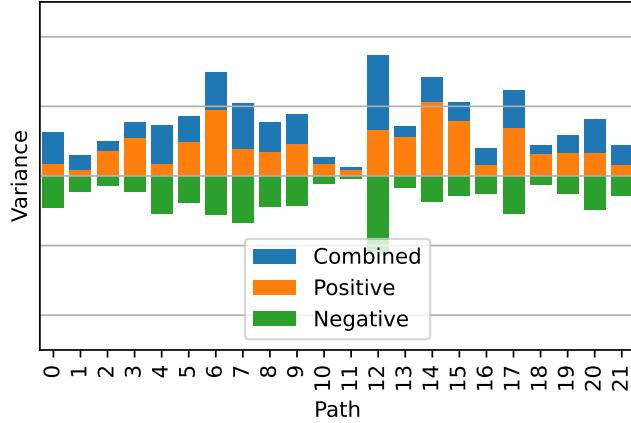


Figure 5.11.: Variance of the frequency sensitivity of the selected functionalpaths ROs with respect to all process parameters. Adapted from [42] © IEEE 2022.

#### 5.3.4.1. Results on Improving the Heuristic Selection

In order to verify the assumption made in Section 5.3.3, thirty randomly selected functional paths of the design are also selected, and SPICE models are generated using the model extractor. The SPICE models are then analyzed regarding their sensitivity to the process parameters to find paths that are even more sensitive to specific parameters and thus improve the heuristic approach.

The number of paths, in this case, is limited to thirty to illustrate the methodology; in reality, many more paths are examined in this way. Figure 5.12 shows that some paths have a significant variance and, thus, a considerable sensitivity in process parameter change.

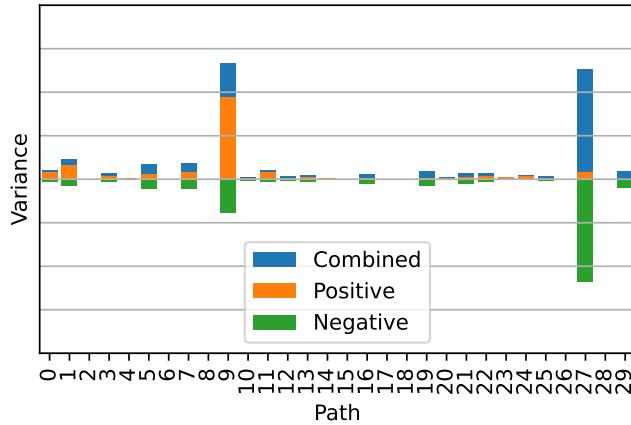


Figure 5.12.: Variance of the delay sensitivity of the randomly selected sensitizable functional paths with respect to all process parameters. Adapted from [42] © IEEE 2022.

The majority of the paths have low sensitivity because they were randomly selected, but a few paths are very sensitive (Path No. 9 & No. 27). Such sensitive functional paths will be further investigated to determine if additional features for the physical design data-based path selection can be derived. For example, if a particular cell topology is very sensitive, the physical design data-based path selection can be optimized to consider these cell topologies in the selection as well. In this way, the heuristic path selection is improved with this feedback and is ready for future derivations and designs.

## 6. PRE-SI Summary

This part explains and illustrates the [Pre-Silicon](#) activities to implement, select, and validate functional path ROs. For this purpose, the most essential segments are divided into three chapters.

Chapter 3 presents the basic concept of the functional path RO, including the implementation methods and related infrastructure. The basic implementation of functional path ROs is presented and illustrated by two functional paths that are implemented as [ROs](#). It is shown that functional path ROs have an advantage in terms of area and leakage current of over 96 % compared with traditional [RO](#) structures for 128 implemented [ROs](#). A proof-of-concept is conducted concerning the advanced implementation concepts. In particular, two concepts are proposed on how to select functional paths to ensure efficient implementation. It is examined whether sufficient paths can be sensitized with a single pattern for the natural loops approach - which is the case. The novel concept of self-enabling functional path ROs is presented, and a feasibility study is conducted. The use of the self-enabling approach can save more than 70 % of routing resources and, in some cases, even up to 80 % for the investigated [MCU](#). Furthermore, two options are proposed for self-enabling: Option 1 and Option 2. Option 1, also labeled direct self-enabling, uses [ATPG](#) constraints of the launch [FF](#) of the path itself. Option 2 (indirect self-enabling) utilizes nearby launch [FF](#) and an unlock gate for enabling. A scan-controlled control infrastructure is introduced. Due to its hybrid properties, the control infrastructure can utilize the basic concept of functional path ROs and the self-enabling functional path ROs. The main advantage is that the sensitization of the functional path and the settings of the control infrastructure are maintained with a single scan pattern. The entire implementation flow of the functional path RO is compatible with standard implementation flows used in the industry.

The path selection methodology is presented in Chapter 4. The chapter presents a clustering-based selection flow to select a representative subset of functional paths on the design for an [RO](#) implementation. 22 functional paths are selected in the [MCU](#) under investigation and implemented as [ROs](#). The path selection flow provides a very low turnaround time and sorts the paths into buckets. The paths in the buckets have similar properties concerning the defined path characteristics.

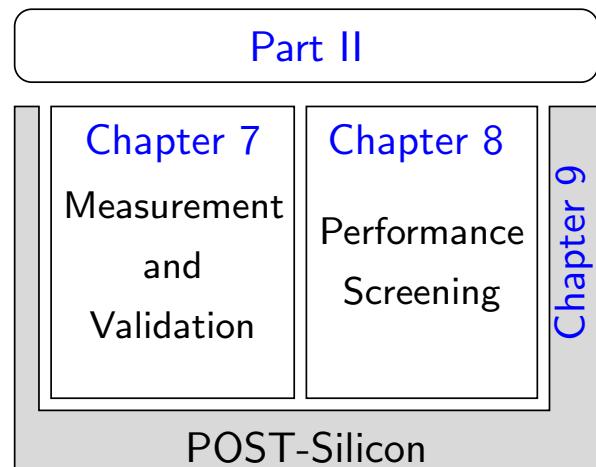
The last chapter in the [Pre-Silicon](#) part describes the [Pre-Si](#) verification and validation of

---

## 6. PRE-SI Summary

---

functional path ROs. The implemented functional path ROs are verified for their functionality via simulation. A side benefit of timing-aware simulation-based verification is an early estimation of the oscillation frequency of the implemented functional path. This is used for a first test-limit setting. Also, the circumstance is analyzed in that the [ECO](#) implementation of the functional path ROs is combined with additional last-minute chances before the tape-out. The results show no significant changes in the functional path selection. Also, a first analysis of the [PDN](#) stability of the regions with functional path ROs is conducted. The path selection flow is validated via sensitivity analysis of the functional path ROs. A methodology is presented to extract analog SPICE models from the design and analyze them regarding [PVT](#) sensitivity. This also allows a continuous improvement of the path selection process. The sensitivity analysis shows that the implemented 22 functional path ROs are more diverse than those not using the path selection process.



Outline of Part II.

## **Part II.**

# **POST-Silicon**

---

Part II of the thesis is dedicated to the Post-Silicon phase. Once the first silicon arrives from the manufacturing and assembly, the MCU is tested, validated, and characterized in depth. The number of produced devices in the early engineering phase is small compared with later mass production. Before ramping to volume production, the devices are extensively tested and analyzed to find design weaknesses and test the specified functionality. The devices in the engineering phase are also manufactured as corner lots to have a considerable spread in the process parameters. An essential part of this phase is the performance characterization and development of an efficient performance screening strategy for the high-volume phase.

What also changes is the abstraction level. The MCU at the top-level is now considered instead of any chip partitions such as Module A, B, or C. In addition to the three modules considered, the MCU consists of further modules that are all merged to top-level design and then manufactured into a uniform MCU. The functional path ROs do not aim to detect defects in the devices for which conventional structural test methods are responsible. Instead, the goal is to determine performance by considering only the frequencies of the functional path ROs.

In this thesis, 22 functional path ROs are implemented on each of the MCUs. The functional path ROs must be measured and validated to determine if they are working as expected, which will be explained in Section 7. The section is basically looking for the RO frequencies themselves.

The Section 8 will describe the methodology of performance characterization and the development of a machine learning (ML) based performance screening. The main research question in this section is how the functional path ROs perform on the performance screening.

## 7. Measurement and Validation of the Functional Path RO

### 7.1. Functional Path RO Measurement Results

The functional path ROs are measured on [automatic test equipment \(ATE\)](#) load-board using a digital probe card extension that taps the frequency of the functional path ROs on the [GPIOs](#) of the packaged device. The measurement sequence is explained in Section 3.1. During the shift-in of the scan pattern, the functional path ROs are sensitized. When the functional path RO is enabled - enable signal is high - the [ATE](#) receives a handshake signal, and the digital probe card is activated. As soon as the frequency measurement is completed and the measured frequency is within the expected range, the procedure is continued with the subsequent functional path RO measurement. If a frequency is outside the expected range, the root cause of the discrepancy must be identified, which can be very broad.

The measurement of the functional path ROs should be at a stable voltage and temperature condition to have a high reproducibility of the measurements. In order to check the reproducibility, the 22 functional path ROs are repetitively measured on the same device. A repetition of 100 times the measurement procedure is shown in Figure 7.1. The measured frequencies of each RO are normalized since the frequencies of the ROs are in different ranges.

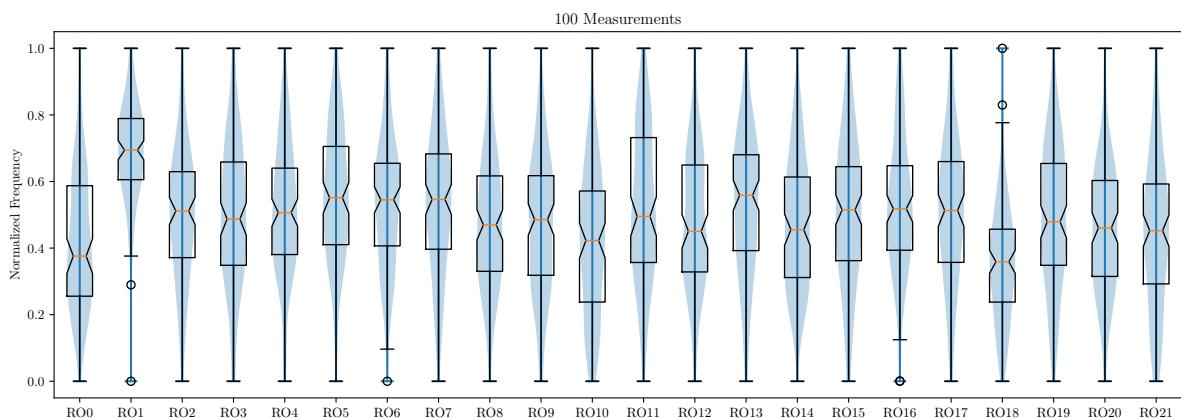


Figure 7.1.: Repetitive measurement of the functional path ROs on a normalized scale.

The repetitive measurements show only a few outliers in some of the ROs. Except for these rare occurring outliers, the measurements show a stable distribution around the mean.

In order to quantify the reproducibility, the **coefficient of variation (CV)** is calculated. The **CV** is a dimensionless number that reflects the relative accuracy of a measurement [111]. The **CV** is calculated with the standard deviation ratio to the measurements' mean and is expressed as a percentage value. The target is to have a **CV** close to 0 % that indicates perfect reproducibility of the measurement. The calculated **CV** is shown in Figure 7.2.

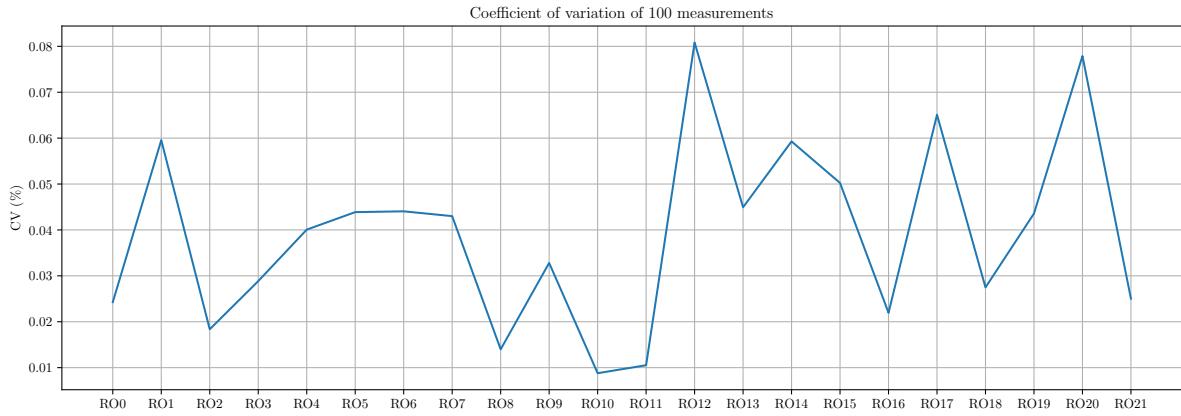


Figure 7.2.: Coefficient of variation of 100 repetitive measurements.

The overall **CV** is below 0.08 %, which indicates the high repeatability of the functional path RO measurement. The experiments are executed on two packaged devices, and the **CV** is in the same range in both measured devices. Thus the functional path ROs can be measured with high reproducibility using the **GPIOs** and the **ATE**.

In order to compare the measured frequency range with the expected frequency range, a statistically relevant amount of devices is measured. The measured **RO** frequencies are then compared with the test limits determined in the **Pre-Si** phase. Section 5.1.2 determined an upper  $f_{p_i}^{upper}$  and lower  $f_{p_i}^{lower}$  test limit for each **RO**. Ideally, all measured **RO** frequencies are within the determined test limits.

The test limits of the functional path ROs are compared among 3858 devices from 25 wafers to see if the measured **RO** frequencies are in the expected range. The results are shown in Figure 7.3.

All measured functional path ROs are within the pre-determined test limits. Therefore, the functional path RO frequencies on silicon correspond to the values specified in Section 5.1.2. Thus, the functional path ROs all work within the verified frequencies.

The functional path ROs deliver promising reproducibility and expected frequency range results. However, checking the voltage at the chip is also essential since the functional path RO measurement requires stable voltage conditions. At-speed test patterns, such as the path

## 7. Measurement and Validation of the Functional Path RO

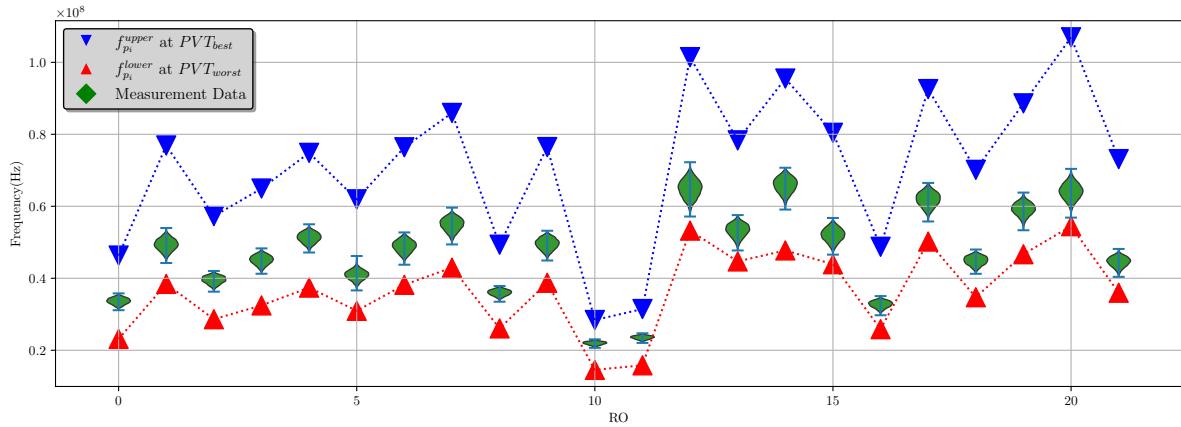


Figure 7.3.: Measurement data from 3858 devices of 25 wafers and the simulated test limits.

delay pattern, can significantly impact the overall power supply stability and cause voltage droops [17]. This is especially true during changes in the circuitry mode, e.g., from the shift-in phase (long clock cycles) to the capture phase (short at-speed clock pulse) [112]. Those abrupt transient events can cause voltage droop, which is hazardous to measurement where a stable voltage condition is needed. In order to investigate the voltage stability in the functional path RO measurement, the voltage of the **MCU** is sensed and observed with a scope.

The voltage is observed at the load board of the test head, where the **DUT** is mounted on the test socket. The transition of the shift-in phase to the measurement phase is observed where the **MUX** of the respective functional path RO is enabled. The snapshot of the scope is shown in Figure 7.4.

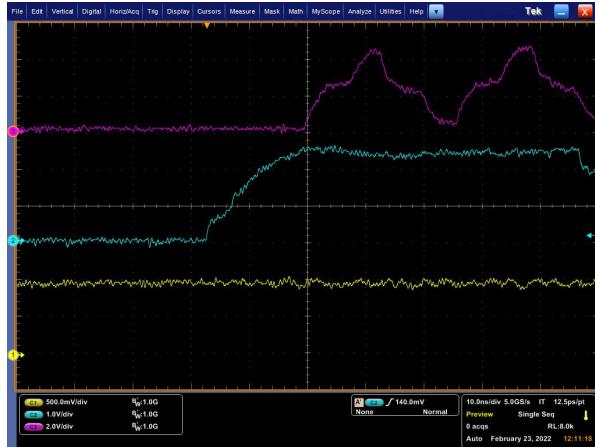


Figure 7.4.: Voltage drop measurements.

The three traces in the snapshot are CH1 (yellow), the supply voltage measured in the

## 7. Measurement and Validation of the Functional Path RO

---

sense pins of the **MCU**; CH2 (turquoise), the trigger signal, which is the enable signal of the **MUX**; and CH3 (magenta), the **GPIO** pin oscillation signal of the **RO**. The critical trace is the yellow signal which does not reveal any droop in the presented transition phase. That confirms the **Pre-Si** dynamic voltage droop simulation in Section 5.2.2, where only a little voltage droop is expected at the **MCU** level during transient events. Thus it can be eliminated that any significant transient effects in the supply voltage are happening in the transitional period of the phases in the functional path RO measurement.

The functional path ROs' frequencies can also visualize the **die-to-die** gradient over the wafers. One functional path RO is selected, and the frequency is observed for all dies on a wafer. Such an approach makes the **D2D** process variation visible and can be used to track process stability or other manufacturing issues. A visualization of the **D2D** process variation based on a functional path RO frequency is shown in Figure 7.5.

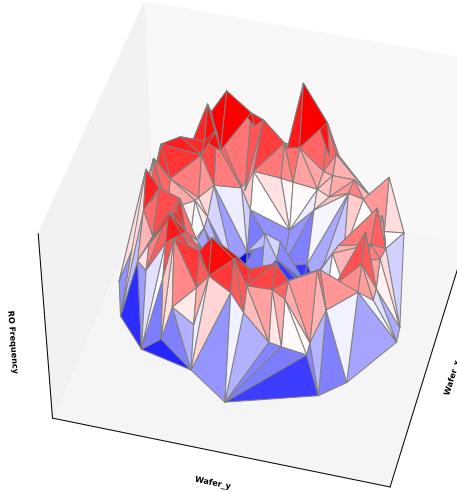


Figure 7.5.: Wafermap of the frequency distribution of an functional path RO.

The observed **RO** frequency can be abstracted as a *donut shape*, which is one of the possible shapes that can be observed in wafer maps [63]. This is only one example of a dedicated use case in which the functional path RO frequencies can help to improve and monitor the manufacturing process.

# 8. Performance Screening Using Functional Path RO

## 8.1. Method and Data Set for the Performance Screening

The main goal of the functional path ROs is to use such structures as performance monitors for performance screening. The performance screening is part of the **MCU** test process, which is shown in Figure 8.1. Each device has to pass the performance screening for a successful final **back-end (BE)** test. The functional path **RO** measurement is part of the **front-end (FE)** and **BE** tests. Therefore for each manufactured device, the functional path **RO** data is obtained. Also, the **SMON** module is measured. Based on the obtained **RO** data, the performance screening is done as part of the final **BE** test using the **RO** data as monitors. The device can be shipped if the performance screening and the final test are passed.

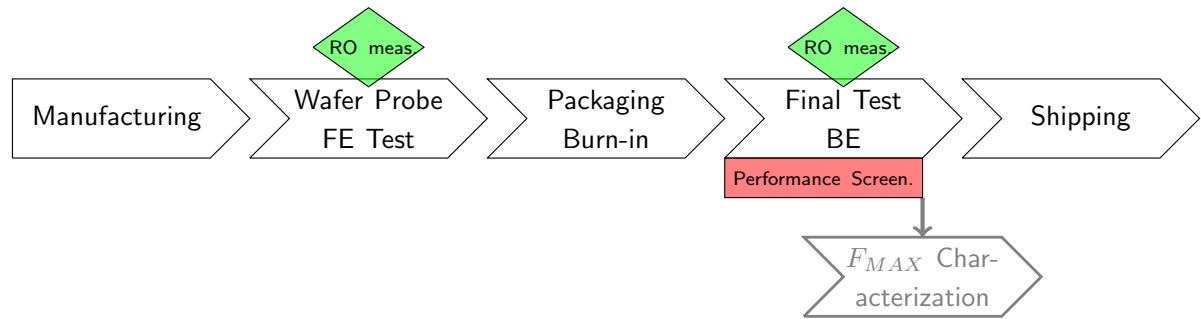


Figure 8.1.: Test flow of an **MCU** including the performance screening.

However, the performance screening is a challenging process. The performance screening shall be done by only considering the **RO** measurement data. Therefore, the **RO** measurement data is used as input for the performance screening, and the output is the pass / fail decision. In order to initialize the performance screening, a set of golden devices is obtained for an elaborate  $F_{MAX}$  characterization used as the *ground truth*. Such characterization is a high-effort and time-consuming process executed on these golden devices. In the first part of this section, the  $F_{MAX}$  characterization is explained; afterward, the data set and principles for the performance screening are described.

### 8.1.1. $F_{MAX}$ Characterization

As stated previously, the performance determination of an **MCU** is challenging. No measurement gear can measure the precise performance of a particular device in a one-shot measurement, as in voltage or current measurement. This is because the performance of the **MCU** depends on many conditions. e.g., use-case, **PVT** conditions, and many more. However, to have an initial reference set to correlate the **RO** data with the device performance, a set of golden devices is deeply analyzed with a  $F_{MAX}$  characterization. The  $F_{MAX}$  characterization determines the performance of a device with an elaborate process that is similar to a **system-level test (SLT)**. The **SLT** initially tests the devices to find defects. In the  $F_{MAX}$  characterization, the **SLT** setup is used to determine the performance of a device in a particular condition for a particular use case.

The outcome of the  $F_{MAX}$  characterization is the maximum achievable clock frequency  $F_{\max}^{Ti}$  for a given functional test pattern  $Ti$  ( $i \in \mathbb{N}$ ). The  $F_{\max}^{Ti}$  is the highest frequency at which the functional test pattern is executed under a predefined condition (voltage, temperature) without errors.

Each golden device is packaged and mounted on an **SLT** board for  $F_{MAX}$  characterization. The **SLT** board has a high-precision **radio frequency (RF)** socket with low resistance and the lowest possible parasitics. In addition, the **SLT** board has high power integrity and built-in energy storage capacity to prevent voltage drop during testing. During characterization, the temperature and voltage of the **MCU** can be precisely controlled and are continuously sampled by on-chip voltage and temperature sensors.

The **DUT** is placed on the **SLT** board in the **RF** socket with a semiconductor test handling system to improve the repeatability of the measurement process. The voltage and temperature are set to defined voltage  $V_{\text{crit}}$  and temperature  $T_{\text{crit}}$  conditions. The **MCU** is programmed with the firmware when the voltage and temperature conditions reach the defined steady-state values. The **MCU**'s clock frequency can also be controlled with the **SLT** setup. Then the functional pattern  $Ti$  is uploaded and launched in an infinite loop on the **MCU**. The **MCU** starts executing the functional pattern at a low frequency. The frequency is slowly increased in each loop until the functional pattern fails. The process is repeated several times to ensure that the failure frequency of the **MCU** remains at the same value. The last working frequency prior to the **MCU** failure frequency is considered and stored as  $F_{\max}^{Ti}$  of the functional pattern  $Ti$ . The measurement of  $F_{\max}$  is performed with several functional patterns [24].

The applied functional patterns in the  $F_{MAX}$  characterization show a wide variety in stressing different design units of the **MCU**. The functional patterns are designed with the intention that no customer application fails at a lower frequency than a functional pattern. Therefore, some patterns are designed to exhaust the maximum power of the **MCU** as

specified in the data sheet. These patterns result in high-load jumps and create massive cross-talk between functional blocks. Other functional patterns stress specific design parts, such as dedicated data bus lines or repetitively executing a specific instruction. Thus, all functional patterns attempt to force the **MCU** to its power limit. However, some  $F_{\max}$  uncertainty remains in the characterization, caused by (i) the diversity of possible customer applications, (ii) measurement uncertainty and (iii) defective devices.

- i): An automotive **MCU** is intended for a large number of applications. Thus there are thousands of use cases, and each application puts more strain on some parts of the **MCU** than on others. Therefore, the applied functional pattern stresses the **MCU** over the specified operating range and ensures a robust **MCU** in each customer application. However, in order to avoid omitting customer use cases and to avoid the applied functional patterns stressing some areas, a guardband is needed in performance screening that covers these contingencies.
- ii): Some measurement uncertainties also affect the  $F_{\max}$  characterization process. The characterization is performed under worst-case operating conditions in terms of voltage ( $V_{\text{crit}}$ ) and temperature ( $T_{\text{crit}}$ ) at the limits of the specified operating window. Even minor deviations of the voltage and temperature conditions affect the measurement result. In addition, small mechanical vibrations cause a change in the base resistance, or statistical noise changes the result. Another consequence is that error messages of the  $F_{\max}$  measurement software are due to contact problems during the measurement. Some **GPIO** pads are also activated in the functional pattern; errors can occur in the presence of a contact problem. Since the measurement process is automated, the setup has appropriate error handling so that errors that occur do not interrupt the characterization. If an error occurs in an  $F_{\max}$  test pattern, the characterization of the device continues, and the specific error-prone test is marked with a not-a-number value. These errors are rare but are worth mentioning. Therefore, many hard-to-predict factors influence the  $F_{\max}$  characterization during acquisition, so careful attention must be paid to any measurement uncertainties and errors when processing the measurement data.
- iii): Another cause of discrepancies in the  $F_{\max}$  characterization are physical defects in the golden devices (e.g., shorts, opens). Those defective devices are usually detected within the productive test flow with stuck-at, transition, or path-delay tests. However, at the point in time when the golden devices are obtained from the production test flow, the production test is not technically mature because the entire product is still in the engineering phase. Consequently, a small fraction of golden devices exhibits a defect. These devices must be removed since the performance screening does not target defective devices. Therefore, it is essential to

have defect-free golden devices for  $F_{MAX}$  characterization since the characterization is the benchmark for the whole performance screening process, and defective devices significantly disturb the screening process.

Outlier detection is used to address the three effects mentioned in the  $F_{MAX}$  characterization. Outlier detection aims to filter out devices that deviate from the wafer median. It is assumed that the effects mentioned above cause such deviating components. The  $F_{MAX}$  characterization for each test is considered on its own, and devices deviating more than  $N$  times the standard deviation from the wafer median are discarded. An example of outlier detection is shown in Figure 8.2.

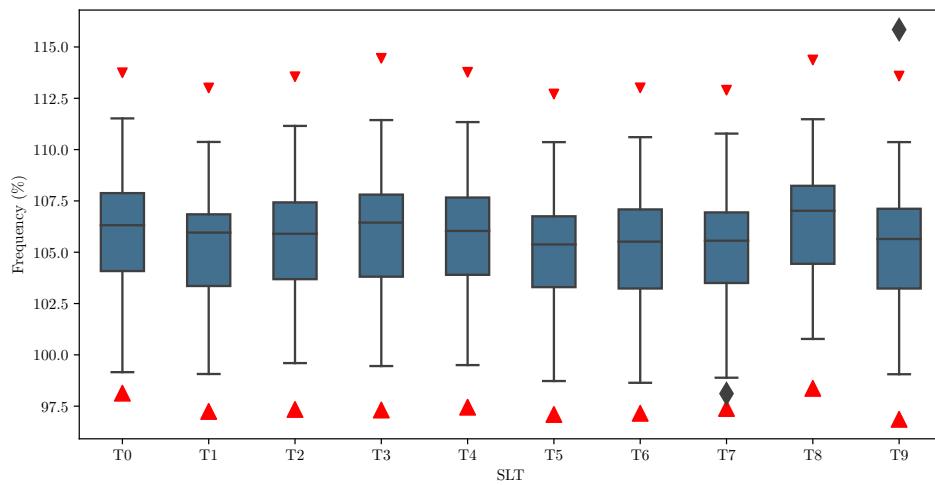


Figure 8.2.:  $F_{MAX}$  characterization results from a wafer after the [SLT](#).

The boxplot shows the distribution of a wafer with 109  $F_{MAX}$ -characterized devices for ten functional test cases. The  $\pm 3$  sigma border is shown as a red triangle for each distribution. In T9 a device deviates from the wafer median by more than three sigma. Accordingly, such devices are discarded in the overall  $F_{MAX}$  characterization even if they are inconspicuous in the other test cases. However, one device is slightly below the whiskers in T7 but within the defined sigma border of the test case. This device is not discarded according to the defined rule.

### 8.1.2. The Data Set for Performance Screening

The data set used for the performance screening can be distinguished into two groups: the *unlabeled data set* and the *labeled data set*. In order to understand the nomenclature, the terms feature and label (explained in Section 2.6) are assigned to the context of performance

screening. The *features* - that are the input data in [machine learning \(ML\)](#) problems - are the [RO](#) frequency data. So each measured [RO](#) acts as one feature. The *labels* - that are the output data in [ML](#) problems - are the  $F_{MAX}$  of a particular device. Each device that undergoes the  $F_{MAX}$  characterization has multiple labels  $F_{max}^{Ti}$  for the different test cases  $Ti$ .

Thus, a manufactured device that has passed the production test flow from Figure 8.1 is, by default, an unlabeled device. The set of golden devices which are  $F_{MAX}$  characterized are denoted as labeled devices.

The schematic structure of the two data sets is shown in Figure 8.3. The unlabeled data set contains the chip ID, which is the unique identification number for each manufactured chip, and the [RO](#) data from all measured ROs during production. The labeled data set also contains the  $F_{MAX}$  values from the  $F_{MAX}$  characterization. The labeled data set is the ground truth to set up and model the performance process.

In the later production, there are only unlabeled devices available. In order to define the performance screening process, which is done in the next section, the labeled data set is used.

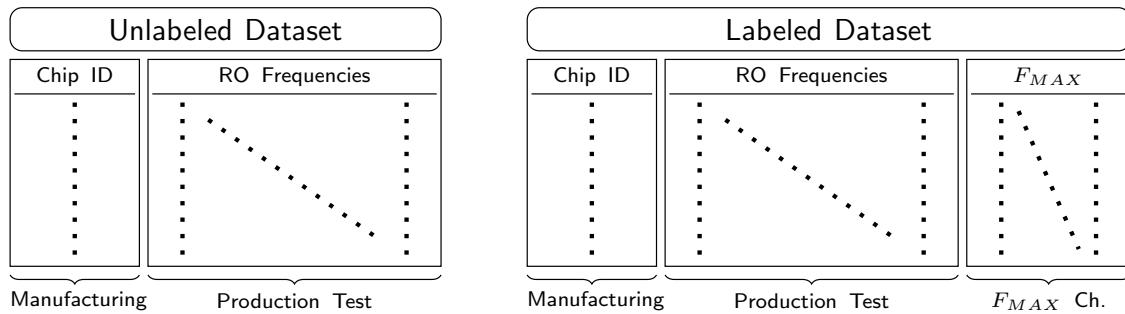


Figure 8.3.: Data set structure from unlabeled and labeled devices.

### 8.1.3. The Principle Concept for Performance Screening

The main objective of the performance screening is to assess if a device meets the specified performance requirements. The frequency of the [RO](#) structures as indirect performance monitors provide the input for this judgment: Functional path ROs and [SMONs](#). Thus the quality of the performance screening depends strongly on the [RO](#) structures. The process is visualized in Figure 8.4.

The [RO](#) frequencies are passed into the performance screening process, and the process decides whether the device passes or fails. The performance screening process, at first glance, can be approximated as a classification problem - numerical input data ([RO](#) frequencies) is used to classify the device as pass or fail. The classification problem also has a significant disadvantage: having a fixed pass/fail border. The entire performance screening model has a fixed border on which all devices are classified. If the border changes slightly, e.g. due to

## 8. Performance Screening Using Functional Path RO



Figure 8.4.: Performance screening process basics.

a change in data sheet performance requirements, the entire model is invalid and must be revised if the requirements change. Therefore, the model can be formulated as a regression problem. The numerical RO frequencies are fed into the performance screening regression model, and a numerical  $F_{MAX}$  is predicted for the device. That approach has a significant advantage: a model valid over the whole  $F_{MAX}$  range.

The classification and regression are classical supervised ML problems. Therefore ML is used to build and train a model for the performance screening. For this training, the labeled data set is used. Once the model is trained, it can be deployed for performance screening based only on the RO frequencies on unlabeled data. The process for training and evaluating the ML model with the labeled and deploying the ML model on unlabeled data is shown in Figure 8.5.

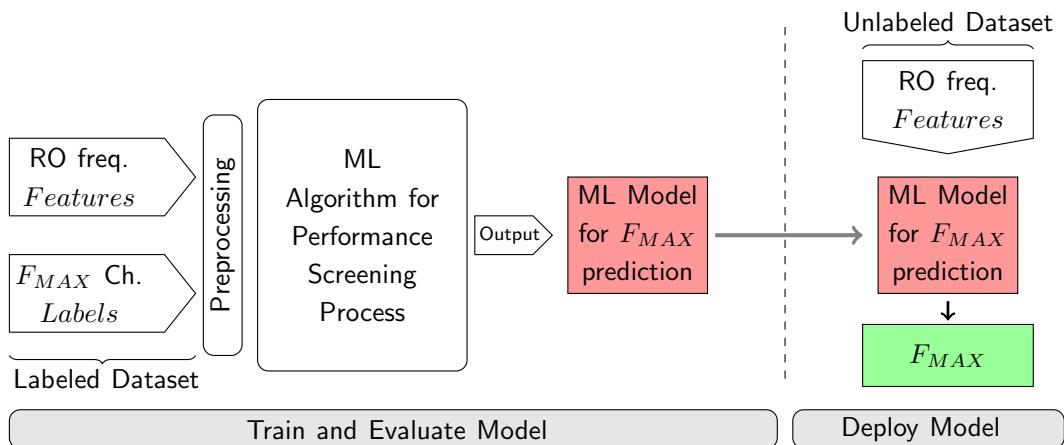


Figure 8.5.: Performance screening process learning and deploying.

**Train and Evaluate the Model:** The initial data set for the training is the labeled set of golden devices with their features (RO frequencies) and the labels from the  $F_{MAX}$  characterization. The labeled data set is preprocessed before being fed into the ML algorithm of the performance screening. The preprocessing steps and obtained analysis will be further explained in Section 8.2. After this, the performance screening flow is defined, and the algorithm is developed, trained, and validated; Section 8.3 describes this process.

**Deploy the Model:** Once the model is trained and the performance verification process is defined, the model can be applied to unlabeled data, in other words, in the production test. This makes the resource-intensive  $F_{MAX}$  characterization obsolete by using only the features for the performance screening. The more accurate the performance screening is, the better it is for automotive quality and yield gain. The utilization of the model and the quality and yield impact will be described in Section 8.4.

## 8.2. Preprocessing of the Data Set

The input data for preprocessing must be labeled devices (see Figure 8.3). The data set is checked for consistency and plausibility, for example, for missing entries or negative numbers. The  $F_{MAX}$  characterization (see Section 8.1.1) includes outlier filtering, so there is a very low probability that such outliers are still present in the labeled data set. The features (RO frequencies) must be within the predefined upper and lower frequency limits of the production test flow (see Section 5.1.2).

Once the data set has passed these checks, the dimensionality reduction starts as the second part of the data preprocessing. This process aims to reduce the complexity of the feature set from a sample feature space, as explained in Section 2.6. The feature selection is performed in this work with the Correlation coefficient to get an indication of the behavior of the features among themselves as well as to the labels. The data set with the functional path ROs as features competes with the SMONs as features. The aim is to identify which input features perform better and which ROs have a high correlation with the labels.

In contrast, the feature reduction transforms the input feature set into a new feature space using, for example, [principal component analysis \(PCA\)](#). PCA decomposes the input feature into its principal components [96]. Thus, it analyzes how many different pieces of information are contained in a data set and how these can be mapped into their principal components without information loss, if possible. The more diverse the data set is, the more principal components are needed to represent the data set. This method is used to investigate which data set contains more information density.

The dimensionality reduction approaches used in this paper aim to compare the two feature sets, the functional path ROs versus the SMONs, to clarify how good the functional path ROs are as performance monitors.

## 8.3. Performance Screening Flow

The high-quality preprocessed data set is applied as input data for the performance screening process. The data set contains either the functional path ROs as features or the SMONs as

features. Two fundamentally different classes of **ML** are used for performance screening: classification and regression. Within the two classes, different algorithms are utilized. However, since there are many algorithms with as many different complexities, this work will focus on a few commonly used algorithms. The main focus is on which **ROs** are better suited for performance screening - with a particular emphasis on the functional path ROs.

In order to get a first overview of the two approaches, namely the classification and the regression approach, Table 8.1 compares the two approaches.

Table 8.1.: Differences and similarities between the classification approach and the regression approach.

Metric	Classification	Regression
Input Data	Labeled data set	Labeled data set
Output Model	Pass/Fail Classifier	Numerical Regression
Encoding	Binary	NO
Applying $F_{TH}$	Beginning	End
Train/Test Split	YES	YES
Cross Validation	YES	YES
Helper Function	Feature Selection	NO
Average Complexity	Low	Medium
Ease of use	High	Medium
Screening Quality	Medium	High
Automotive Quality	Difficult	Guardbanding

The main difference is that the classification approach generates a model for a pass/fail decision given a dedicated threshold frequency  $F_{TH}$  at which the performance pass/fail border is set. In contrast, the regression approach provides a predicted numerical device performance, and the dedicated performance threshold value is applied afterward for a pass/fail decision. The regression model can reach automotive quality requirements using a guard banding technique. The two approaches are explained in the following two subsections. The approaches are independent of which kind of **ROs** are used.

**Notation:** In order to simplify the domain-specific problem into standard **ML** notation, the following notation is used:

- Set of labeled devices  $\rightarrow \mathbb{X}$
- Set of unlabeled devices  $\rightarrow \mathbb{U}$
- Features (**RO** frequencies) of a device  $j \rightarrow x_j$
- Measured Label ( $F_{MAX}$  characterization) of a device  $j \rightarrow y_j$

- Predicted Label (from **ML** Model) of a device  $j \rightarrow \hat{y}_j$

**Specification:** The following performance screening aims to use the input vector  $x_i$  (**RO** frequencies) to predict *one* label  $y_i$  in contrast to the work in [51], where a multilabel performance screening is developed. Thus, *one* model is generated for *one* functional test pattern. That way, the relation between the input features and the functional test cases from the  $F_{MAX}$  characterization is more visible.

### 8.3.1. Performance Screening using Classification

The classification-based performance screening trains a function  $f$  that classifies the devices in pass (1) or fail (0), given a dedicated target frequency  $F_{TH}$ . The labeled data set  $\mathbb{X}$  is binary coded in the first step to transform the numeric value into a binary value depending on  $F_{TH}$ . For each device  $j$  in  $\mathbb{X}$ , an encoded label  $y_j^*$  is generated according to

$$y_j^* := \begin{cases} 1 & \text{if } y_j \geq F_{TH} \\ 0 & \text{otherwise.} \end{cases}$$

The set  $\mathbb{X}$  is then partitioned into a training set  $\mathbb{S}$  and test set  $\mathbb{T}$ , where  $\mathbb{X} = \mathbb{S} \cup \mathbb{T}$  and  $\mathbb{S} \cap \mathbb{T} = \emptyset$ . The training set is used to train the model. Once this is done, the test set is used to validate the model and derive critical metrics, e.g., error and model quality. A reasonable split ratio of the initial set into training and test set is 2/3 to 4/5 [113]. The test set should contain at least 30 samples to make an adequate statistical statement concerning error and prediction quality [114]. Such a training/test split approach is called *Hold-Out*.

The set  $\mathbb{S}$  is used to train the model. Depending on the used algorithm,  $\mathbb{S}$  needs normalization or standardization. Two common classification algorithms are trained with the set  $\mathbb{S}$ : the logistic regression and the random forest [115].

Logistic regression is a linear binary classification model categorizing the devices into pass or fail. In order to do that, logistic regression utilizes a linear regression and applies the outcome of the regression model to the so-called sigmoid function. The sigmoid function employs the linear regression outcome, providing a probability value between 0 and 1 [113]. The higher the sigmoid function value, the higher the probability that the current device is classified as pass and vice versa. The model aims to find a hyperplane according to which the devices are classified as pass or fail; that is, the model's outcome.

The logistic regression model  $f_{LR}$  is trained with the set  $\mathbb{S}$  and evaluated with the set  $\mathbb{T}$ . In order to measure how well the models perform, key metrics such as accuracy and F1 Score are used.

In contrast to the logistic regression, a more advanced model is also applied for the

classification problem: the random forest. The random forest is a decision tree-based classifier. It is established on the Bagging [116] and uses a randomized feature selection to build multiple uncorrelated decision trees, which are combined in the random forest model [117]. The ease of implementation, low computational effort, and good generalization [117] make the random forest a good-fitting model for performance screening.

The random forest model  $f_{RF}$  is trained with the set  $S$  and evaluated with the set  $T$ . In addition to the accuracy and F1 Score to evaluate the model, the trained random forest model also provides a feature importance ranking. Such ranking indicated which ROs are more critical for the model.

Depending on the features used (functional path ROs or SMONs), the trained classification models provide different results, which in turn indicate how well the features are suited for performance screening.

This classification model-based performance screening is always tailored to the determined  $F_{TH}$ . If the  $F_{TH}$  changes, the entire model must be retrained and replaced. In contrast, regression-based performance screening works differently.

**Scoring metrics for Classification:** The scoring metrics for the classification-based performance screening are the accuracy and F1 score. The metrics are calculated using the trained model applied to the evaluation set  $T$ .

The accuracy is based on the nomenclature used in the confusion metrics (see Figure 2.14). This allows a specific device to be classified into four categories based on the actual device and the result of the prediction model:

- TP: the actual device is pass, and the model predicts the device as pass
- TN: the actual device is fail, and the model predicts the device as fail
- FP: the actual device is fail, and the model predicts the device as pass  $\Rightarrow$  Quality issue
- FN: the actual device is pass, and the model predicts the device as fail  $\Rightarrow$  Yield loss

The accuracy of a model is calculated with the formula [92]

$$Accuracy = \frac{True\ positives + True\ negatives}{True\ positives + True\ negatives + False\ positives + False\ negatives} \quad (8.1)$$

applied to the devices in  $T$ . Such an accuracy metric represents the ratio between correctly predicted devices and the population. However, the accuracy does not provide a fair comparison if the data set is unbalanced [118]. In an unbalanced data set, the proportion of passing and failing devices is unequal, which is usually the case in performance screening

since the slow tail of the distribution is screened. Therefore a second metrics is introduced: the **Matthews correlation coefficient (MCC)**.

The **MCC** can be calculated as follows [118]:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP - FP) \cdot (TP + FN) \cdot (TN + FP) \cdot (TN + FN)}} \quad (8.2)$$

The **MCC** is a more recent evaluation score, especially for binary classification problems, as in the present work. Thus the **MCC** is a suitable metric that also considers the prediction accuracy in the corner cases [118].

### 8.3.2. Performance Screening using Regression

In contrast to the previously explained classification approach, the regression approach behaves differently. A regression model is used that does not require any threshold level previous to the training; also, the binary coding is not applicable anymore. The trained regression model is valid for the entire  $F_{MAX}$  range of the set  $\mathbb{X}$ . The threshold level  $F_{TH}$  is applied afterward to the model outcome. This approach aims for a one-fits-all solution independent of the targeted threshold.

The regression model  $f : x \mapsto \hat{y}_j$  uses the numerical values of the features  $x_i$  (**RO** frequencies) and predicts a numerical  $F_{MAX}$  for a particular label  $\hat{y}_j$ . Exactly one model is trained on one  $F_{MAX}$  characterization test  $T_i$ . In order to train the regression model, the same train/test split approach is used as in Section 8.3.1.

The set  $\mathbb{X}$  is split into a training set  $\mathbb{S}$  and test set  $\mathbb{T}$ .  $\mathbb{S}$  is used to train the regression model and  $\mathbb{T}$  is used for validating the model. Whether normalization or standardization of the data must be performed depends on the model algorithm used. The regression algorithm provides the numerical  $F_{MAX}$  in contrast to the classification model, where only a pass/fail decision is the model's outcome. That numerical outcome made the regression approach more suitable for performance screening because a slight change in the threshold level  $F_{TH}$  does not require retraining of the model.

In this work, three regression models are used, which are fundamentally different in their underlying algorithms: *RidgeCV*, *PCA Regression*, and *Random Forest Regression*.

The *RidgeCV* regression  $f_{Ridge}$  combines a built-in K-Fold cross-validation, which is beneficial in overfitting, especially in small data sets and *Ridge Regression*. The built-in K-Fold cross-validation divides the training set into K equal parts. Then the training uses K-1 parts, and the built-in test uses the remaining part of the training set. This procedure is repeated until all K parts are used for the built-in test. Such cross-validation in training provides a more robust model concerning overfitting and generalization. The Ridge regression is a model based on the linear least squares function. It is a more advanced model that efficiently

implements collinearity in the multiple linear regression context [119].

The Principal Component Regression  $f_{PCR}$  combines the **PCA** and linear regression techniques [120]. The **PCA** transforms the given feature set into a new, reduced one. The dimensionality of the new feature set is determined in the PCR settings of the algorithm. The resulting new feature set is then passed to a linear regression algorithm.

Furthermore, the Random Forest Regression  $f_{RFR}$  is an extension for the Random forest classifier. The RFR deals with numerical values and deploys averaging and multiple decision trees to serve as a regression model [115]. Such an algorithm is different from the two models proposed before - which are based on a linear least square approach - the RFR is a tree-based approach.

**Scoring metrics for Regression:** The regression models have slightly different scoring metrics compared with the classification approach. The two widely used metrics are the **mean absolute error (MAE)** and the **root mean square error (RMSE)**, calculated as follows [92]:

$$\begin{aligned} MAE &= \frac{\sum_n |\hat{y} - y|}{N} \\ RMSE &= \sqrt{\frac{\sum_n (\hat{y} - y)^2}{N}}, \end{aligned} \tag{8.3}$$

where  $N$  is the number of devices in the test set  $\mathbb{T}$ . The **RMSE** is less sensitive to a few outliers but is a more sensitive error metric than the **MAE** [92].

In order to compare the error metric across the different test cases and to previous research, the metrics are normalized as follows:

$$\begin{aligned} nMAE &= \frac{MAE}{mean(y)} \\ nRMSE &= \frac{RMSE}{mean(y)}. \end{aligned} \tag{8.4}$$

The above error metrics may not correctly reflect the prediction accuracy of the regression model in some cases, so an additional metric is used: the coefficient of determination, also known as the  $R^2$  score [121].

$$R^2 = 1 - \frac{\sum_n (y - \hat{y})^2}{\sum_n (y - mean(y))^2} \tag{8.5}$$

The regression model makes a good prediction if the  $R^2$  score is between 0 and 1. In such cases, the  $R^2$  score can also be considered identical to the percentage of correctness obtained by the regression [121].

As a result, the scoring metrics are used to evaluate the regression models and the prediction

accuracy using the labeled data set  $\mathbb{X}$  with the different feature sets - the functional path ROs and the **SMONs**. The resulting prediction value  $\hat{y}_j$  is then compared with the target frequency value  $F_{TH}$ . If  $\hat{y}_j \geq F_{TH}$ , the device is pass; otherwise, the device is fail.

However, to ensure automotive quality (see Section 2.7), a guardband is needed for the performance screening with the regression approach.

**Guardband for Regression to ensure automotive Quality:** The guardband was introduced in [24] and denoted as guardband  $G$ . The guardband intends to ensure automotive quality by considering the accuracy of the trained regression model. In other words, the guardband is the value by which the target frequency must be increased to ensure that no more than a certain false positives are in production.

The guardband is calculated with the residual error ( $e = \hat{y}_j - y_j$ ) considering the test set  $\mathbb{T}$ , with  $y_j$  being the measurements and  $\hat{y}_j$  being the predicted values. The resulting distribution of the residual error is assumed to be approximately Gaussian distributed, and the mean( $\mu_e$ ) and standard deviation ( $\sigma_e$ ) are computed. The guardband is considering the six-sigma approach and is defined as:

$$G = \mu_e + 6\sigma_e \quad (8.6)$$

$G$  is reported in the experiments as a percentage of the target threshold frequency  $F_{TH}$ .

Involving the inverse normal distribution, it can be seen that the calculated guardband corresponds with a defective level of below 0.001 ppm. For example, to reach a defect level of 0.1 ppm, a  $5.2\sigma_e$  in Equation 8.6 is required. Nevertheless, the calculated guardband is valid on the utilized test set; deploying  $G$  for the production screening requires statistical methods like bootstrapping to have confidence. However, no such methods are used in this work, and Equation 8.6 is utilized for the calculation.

The screening frequency in automotive quality grade  $F_{SCREEN}$  is calculated with

$$F_{SCREEN} = F_{TH} + G. \quad (8.7)$$

Such screening frequency ensures that the screening quality is following the six-sigma approach. Equation 8.7 suggests that the model's accuracy impacts the later screening frequency since  $G$  is one crucial part. The target is to reduce this guardband by a very accurate performance screening as much as possible - either using the functional path ROs or the **SMONs**.

As shown in Figure 8.5, the previously explained training and evaluation steps are performed on the labeled devices. Once the trained model is accomplished, it is used on unlabeled devices  $\mathbb{U}$ , as they will also occur in later production.

## 8.4. Deploy the Performance Screening

The **RO** frequencies  $x_j$  of the unlabeled devices  $\mathbb{U}$  are provided to the trained model (classification-based or regression-based). The trained model provides a pass/fail decision for the classification-based approach for each device. The regression model provides a numerical frequency, and the pass/fail decision is then made on the  $F_{TH}$  - or for automotive grade screening on  $F_{SCREEN}$ . This approach makes the extensive  $F_{MAX}$  characterization obsolete, since only the **RO** frequencies are used.

Error metrics like those proposed for the training are not usable for unlabeled devices, since the device has no reference label ( $F_{MAX}$ ). The metric used for unlabeled devices is *yield*. The yield is calculated with

$$yield = \frac{\text{Total number of non-defective devices}}{\text{Total number of manufactured devices}}. \quad (8.8)$$

The yield describes the ratio between the non-defective devices and the total number of manufactured devices. The goal is to have a product with a yield close to 100 %. The yield is, therefore, a measure of the entire manufacturing process. In this work, however, only the yield attributable to performance screening is reported. Thus, the non-defective devices in Equation 8.8 are those devices that pass the performance screening.

## 8.5. Results of the Performance Screening

This section reports the performance screening results using the classification and regression approaches. Both approaches are performed using the functional path ROs versus the **SMONs**. The key objective is to analyse how well the implemented functional path ROs perform using them as performance monitors.

The large automotive **MCU** proposed in Part I is manufactured using a 28 nm CMOS technology. The **MCU** contains 22 functional path ROs and an **SMON** module (Section 1.2.1) containing 27 **SMONs**. This section's results slightly deviate from those published in [42] and [49] due to a larger and more mature data set in this work.

This section is divided into four parts:

- Section 8.5.1: The data set that is used for the analysis.
- Section 8.5.2: Results of performance screening using the classification approach.
- Section 8.5.3: Results of performance screening using the regression approach.
- Section 8.5.4: The applied performance screening on unlabeled data.

### 8.5.1. The data set

The data set used for the results consists of the labeled data set  $\mathbb{X}$  and unlabeled data set  $\mathbb{U}$ .  $\mathbb{X}$  consists of 1923 golden devices which are derived from different wafers. Such wafers come from multiple corner lots. Thus a large process variety is covered.

All golden devices  $\mathbb{X}$  undergo the production test flow shown in Figure 8.1 until the final test. Afterwards, the devices are passed to the  $F_{MAX}$  characterization. In each device, 10 **SLT** test cases ( $T_0 - T_9$ ) are performed, and their maximum achievable clock frequency  $F_{max}^{T_i}$  is stored. All  $F_{max}$  values are normalized in this work so that the design's target frequency  $F_{max}$  equals 100 %. All **SLTs** are performed with the worst-case voltage  $V_{crit}$  and temperature  $T_{crit}$  conditions. The results of the  $T_0$  to  $T_9$  are shown in Figure 8.6.

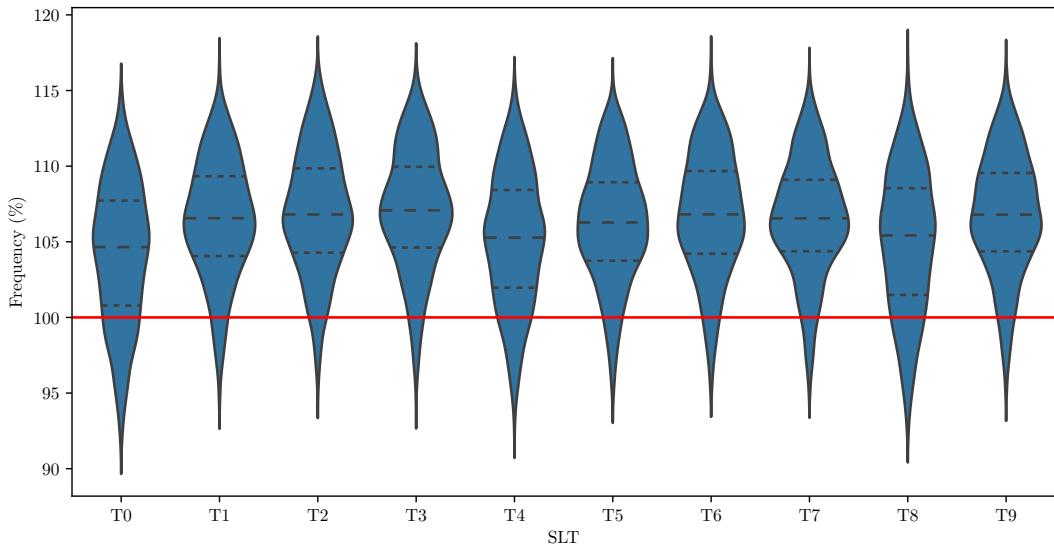


Figure 8.6.: Violin plot of the 10 **SLT** test cases of the  $F_{MAX}$  characterization.

The violin plot shows that all **SLTs** are in the same frequency region. However,  $T_0$  and  $T_8$  are performing slightly worse than the other **SLTs**. Thus  $T_0$  and  $T_8$  are defined as the most critical test cases. The data is already outlier filtered in Figure 8.6; thus, the **SLT** test case might have fewer devices as initially passed through the test. The number of devices in the different test cases after filtering is shown in Table 8.2.

Table 8.2.: Number of devices after the filtering in the **SLT**.

<b>SLT</b>	$T_0$	$T_1$	$T_2$	$T_3$	$T_4$	$T_5$	$T_6$	$T_7$	$T_8$	$T_9$
Devices	1824	1817	1738	1686	1796	1737	1779	1782	1809	1814

In order to evaluate the relation between the **SLTs** and the **ROs**, the **Pearson correlation**

coefficient (PCC) is calculated for all **SLTs** with the 22 functional path ROs to see the 1-to-1 correlation between the features and labels as well as the labels and features among themselves. The results are shown in Figure 8.7.

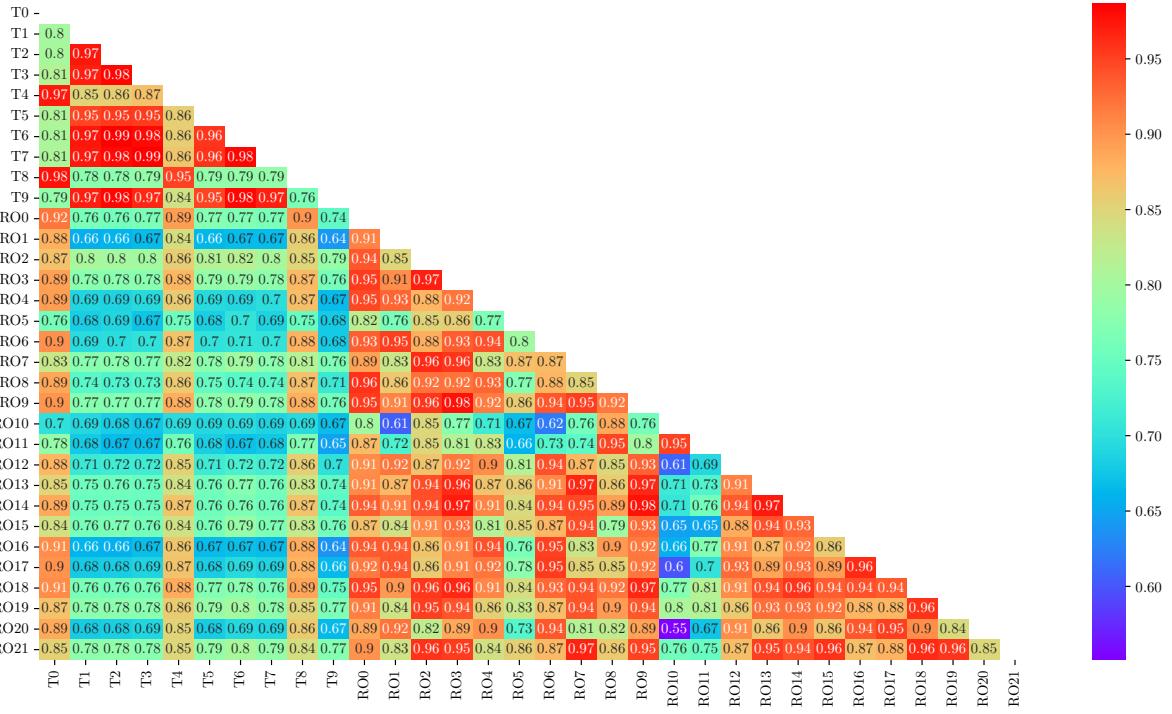


Figure 8.7.: Correlation heatmap between functional path ROs and **SLTs**.

The labels among themselves are presenting a high correlation (upper left section in Figure 8.7). The correlation heatmap shows two significant subgroups within the **SLTs** with a very high correlation.  $T_0$ ,  $T_4$ , and  $T_8$  belong to one subgroup; the remaining **SLTs** belong to the other subgroup. The correlation of the functional path ROs varies more. Especially  $RO_{10}$  and  $RO_{11}$  show a different behavior than the other ROs. The correlation of the functional path ROs concerning the **SLT** is the most critical metric. In order to obtain and create a high-quality ML model, the ROs must correlate as well as possible with the **SLTs**. Especially  $RO_0$  and  $T_0$  show a PCC of more than 92 %.

Figure 8.8 presents the correlation heatmap of the **SMONs** concerning the **SLTs**. The **SMONs** also achieve a PCC in the range of 91 % by looking at  $T_0$ . The overall correlation of the **SMONs** with the **SLTs** is lower than the results with the functional path ROs. Some **SMONs** share the same structure seen in their high correlation between themselves (e.g.,  $SMON_{19}$  to  $SMON_{24}$ ).

The minimum, median, and maximum PCCs are determined from Figure 8.7 and Figure 8.8 and shown in Table 8.3.

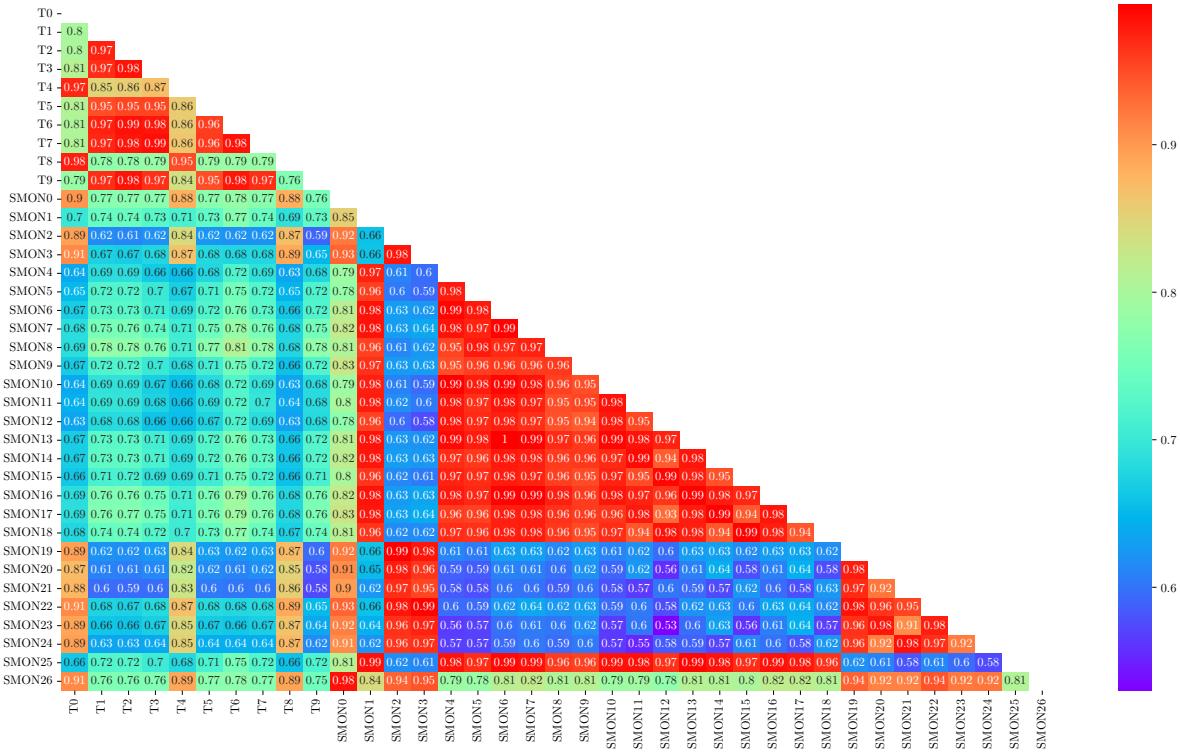


Figure 8.8.: Correlation heatmap between the SMONs and SLTs.

Table 8.3.: PCC comparison between functional path ROs and SMONs.

Label	functional path ROs			SMONs		
	Min	Median	Max	Min	Median	Max
T0	70.0%	88.3%	92.0%	63.4%	68.5%	91.1%
T1	65.6%	74.4%	80.4%	59.5%	71.8%	77.9%
T2	65.7%	74.0%	80.0%	59.4%	72.0%	77.9%
T3	66.5%	74.3%	79.9%	59.6%	69.9%	77.0%
T4	69.4%	85.7%	89.5%	65.6%	70.6%	89.0%
T5	66.2%	75.1%	81.2%	60.3%	71.0%	77.3%
T6	66.6%	75.0%	81.6%	60.2%	75.0%	80.7%
T7	66.6%	74.6%	80.4%	60.1%	71.9%	78.0%
T8	69.4%	85.9%	89.9%	63.0%	78.2%	89.1%
T9	63.8%	72.4%	78.7%	57.5%	71.6%	77.8%

The minimum, median and maximum PCCs of the functional path ROs are higher in all SLT than the SMONs. The functional path ROs exceed the SMONs if only the PCC is considered.

### 8.5.2. Classification-based Performance Screening

The classification-based performance screening is trained with the labeled data set  $\mathbb{X}$  using a dedicated frequency  $F_{TH}$ . The  $F_{TH}$  is normalized to 100 %. The classification model is trained with the functional path ROs as features and the **SMONs** as features, respectively. For each **SLT** test case, a separate model is trained and evaluated. The results using the functional path ROs are presented in Table 8.4.

Table 8.4.: Results of the classification-based **ML** models using functional path ROs.

Label	Logistic Reg.						Random Forest					
	TP	TN	FP	FN	ACC	MCC	TP	TN	FP	FN	ACC	MCC
T0	336	50	47	23	84.65%	0.50	342	80	17	17	92.54%	0.78
T1	435	2	14	4	96.04%	0.19	438	5	11	1	97.36%	0.50
T2	420	5	9	1	97.70%	0.53	419	5	9	2	97.47%	0.49
T3	402	4	12	4	96.21%	0.34	405	4	12	1	96.92%	0.44
T4	376	24	36	13	89.09%	0.45	380	41	19	9	93.76%	0.71
T5	412	3	18	2	95.40%	0.28	411	10	11	3	96.78%	0.59
T6	430	1	14	0	96.85%	0.25	427	4	11	3	96.85%	0.38
T7	420	2	21	3	94.62%	0.17	421	4	19	2	95.29%	0.32
T8	361	35	44	13	87.42%	0.50	362	61	18	12	93.38%	0.76
T9	436	2	13	3	96.48%	0.22	435	2	13	4	96.26%	0.19

The calculated accuracy in the **SLT** cases ranges between 84.65 % and 97.70 % deploying the logistic regression and between 92.54 % and 97.47 % for the random forest. The accuracy in  $T_0$ ,  $T_4$ , and  $T_8$  is significantly lower than in the other **SLTs**.

The **MCC** also has a substantial deviation. However, the random forest model has a higher **MCC** score than the logistic regression, which also applies to the comparison of the accuracy of the two models. Therefore, the random forest is the preferable model in nearly all **SLTs**.

The results using the **SMONs** as features are presented in Table 8.5.

There is no significant difference compared with the functional path ROs. In some test cases, the **SMONs** perform better; in other test cases, the functional path ROs are better.

Therefore a third run uses the combined features set (functional path ROs and **SMONs**). The results are shown in Table 8.6.

Also, in this case, the accuracy and **MCC** depend on the **SLT**. There is a slight benefit, e.g., in  $T_0$ , with respect to both scoring metrics. The **MCC** overall shows slightly higher results in the combined data set.

The functional path ROs and the **SMONs** provide good accuracy in the performance screening using the classification-based approach. The classification-based screening provides a satisfactory result to get a first estimation of the accuracy achieved in the performance

Table 8.5.: Results of the classification-based [ML](#) models using [SMONs](#).

Label	Logistic Reg.						Random Forest					
	TP	TN	FP	FN	ACC	MCC	TP	TN	FP	FN	ACC	MCC
T0	330	62	35	29	85.96%	0.57	338	78	19	21	91.23%	0.74
T1	434	5	11	5	96.48%	0.37	436	7	9	3	97.36%	0.54
T2	418	2	12	3	96.55%	0.22	419	2	12	2	96.78%	0.25
T3	403	1	15	3	95.73%	0.11	404	6	10	2	97.16%	0.52
T4	363	31	29	26	87.75%	0.46	380	44	16	9	94.43%	0.75
T5	408	8	13	6	95.63%	0.45	412	11	10	2	97.24%	0.65
T6	427	7	8	3	97.53%	0.56	427	9	6	3	97.98%	0.66
T7	421	5	18	2	95.52%	0.38	421	3	20	2	95.07%	0.26
T8	362	64	15	12	91.39%	0.68	362	64	15	12	94.04%	0.79
T9	436	6	9	3	97.36%	0.50	436	5	10	3	97.14%	0.44

Table 8.6.: Results of the classification-based [ML](#) models using the combined data set.

Label	Logistic Reg.						Random Forest					
	TP	TN	FP	FN	ACC	MCC	TP	TN	FP	FN	ACC	MCC
T0	334	72	25	25	89.04%	0.67	340	81	16	19	92.23%	0.77
T1	436	5	11	3	96.92%	0.42	438	7	9	1	97.80%	0.61
T2	415	3	11	6	96.09%	0.25	420	3	11	1	97.24%	0.39
T3	400	4	12	6	95.73%	0.30	406	5	11	0	97.39%	0.55
T4	361	38	22	28	88.86%	0.54	378	41	19	11	93.32%	0.70
T5	409	8	13	5	95.86%	0.46	413	11	10	1	97.47%	0.68
T6	429	5	10	1	97.53%	0.52	428	7	8	2	97.75%	0.59
T7	420	6	17	3	95.52%	0.40	422	5	18	1	95.74%	0.41
T8	363	56	23	11	92.49%	0.73	365	64	15	9	94.70%	0.81
T9	433	5	10	6	96.48%	0.37	436	3	12	3	96.70%	0.30

screening. The classification-based accuracy of the approach is also strongly dependent on the [SLT](#) used.

### 8.5.3. Regression-based Performance Screening

The regression-based performance screening is utilized for the labeled data set  $\mathbb{X}$ . In contrast to the classification-based approach, the regression model provides a numerical performance value which is the precise performance value of the [SLTs](#). Three different [ML](#) models are separately trained on the data set. A scatter plot shows the predicted performance value vs. the measured performance value from the labeled devices. The scatter plot using the functional path ROs as features and the RidgeCV regression is shown in Figure 8.9 employing

SLT T0.

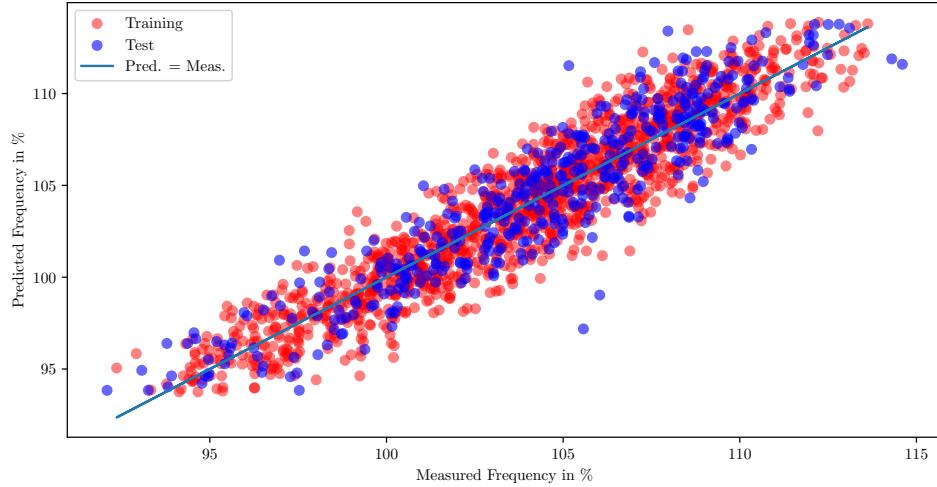


Figure 8.9.: Measured vs. predicted performance for T0.

The solid line within the scatter plot represents the ideal line where the model's predicted frequency equals the  $F_{MAX}$  characterization's measured frequency. Each dot represents one device in the scatter plot; the red devices belong to the training set  $S$ , and the blue to the validation set  $T$ . The nRMSE of this scatter plot is 1.68 %.

The results of the different ML models using the functional path ROs and the SMONs are presented in Table 8.7 and Table 8.8.

Table 8.7.: Prediction accuracy on the functional path ROs.

Label	RidgeCV				PCA Reg.				Rand. For.			
	nMAE	nRMSE	$R^2$	G	nMAE	nRMSE	$R^2$	G	nMAE	nRMSE	$R^2$	G
T0	1.34%	1.68%	85.3%	10.5%	1.42%	1.80%	82.9%	11.3%	1.36%	1.71%	84.7%	10.8%
T1	1.66%	2.04%	64.7%	13.0%	1.67%	2.06%	64.9%	12.9%	1.71%	2.15%	61.6%	13.6%
T2	1.63%	2.03%	66.9%	13.0%	1.68%	2.08%	65.7%	13.3%	1.80%	2.22%	61.1%	14.2%
T3	1.65%	2.09%	68.0%	13.5%	1.67%	2.12%	67.7%	13.5%	1.79%	2.26%	63.3%	14.3%
T4	1.41%	1.77%	78.7%	11.2%	1.45%	1.83%	77.2%	11.6%	1.44%	1.77%	78.4%	11.1%
T5	1.50%	1.87%	67.9%	12.0%	1.55%	1.95%	65.6%	12.2%	1.60%	2.01%	63.4%	12.6%
T6	1.61%	2.00%	67.6%	12.8%	1.63%	2.04%	67.0%	13.0%	1.68%	2.12%	64.2%	13.5%
T7	1.55%	1.97%	65.5%	12.7%	1.62%	2.06%	63.1%	12.9%	1.74%	2.20%	58.5%	13.5%
T8	1.34%	1.68%	85.4%	10.7%	1.34%	1.69%	85.3%	10.7%	1.34%	1.69%	85.3%	10.4%
T9	1.66%	2.06%	66.2%	13.2%	1.73%	2.12%	64.4%	13.6%	1.78%	2.19%	62.3%	13.8%

Regression models for both feature sets provide the lowest error in T0, T4, and T8; in such SLTs, the classification approaches (see Section 8.5.2) performed worst. Comparing the results of the functional path ROs with the SMONs, it can be seen that the SMONs provide a lower

Table 8.8.: Prediction accuracy on the SMONs.

Label	RidgeCV				PCA Reg.				Rand. For.			
	nMAE	nRMSE	$R^2$	G	nMAE	nRMSE	$R^2$	G	nMAE	nRMSE	$R^2$	G
T0	1.32%	1.68%	85.4%	10.3%	1.41%	1.78%	83.4%	11.1%	1.38%	1.76%	83.9%	11.1%
T1	1.32%	1.68%	76.0%	10.7%	1.36%	1.73%	74.9%	10.9%	1.53%	1.92%	69.5%	12.1%
T2	1.44%	1.80%	74.0%	11.4%	1.46%	1.85%	72.6%	11.9%	1.63%	2.05%	66.6%	13.2%
T3	1.39%	1.78%	76.7%	11.4%	1.44%	1.85%	75.3%	11.8%	1.69%	2.15%	67.1%	13.6%
T4	1.37%	1.73%	79.5%	10.9%	1.40%	1.76%	78.9%	11.1%	1.43%	1.79%	78.0%	11.3%
T5	1.29%	1.61%	76.3%	10.0%	1.32%	1.65%	75.3%	10.3%	1.45%	1.81%	70.3%	11.5%
T6	1.37%	1.71%	76.5%	10.9%	1.39%	1.74%	75.7%	11.1%	1.53%	1.94%	70.2%	12.3%
T7	1.41%	1.75%	72.6%	11.2%	1.42%	1.76%	73.0%	11.0%	1.61%	2.06%	63.5%	12.8%
T8	1.37%	1.69%	85.1%	10.7%	1.40%	1.71%	84.8%	10.9%	1.39%	1.73%	84.6%	10.7%
T9	1.39%	1.71%	76.5%	10.9%	1.42%	1.76%	75.2%	11.4%	1.62%	1.99%	68.8%	12.7%

error in nearly all **SLTs**. However, in **SLT T8**, the functional path ROs show a lower error.

Also, a third run is performed with the combined data set in the regression-based screening. The results of the combined feature set are shown in Table 8.9.

Table 8.9.: Prediction Accuracy on the Combined data set.

Label	RidgeCV				PCA Reg.				Rand. For.			
	nMAE	nRMSE	$R^2$	G	nMAE	nRMSE	$R^2$	G	nMAE	nRMSE	$R^2$	G
T0	1.25%	1.58%	87.0%	9.7%	1.38%	1.74%	84.1%	10.9%	1.32%	1.67%	85.5%	10.6%
T1	1.29%	1.63%	77.5%	10.4%	1.41%	1.77%	73.9%	11.1%	1.52%	1.93%	69.1%	12.1%
T2	1.33%	1.68%	77.3%	10.7%	1.46%	1.84%	73.1%	11.8%	1.62%	2.03%	67.4%	13.0%
T3	1.30%	1.68%	79.2%	10.8%	1.43%	1.85%	75.5%	11.7%	1.65%	2.12%	67.9%	13.4%
T4	1.28%	1.62%	82.1%	10.2%	1.39%	1.74%	79.3%	11.0%	1.39%	1.71%	79.8%	10.8%
T5	1.21%	1.50%	79.3%	9.4%	1.34%	1.69%	74.1%	10.6%	1.47%	1.84%	69.4%	11.6%
T6	1.25%	1.55%	80.6%	9.9%	1.40%	1.77%	75.1%	11.3%	1.47%	1.89%	71.7%	12.0%
T7	1.31%	1.64%	76.1%	10.5%	1.40%	1.78%	72.4%	11.1%	1.55%	2.02%	65.0%	12.4%
T8	1.27%	1.58%	87.0%	10.0%	1.30%	1.63%	86.3%	10.3%	1.34%	1.66%	85.9%	10.2%
T9	1.27%	1.60%	79.6%	10.1%	1.46%	1.81%	74.1%	11.6%	1.59%	1.98%	69.2%	12.5%

In all **SLTs**, the **nMAE** and **nRMSE** are lowered compared with the separate feature set. The regression models can utilize the benefits of the functional path ROs and the **SMONs** to reduce the error further and minimize the screening guardband. The comparison of the screening guardband needed is presented in Figure 8.10. Also, the screening guardband can be reduced by using the combined features set in all **SLTs**.

A characteristic of the random forest algorithm is the built-in feature importance ranking based on the mean decrease in impurity. That provides an overview of the features' importance in the trained **ML** model. In order to get an overview of the contributions and importance of the **ROs**, the cumulated mean decrease in impurity is calculated for all **SLTs**

## 8. Performance Screening Using Functional Path RO

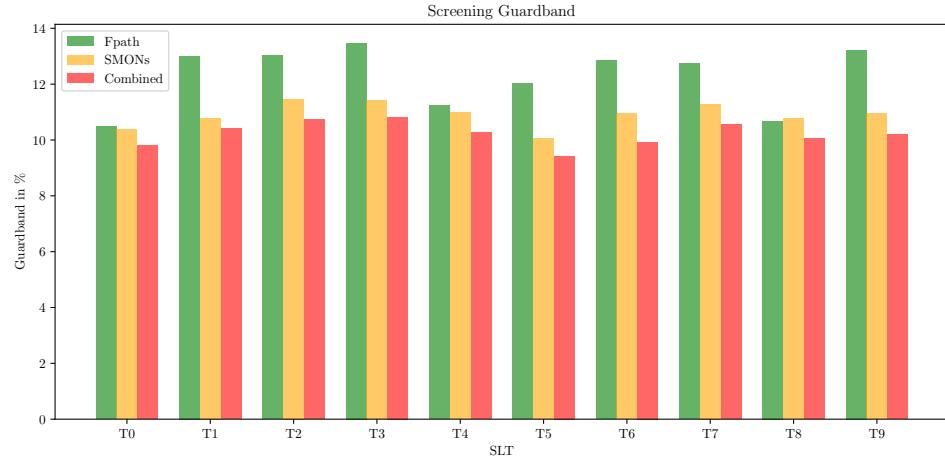


Figure 8.10.: Guardband comparison using the RidgeCV regression.

and shown in Figure 8.11. The combined feature set from Table 8.9 is used in this analysis.

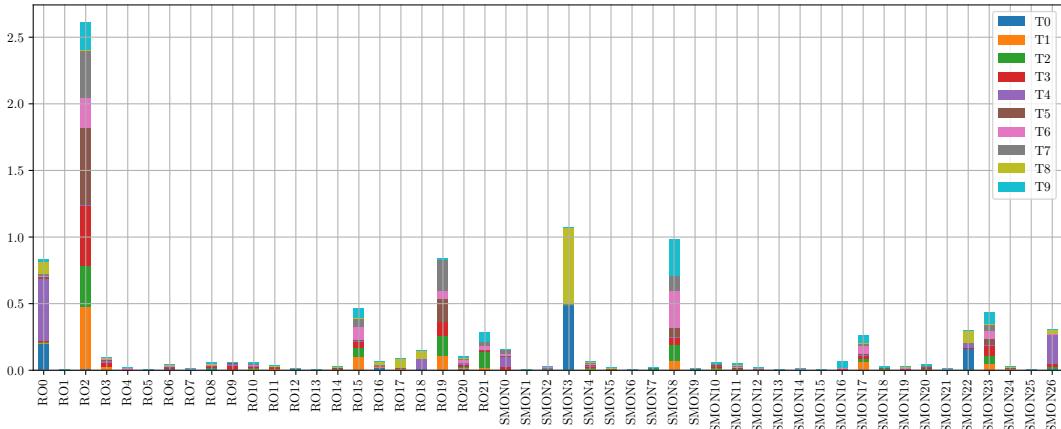


Figure 8.11.: Cumulated RO importance.

The cumulated mean decrease in impurity reveals that some dedicated ROs (RO0, RO2, SMON3, and SMON8) have substantial importance in predicting some SLTs.

The regression-based results suggest that the functional path ROs provide good performance screening results; however, the SMONs in most SLTs give a more satisfactory result. A reasonable performance screening can be made with both types of ROs, which meet the automotive quality requirements. Performance screening with the highest accuracy is achieved by combining the two sets of ROs. Thus, the functional path ROs are one central pillar in the performance screening flow.

#### 8.5.4. Applied Performance Screening on unlabeled Data

This section deploys the developed [ML](#) models using the regression-based approach to unlabeled data  $\mathbb{U}$ . Then the yield is calculated based on the performance screening results. This analysis is only done for the regression-based performance screening using the RidgeCV since the calculated guardband is investigated to achieve automotive quality.

Each manufactured device is, by default, an unlabeled device since the [RO](#) measurements are part of the test program. However, the unlabeled devices are produced in the engineering phase, in which the process is less stable than in the later high-volume production. Therefore, this estimation and analysis do not represent the later production screening from the yield perspective. Also,  $F_{TH}$  and the resulting  $F_{SCREEN}$  are theoretically selected. In this experimental performance screening, six lots that have been manufactured under similar process conditions are selected. Thus, the selected devices have a first approximation of natural process variation as found in later production.  $\mathbb{U}$  contains 13 565 devices from the 6 production lots.

The functional path RO frequencies of the unlabeled devices are then passed to the pre-trained [ML](#) model, and the performance of each device is predicted. The results of using the functional path ROs and  $T_0$  are shown in Figure 8.12 for the RidgeCV algorithm.

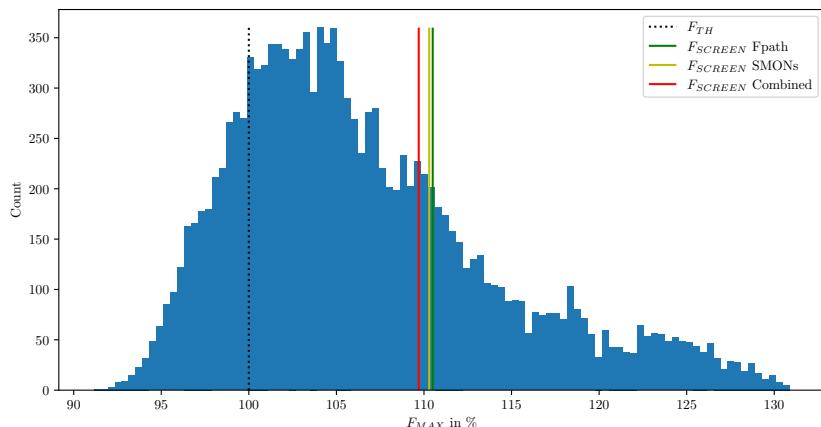


Figure 8.12.: Guardband comparison RidgeCV regression for  $T_0$ .

The dotted line is the defined  $F_{TH}$  for which the performance screening is set. The colored vertical lines divide the resulting distribution into pass and fail. The different colors are the resulting  $F_{SCREEN}$  for the different feature sets: the functional path ROs, the [SMONs](#), and the combined feature set. All devices on the left side of this screening border are fail, and the pass devices are on the right. In particular, the screening frequency using the combined feature caused a significant shift left of the border - since the prediction does have a higher accuracy. This can also be seen in the calculated performance yield. The performance-related yield is

calculated for the unlabeled devices in all **SLTs**. The  $F_{SCREEN}$  is calculated using Equation 8.7, and the yield is calculated using Equation 8.8. The results are shown in Table 8.10.

Table 8.10.:  $F_{SCREEN}$  and yield estimation for performance screening.

Label	$F_{SCREEN}$			Yield			Yield Gain (Combined vs. <b>SMONs</b> ) (Pct. Points)
	Fpath	SMON	Combined	Fpath	SMON	Combined	
T0	110.5%	110.3%	109.7%	25.4%	26.0%	28.6%	2.6
T1	113.0%	110.7%	110.4%	20.4%	31.9%	34.4%	2.5
T2	113.0%	111.4%	110.7%	21.4%	31.1%	34.8%	3.7
T3	113.5%	111.4%	110.8%	20.0%	31.3%	34.7%	3.4
T4	111.2%	110.9%	110.2%	24.5%	26.2%	29.3%	3.1
T5	112.0%	110.0%	109.4%	24.3%	34.2%	37.6%	3.4
T6	112.8%	110.9%	109.9%	21.9%	33.2%	37.9%	4.7
T7	112.7%	111.2%	110.5%	19.5%	28.0%	32.7%	4.7
T8	110.7%	110.7%	110.0%	27.7%	27.7%	30.4%	2.7
T9	113.2%	110.9%	110.1%	19.2%	31.8%	36.1%	4.3

The **ML** model accuracy and the resulting  $F_{SCREEN}$  are in the same range for  $T0$ ,  $T4$ , and  $T8$  using the functional path ROs and the **SMONs**, as highlighted in the previous section. In the other **SLTs**, the **SMONs** provide a better prediction and, therefore, a lower  $F_{SCREEN}$ , resulting in a higher yield. The right column in Table 8.10 shows the yield gain of the combined data set compared with the **SMONs**. Thus, the positive effect of the functional path ROs on the yield becomes evident here. For every **SLT**, a yield gain of 2.5 up to 4.7 percentage point is achieved by using the combined data set. Thus the functional path ROs act as a supporting structure to make the performance screening more accurate and help to increase the yield.

## 9. POST-SI Summary

The [Post-Silicon](#) part presents the activities once the manufactured [MCU](#) contains the implemented functional path ROs. Chapter 7 presents how the functional path ROs are measured and validates the quality of the measurements. The results show that the functional path ROs are accurately measured with a high reproducibility on the [ATE](#).

Chapter 8 elaborates on using the functional path RO as a performance monitor. The basics of performance screening and the data set needed for that are discussed. ML is used to determine the performance of an [MCU](#) based on indirect monitors such as functional path ROs. In particular, two approaches are presented: the classification-based performance screening and the regression-based performance screening. For both approaches, the functional path ROs are used for performance screening, and in order to compare the functional path RO results with a benchmark, the [SMON](#) module is used in the same manner for performance screening. The combined dataset is used in the third use case, which means functional path RO and the [SMON](#) module. Also, several algorithms are used in the ML. The results of classification-based performance screening show good accuracy in performance screening with all data sets used. In some test cases, the functional path ROs are better; in others, the [SMONs](#). Whereas with the regression-based approach, the results change. First of all, both structures (functional path ROs and [SMONs](#)) provide good accuracy in performance screening. However, the [SMONs](#) show slightly better results than the functional path ROs in most test cases. The significant advantage is the combined data set. The combination of functional path ROs and [SMONs](#) in performance screening improved the accuracy of performance screening in all test cases, resulting in a lower screening guardband. The impact of the reduced screening guardband is evaluated on a more considerable amount of production data, and the resulting yield is shown. Using the functional path ROs as supporting monitors in the combined data set reduces the yield loss by 2.5 up to 4.7 percentage points. This indicates the significant advantage of the functional path ROs as performance monitors especially by combining them with dedicated [SMONs](#).

However, it should be mentioned here that only 22 functional path ROs were used in this analysis; the number of implemented [ROs](#) can be increased due to the scalable implementation.

# 10. Conclusion

## 10.1. Summary of Key Contributions

The automotive [Microcontrollers \(MCUs\)](#) have high-quality requirements. This is the reason for the extensive testing of such [MCUs](#), and one crucial test is the performance screening. The performance screening is a challenging task and requires effort. A well-established and accurate methodology uses indirect performance monitors, namely the [Ring oscillators \(ROs\)](#). However, the accuracy of the performance screening depends strongly on the structure of such [ROs](#). Furthermore, traditional [ROs](#) add a significant area and leakage overhead to the design. In this work, the functional path [ROs](#) are presented, which are on-chip integrated monitors for performance screening. This work is divided into two main parts: Part [I](#) - the [Pre-Silicon](#) part, and Part [II](#) - the [Post-Silicon](#) part.

Part [I](#) presents the implementation of functional path [ROs](#), including methodologies to improve the functional path [ROs](#) regarding routing and efficiency. The main contributions of this work in that part are:

- The scalable implementation of the functional path [RO](#) in an industrial design, which leads up to 96 % savings in terms of leakage and area compared to the traditional [RO](#) structures.
- Two advanced implementation methods to reduce the implementation effort: the concept of natural loops and the Best Paper Award- winning concept of self-enabling.
- The self-enabling approach which leads to a significant reduction in routing effort.
- The path selection flow selects promising functional paths for [RO](#) implementation with minimal turnaround time and compatibility with industrial design flows.
- The path selection flow is validated through analog simulation and sensitivity analysis to continuously improve the path selection flow in the specified [PVT](#) space.

In Part [II](#), the implemented functional path [ROs](#) on silicon are measured and validated. Ultimately, their frequencies are used for the performance screening. The main contributions of this work in that part are:

- The measured functional path RO frequencies have high quality and repeatability and meet the expected pre-silicon simulation results.
- The measured frequencies are used for the **ML**-based performance screening that provides a high accuracy.
- The **ML** problem is modeled as a classification problem and regression problem, and the accuracy using the functional path ROs is discussed.
- The performance screening accuracy can be significantly improved by combining the functional path ROs with the **SMONs**.

Finally, the functional path ROs can compete with the traditional **SMONs** in terms of prediction accuracy in all test cases. The Functional path ROs are considerably more efficient than traditional **SMONs**, particularly regarding leakage, area consumption, and routing effort. Combining a few dedicated **SMONs** and functional path ROs provides outstanding performance monitors for highly accurate screening. Thus, the functional path ROs contribute to producing more sustainable and economical **MCUs**.

## 10.2. Obstacles in an Industrial Context

In addition to the many advantages of functional path ROs, such structures can lead to obstacles in the industrial environment that should be mentioned.

The **ECO** implementation of the functional path ROs can lead to an obstacle - however, it is only applicable in the industrial context. Ideally, the **MCU** design passes through the design flow step by step, the **ECO** phase fixes all issues, and the functional path ROs are implemented. In reality, there is a time to market pressure, which accelerates the design flow. Thus, the timeline becomes tighter in the **ECO** phase - one of the last steps before tape-out. The tight timeline results in the optimized **ECO** phase, and the functional path RO implementation is executed with some other **ECO** steps in parallel. The impact is shown in Section 5.2.1, which has not had a significant impact in the test case in this thesis. However, that could be changed in other designs. Therefore, carefully planning the functional path RO implementation via **ECO** is necessary.

Another potential obstacle is the circumstance of the path delay fault model for path sensitization. In this work, there were no barriers due to a large automobile **MCU** with an extensive **DFT** environment. However, this might change in other designs. In this work, many functional paths can be sensitized using a commercial **ATPG** tool in path delay fault mode. The number of sensitizable paths and the used scan environment are crucial for this path

sensitization. Therefore, the functional path RO approach may have limited applicability for designs with few sensitizable paths.

### 10.3. Future Work

The thesis delivers a proof of concept for implementing and using functional path ROs. Even the 22 implemented functional path ROs improve the performance screening and increase the yield. Future MCUs products shall have several hundreds of functional path ROs, including the self-enabling mechanism and all architectural and conceptional methodologies presented in this work. In addition, the path selection process shall be continually improved with the sensitivity-based approach proposed in Section 5.3. With such improvements, a further decrease in the screening guardband is possible.

As mentioned in Section 10.2, the sensitization using commercial ATPG tools in path delay fault mode can lead to obstacles. In order to overcome this obstacle, a particular path sensitization mode for functional path ROs is meaningful. That additional mode will not be forced to ensure path sensitization during clock events since the oscillation of the functional path ROs is clock-independent. With such ATPG mode in place, more functional paths can be enabled for the RO implementation.

The ATE is used to sensitize and measure the functional path ROs. Such utilization allows the measurement of functional path ROs in the initial production state on ATE. An in-field usage of the functional path ROs can be enabled by using a deterministic scan pattern on-chip. Methods to do this are proposed in [122]. Using such an approach, path sensitization and in-field measurement of the functional path RO frequencies is possible using an on-chip counter. Such an approach can enable functional path RO for predictive maintenance and lifecycle monitoring.

In the end, functional path ROs are promising and efficient structures for performance screening, and they are even more powerful by utilizing ML approaches. The further work proposal indicates the potential of functional path ROs, either for more advanced performance screening or emerging applications for in-field usage.

# List of Figures

1.1. Process window after tightening the limits. . . . .	3
1.2. Test approaches for $F_{MAX}$ testing. . . . .	4
1.3. Outline of the thesis. . . . .	7
2.1. Development flow of a prototype MCU divided into Pre-Silicon and Post-Silicon. . . . .	8
2.2. Synchronous combinational logic path with launch and capture FF. . . . .	10
2.3. Timing diagram of a capture FF in a synchronous digital circuit illustrating setup and hold time and slack. . . . .	11
2.4. Wafermap shows the D2D variations from the frequency of an RO. . . . .	13
2.5. Relation between PVT variations and performance. . . . .	16
2.6. Corner cases of a CMOS transistor. . . . .	17
2.7. Scan FF contains an ordinary FF and a MUX. . . . .	20
2.8. Scan insertion replaces three FFs with scan FFs and connects them to a scan chain. . . . .	20
2.9. Scan test procedure with a capture pulse. . . . .	21
2.10. Scan compression allows the parallel loading of scan chains with a decompressor and compactor through a single pin. . . . .	21
2.11. Scan test using a path delay fault model. . . . .	22
2.12. Basic principle of an RO using inverter gates. . . . .	25
2.13. ML overview of approaches in this work. . . . .	27
2.14. Confusion matrix of a classification problem. . . . .	29
3.1. Example of a functional path RO. Adapted from [40] © IEEE 2021. . . . .	32
3.2. Path sensitization of a functional path RO with the scan architecture. Adapted from [40] © IEEE 2021. . . . .	33
3.3. The test sequence during the scan test pattern. Adapted from [40] © IEEE 2021. . . . .	34
3.4. Four path topologies of functional paths. . . . .	36
3.5. Basic concept of the natural loop approach. . . . .	37
3.6. The basic principle of the self-enabling approach of a functional path RO. Adapted from [41] © IEEE 2022. . . . .	38
3.7. The library gate called RO-MUX. Adapted from [49] © IEEE 2023. . . . .	40

3.8. XOR-tree for compacting the observe signals and forwarding it to a GPIO. Adapted from [49] © IEEE 2023. . . . .	41
3.9. Detailed implementation of Option 1. Adapted from [49] © IEEE 2023. . . . .	41
3.10. Option 2 - the indirect self-enabling controlled by three surrounding scan FFs. . . . .	42
3.11. Basic control infrastructure for the functional path ROs with 8 ports. . . . .	44
3.12. The control infrastructure for the functional path ROs. Adapted from [49] © IEEE 2023. . . . .	45
3.13. Implementation flow of functional path RO. Adapted from [49] © IEEE 2023. . . . .	47
3.14. Implementation on layout based on <i>Path a</i> after ECO. Adapted from [40] © IEEE 2021. . . . .	51
3.15. Implementation on layout based on <i>Path b</i> after ECO. Adapted from [40] © IEEE 2021. . . . .	51
3.16. Sensitizable paths per pattern for three modules. . . . .	54
3.17. Candidates for natural loops. . . . .	55
3.18. Routing visualization of 8 sample paths implemented in Option 0. . . . .	57
3.19. Routing visualization of 8 sample paths implemented in Option 1. . . . .	58
4.1. Physical-aware functional path selection flow. . . . .	61
4.2. Elbow plots of the three modules. . . . .	64
4.3. Violin plots presenting the distribution of paths on Module B according to the defined path characteristics. . . . .	65
5.1. Overview of the Section Pre-Silicon verification and validation. . . . .	68
5.2. Digital simulation snapshot at the beginning of the oscillation. Adapted from [49] © IEEE 2023. . . . .	69
5.3. Digital simulation sequence with 6 functional path ROs. . . . .	70
5.4. Limits from the digital simulation using the worst and best case SDF. . . . .	71
5.5. Path characteristic pre- and post-tape-out. . . . .	73
5.6. Ansys RedHawk-SC simulation on Module A including the visualization of the implemented functional path ROs. . . . .	74
5.7. SPICE Model extractor with data in- and output. Adapted from [42] © IEEE 2022. . . . .	76
5.8. Two alternatives of the extracted analog SPICE models. Adapted from [42] © IEEE 2022. . . . .	77
5.9. Voltage and temperature sensitivity of the functional path ROs. Adapted from [42] © IEEE 2022. . . . .	79
5.10. Variance of the frequency sensitivity with respect to the individual process parameters of all selected functional paths ROs. Adapted from [42] © IEEE 2022. .	80

5.11. Variance of the frequency sensitivity of the selected functional paths ROs with respect to all process parameters. Adapted from [42] © IEEE 2022. . . . .	81
5.12. Variance of the delay sensitivity of the randomly selected sensitizable functional paths with respect to all process parameters. Adapted from [42] © IEEE 2022. . . . .	81
7.1. Repetitive measurement of the functional path ROs on a normalized scale. . . . .	87
7.2. Coefficient of variation of 100 repetitive measurements. . . . .	88
7.3. Measurement data from 3858 devices of 25 wafers and the simulated test limits. . . . .	89
7.4. Voltage drop measurements. . . . .	89
7.5. Wafermap of the frequency distribution of an functional path RO. . . . .	90
8.1. Test flow of an MCU including the performance screening. . . . .	91
8.2. $F_{MAX}$ characterization results from a wafer after the SLT. . . . .	94
8.3. Data set structure from unlabeled and labeled devices. . . . .	95
8.4. Performance screening process basics. . . . .	96
8.5. Performance screening process learning and deploying. . . . .	96
8.6. Violin plot of the 10 SLT test cases of the $F_{MAX}$ characterization. . . . .	105
8.7. Correlation heatmap between functional path ROs and SLTs. . . . .	106
8.8. Correlation heatmap between the SMONs and SLTs. . . . .	107
8.9. Measured vs. predicted performance for $T_0$ . . . . .	110
8.10. Guardband comparison using the RidgeCV regression. . . . .	112
8.11. Cumulated RO importance. . . . .	112
8.12. Guardband comparison RidgeCV regression for $T_0$ . . . . .	113

# List of Tables

3.1. Properties of Option 0, Option 1, and Option 2 . . . . .	44
3.2. Basic information of the MCU modules . . . . .	49
3.3. Path characteristics of <i>Path a</i> and <i>Path b</i> . . . . .	50
3.4. Estimated area overhead and leakage increase of the control infrastructure . . . . .	52
3.5. Area overhead and leakage increase of 128 functional path ROs in comparison to an equal sized SMON module . . . . .	52
3.6. Path analysis analysing the structure of the functional paths . . . . .	53
3.7. ATPG results reveal the number of patterns necessary for path sensitization . . . . .	54
3.8. Number of paths suitable for functional path RO implementation . . . . .	56
3.9. Routing reduction of the RO implementation for <i>Option 1</i> . . . . .	57
3.10. Routing reduction of the RO implementation for <i>Option 1</i> including the general enable signal . . . . .	59
3.11. Paths with nearby FFs per module . . . . .	59
4.1. Number of paths per bucket after the K-means clustering . . . . .	65
4.2. Final selected functional path to be implemented as an RO . . . . .	67
5.1. Deviation of the implemented functional path ROs pre- and post-tape-out . . . . .	72
8.1. Differences and similarities between the classification approach and the regression approach . . . . .	98
8.2. Number of devices after the filtering in the SLT . . . . .	105
8.3. PCC comparison between functional path ROs and SMONs . . . . .	107
8.4. Results of the classification-based ML models using functional path ROs . . . . .	108
8.5. Results of the classification-based ML models using SMONs . . . . .	109
8.6. Results of the classification-based ML models using the combined data set . . . . .	109
8.7. Prediction accuracy on the functional path ROs . . . . .	110
8.8. Prediction accuracy on the SMONs . . . . .	111
8.9. Prediction Accuracy on the Combined data set . . . . .	111
8.10. $F_{SCREEN}$ and yield estimation for performance screening . . . . .	114

# Acronyms

**ATE** automatic test equipment 4, 24, 26, 34, 69, 87, 88, 115, 118

**ATPG** automatic test pattern generation 23, 33, 34, 37, 39, 41–44, 48, 54, 56, 83, 117, 118, 122

**BE** back-end 9, 91

**BIST** built-in self-test 19

**CMOS** complementary metal oxide semiconductor 9, 10, 12, 16, 17, 49, 50, 62, 119

**CV** coefficient of variation 88

**D2D** die-to-die 13, 26, 30, 62, 90, 119

**DfT** Design for Testability 9, 19, 24, 33, 35, 38–40, 42, 56, 117

**DPPM** defective parts per million 29, 30

**DRC** Design Rule Check 9

**DUT** device under test 24, 26, 89, 92

**ECO** engineering change order 9, 46, 48, 49, 51, 52, 64, 66, 68, 70–72, 84, 117, 120

**EDA** electronic design automation 16, 17, 23, 41, 48, 52, 58

**FE** front-end 9, 91

**FF** flip-flop 9–11, 20, 21, 23, 35, 36, 38–46, 53, 59, 60, 83, 119, 120, 122

**GPIO** general purpose input/output 43, 44, 69, 70, 87, 88, 90, 93

**HCI** hot carrier injection 18

**IC** integrated circuit 1

**IDT** inverse temperature dependence 15, 16

**JTAG** Joint Test Action Group 19

**L2L** lot-to-lot 13

**LBIST** logic built-in self-test 19

**LEC** Logic Equivalence Check 9, 68

**LOC** launch-off-capture 23, 34

**LOS** launch-off-shift 23

**MAE** mean absolute error 27, 102, 111

**MBIST** memory built-in self-test 19

**MCC** Matthews correlation coefficient 101, 108

**MCU** microcontroller 1–10, 12–15, 17–26, 28, 29, 32–35, 37, 39, 40, 46, 49, 53, 56, 59–61, 66, 68, 69, 71–73, 76, 83, 86, 89–93, 104, 115–119, 121, 122

**ML** machine learning 8, 27–29, 86, 95, 96, 98, 99, 106, 108–111, 113, 114, 117–119, 122

**MUX** multiplexer 20, 32–35, 37–40, 44, 48, 50–52, 69, 89, 90, 119

**NBTI** negative bias temperature instability 18

**PCA** principal component analysis 28, 97, 102

**PCC** Pearson correlation coefficient 105–107, 122

**PDN** power delivery network 14, 15, 73, 75, 84

**Post-Si** Post-Silicon 8, 9, 15, 16, 18, 86, 115, 116, 119

**Pre-Si** Pre-Silicon 8, 9, 16, 18, 68, 75, 83, 88, 90, 116, 119, 120

**PUT** path under test 33, 34

**PVT** Process-Voltage-Temperature 2, 12, 15, 16, 18, 24, 26, 71, 75, 77–80, 84, 92, 116, 119

**RF** radio frequency 92

- RMSE** root mean square error 27, 102, 110, 111
- RO** ring oscillator 3, 5, 13, 25, 26, 32–41, 43–52, 56, 58, 63, 64, 66, 69, 76, 79, 80, 83, 86, 88, 90–92, 95–101, 104–106, 111–113, 115, 116, 118–121
- RTL** register-transfer level 9, 10, 46, 68
- SDF** standard delay format 69, 71, 120
- SLT** system-level test 26, 92, 94, 105–112, 114, 121, 122
- SMON** speed monitor ring oscillator 50–52, 91, 95, 97, 100, 103, 104, 106–115, 117, 121, 122
- SoC** system-on-chip 1, 3, 8
- SPEF** standard parasitic exchange format 76, 77
- SPICE** Simulation Program with Integrated Circuit Emphasis 17
- SSTA** statistical static timing analysis 17
- STA** static timing analysis 9, 17, 46, 48, 53, 54, 56, 62, 66, 76
- STIL** standard test interface language 24
- TCL** tool command language 49
- W2W** wafer-to-wafer 13
- WCSS** within-cluster-sum-of-squares 63, 64
- WGL** waveform generation logic 24
- WID** within-die 13, 17, 26, 62

# Bibliography

- [1] 117th United States Congress. *Research and Development, Competition, and Innovation Act Supreme Court Security Funding Act of 2022*. Accessed: 2023-08-11. URL: <https://www.govinfo.gov/content/pkg/PLAW-117publ167/pdf/PLAW-117publ167.pdf>.
- [2] EUROPEAN COMMISSION. *REGULATION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL establishing a framework of measures for strengthening Europe's semiconductor ecosystem (Chips Act)*. Accessed: 2023-08-11. URL: <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex:52022PC0046>.
- [3] S. Pateras and T.-P. Tai. "Automotive semiconductor test". In: *International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*. IEEE, Apr. 2017. doi: [10.1109/vlsi-dat.2017.7939655](https://doi.org/10.1109/vlsi-dat.2017.7939655).
- [4] G. Georgakos, U. Schlichtmann, R. Schneider, and S. Chakraborty. "Reliability challenges for electric vehicles: From devices to architecture and systems software". In: *50th ACM/EDAC/IEEE Design Automation Conference (DAC)*. 2013, pp. 1–9. doi: [10.1145/2463209.2488855](https://doi.org/10.1145/2463209.2488855).
- [5] AEC Component Technical Committee. *AEC Documents*. URL: <http://www.aecouncil.com/AECDocuments.html>.
- [6] R. Raina. "Achieving Zero-Defects for Automotive Applications". In: *IEEE International Test Conference*. 2008, pp. 1–10. doi: [10.1109/TEST.2008.5483611](https://doi.org/10.1109/TEST.2008.5483611).
- [7] S. Mitra, E. Volkerink, E. McCluskey, and S. Eichenberger. "Delay defect screening using process monitor structures". In: *22<sup>nd</sup> IEEE VLSI Test Symposium*. IEEE, 2004. doi: [10.1109/vtest.2004.1299224](https://doi.org/10.1109/vtest.2004.1299224).
- [8] B. Cory, R. Kapur, and B. Underwood. "Speed binning with path delay test in 150-nm technology". In: *IEEE Design & Test of Computers*. IEEE, Sept. 2003, pp. 41–45. doi: [10.1109/mdt.2003.1232255](https://doi.org/10.1109/mdt.2003.1232255).
- [9] J. Zeng, M. Abadir, G. Vandling, L.-C. Wang, S. Karako, and J. Abraham. "On correlating structural tests with functional tests for speed binning of high performance design". In: *Fifth International Workshop on Microprocessor Test and Verification (MTV'04)*. 2004, pp. 103–109. doi: [10.1109/MTV.2004.17](https://doi.org/10.1109/MTV.2004.17).

- [10] P. Maxwell, I. Hartanto, and L. Bentz. "Comparing functional and structural tests". In: *IEEE International Test Conference (ITC)*. 2000, pp. 400–407. doi: [10.1109/TEST.2000.894231](https://doi.org/10.1109/TEST.2000.894231).
- [11] R. McLaughlin, S. Venkataraman, and C. Lim. "Automated Debug of Speed Path Failures Using Functional Tests". In: *27th IEEE VLSI Test Symposium*. IEEE, May 2009. doi: [10.1109/vts.2009.53](https://doi.org/10.1109/vts.2009.53).
- [12] P. Bernardi, M. Restifo, M. S. Reorda, D. Appello, C. Bertani, and D. Petrali. "Applicative System Level Test introduction to Increase Confidence on Screening Quality". In: *23rd International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*. IEEE, Apr. 2020. doi: [10.1109/ddecs50862.2020.9095569](https://doi.org/10.1109/ddecs50862.2020.9095569).
- [13] I. Polian, J. Anders, S. Becker, P. Bernardi, K. Chakrabarty, N. ElHamawy, M. Sauer, A. Singh, M. S. Reorda, and S. Wagner. "Exploring the Mysteries of System-Level Test". In: *29th Asian Test Symposium (ATS)*. IEEE, Nov. 2020. doi: [10.1109/ats49688.2020.9301557](https://doi.org/10.1109/ats49688.2020.9301557).
- [14] J. Chen, L.-C. Wang, P.-H. Chang, J. Zeng, S. Yu, and M. Mateja. "Data learning techniques and methodology for Fmax prediction". In: *International Test Conference*. IEEE, Nov. 2009. doi: [10.1109/test.2009.5355620](https://doi.org/10.1109/test.2009.5355620).
- [15] J. Chen, J. Zeng, L.-C. Wang, J. Rearick, and M. Mateja. "Selecting the most relevant structural Fmax for system Fmax correlation". In: *28th VLSI Test Symposium (VTS)*. IEEE, Apr. 2010. doi: [10.1109/vts.2010.5469604](https://doi.org/10.1109/vts.2010.5469604).
- [16] S.-P. Mu, M. C.-T. Chao, S.-H. Chen, and Y.-M. Wang. "Statistical Framework and Built-In Self-Speed-Binning System for Speed Binning Using On-Chip Ring Oscillators". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. IEEE, May 2016, pp. 1675–1687. doi: [10.1109/tvlsi.2015.2478921](https://doi.org/10.1109/tvlsi.2015.2478921).
- [17] S. Asai, ed. *VLSI Design and Test for Systems Dependability*. Springer Japan, 2019. doi: [10.1007/978-4-431-56594-9](https://doi.org/10.1007/978-4-431-56594-9).
- [18] M. Sadi, S. Kannan, L. Winemberg, and M. Tehranipoor. "SoC Speed Binning Using Machine Learning and On-Chip Slack Sensors". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. IEEE, May 2017, pp. 842–854. doi: [10.1109/tcad.2016.2602806](https://doi.org/10.1109/tcad.2016.2602806).
- [19] T.-B. Chan, P. Gupta, A. B. Kahng, and L. Lai. "Synthesis and Analysis of Design-Dependent Ring Oscillator (DDRO) Performance Monitors". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. IEEE, Oct. 2014, pp. 2117–2130. doi: [10.1109/tvlsi.2013.2282742](https://doi.org/10.1109/tvlsi.2013.2282742).

- [20] J. Heo, K. Jeong, T. Kim, and K. Choi. "Synthesis of Hardware Performance Monitoring and Prediction Flow Adapting to Near-Threshold Computing and Advanced Process Nodes". In: *25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, Jan. 2020. doi: [10.1109/asp-dac47756.2020.9045392](https://doi.org/10.1109/asp-dac47756.2020.9045392).
- [21] X. Wang, M. Tehranipoor, and R. Datta. "A novel architecture for on-chip path delay measurement". In: *International Test Conference*. IEEE, Nov. 2009. doi: [10.1109/test.2009.5355742](https://doi.org/10.1109/test.2009.5355742).
- [22] J. Waicukauski, E. Lindbloom, B. Rosen, and V. Iyengar. "Transition Fault Simulation". In: *IEEE Design & Test of Computers*. IEEE, 1987, pp. 32–38. doi: [10.1109/tdt.1987.295104](https://doi.org/10.1109/tdt.1987.295104).
- [23] C. J. Lin and S. Reddy. "On Delay Fault Testing in Logic Circuits". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. IEEE, Sept. 1987, pp. 694–703. doi: [10.1109/tcad.1987.1270315](https://doi.org/10.1109/tcad.1987.1270315).
- [24] R. Cantoro, M. Huch, T. Kilian, R. Martone, U. Schlichtmann, and G. Squillero. "Machine Learning based Performance Prediction of Microcontrollers using Speed Monitors". In: *IEEE International Test Conference (ITC)*. IEEE, Nov. 2020. doi: [10.1109/itc44778.2020.9325253](https://doi.org/10.1109/itc44778.2020.9325253).
- [25] A. D. Singh. "An Adaptive Approach to Minimize System Level Tests Targeting Low Voltage DVFS Failures". In: *IEEE International Test Conference (ITC)*. IEEE, Nov. 2019. doi: [10.1109/itc44170.2019.9000173](https://doi.org/10.1109/itc44170.2019.9000173).
- [26] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge. "Razor: a low-power pipeline based on circuit-level timing speculation". In: *36th Annual IEEE/ACM International Symposium on Microarchitecture*. MICRO-36. 2003, pp. 7–18. doi: [10.1109/MICRO.2003.1253179](https://doi.org/10.1109/MICRO.2003.1253179).
- [27] D. Blaauw, S. Kalaiselvan, K. Lai, W.-H. Ma, S. Pant, C. Tokunaga, S. Das, and D. Bull. "Razor II: In Situ Error Detection and Correction for PVT and SER Tolerance". In: *IEEE International Solid-State Circuits Conference - Digest of Technical Papers*. IEEE, Feb. 2008. doi: [10.1109/isscc.2008.4523226](https://doi.org/10.1109/isscc.2008.4523226).
- [28] A. Benhassain, S. Mhira, F. Cacho, V. Huard, and L. Anghel. "In-situ slack monitors: taking up the challenge of on-die monitoring of variability and reliability". In: *1<sup>st</sup> IEEE International Verification and Security Workshop (IVSW)*. IEEE, July 2016. doi: [10.1109/ivsw.2016.7566606](https://doi.org/10.1109/ivsw.2016.7566606).

- [29] T.-B. Chan, P. Gupta, A. B. Kahng, and L. Lai. "DDRO: A novel performance monitoring methodology based on design-dependent ring oscillators". In: *Thirteenth International Symposium on Quality Electronic Design (ISQED)*. IEEE, Mar. 2012. doi: [10.1109/isqed.2012.6187559](https://doi.org/10.1109/isqed.2012.6187559).
- [30] W. C. Wu, C. L. Lee, M. S. Wu, J. E. Chen, and M. S. Abadir. "Oscillation ring delay test for high performance microprocessors". In: *Journal of Electronic Testing*. Springer Science and Business Media LLC, 2000, pp. 147–155. doi: [10.1023/a:1008365428314](https://doi.org/10.1023/a:1008365428314).
- [31] X. Wang, M. Tehranipoor, and R. Datta. "Path-RO: A novel on-chip critical path delay measurement under process variations". In: *2008 IEEE/ACM International Conference on Computer-Aided Design*. IEEE, Nov. 2008. doi: [10.1109/iccad.2008.4681644](https://doi.org/10.1109/iccad.2008.4681644).
- [32] U. Schlichtmann. "Frontiers of timing". In: *ACM/IEEE International Workshop on System Level Interconnect Prediction (SLIP)*. IEEE, June 2017. doi: [10.1109/slip.2017.7974912](https://doi.org/10.1109/slip.2017.7974912).
- [33] F. Firouzi, F. Ye, K. Chakrabarty, and M. B. Tahoori. "Representative critical-path selection for aging-induced delay monitoring". In: *2013 IEEE International Test Conference (ITC)*. IEEE, Sept. 2013. doi: [10.1109/test.2013.6651924](https://doi.org/10.1109/test.2013.6651924).
- [34] D. Lorenz, M. Barke, and U. Schlichtmann. "Aging analysis at gate and macro cell level". In: *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, Nov. 2010. doi: [10.1109/iccad.2010.5654309](https://doi.org/10.1109/iccad.2010.5654309).
- [35] Y. Hu, J. YE, Z. Shi, and X. LI. "LAPS: Layout-Aware Path Selection for Post-Silicon Timing Characterization". In: *IEICE Transactions on Information and Systems*. Feb. 2017, pp. 323–331. doi: [10.1587/transinf.2016EDP7184](https://doi.org/10.1587/transinf.2016EDP7184).
- [36] N. Callegari, P. Bastani, L.-C. Wang, S. Chakravarty, and A. Tetelbaum. "Path selection for monitoring unexpected systematic timing effects". In: *Asia and South Pacific Design Automation Conference*. IEEE, Jan. 2009. doi: [10.1109/aspdac.2009.4796575](https://doi.org/10.1109/aspdac.2009.4796575).
- [37] P. Bastani, N. Callegari, L.-C. Wang, and M. Abadir. "Diagnosis of design-silicon timing mismatch with feature encoding and importance ranking - the methodology explained". In: *IEEE International Test Conference*. IEEE, Oct. 2008. doi: [10.1109/test.2008.4700588](https://doi.org/10.1109/test.2008.4700588).
- [38] J. K. Rangan, N. P. Aryan, J. Bargfrede, C. Funke, and H. Graeb. "Timing Variability Analysis of Digital CMOS Circuits". In: *Reliability by Design; 9. ITG/GMM/GI-Symposium*. 2017, pp. 1–2.
- [39] C.-L. Chang and C. H.-P. Wen. "Accurate performance evaluation of VLSI designs with selected CMOS process parameters". In: *IET Circuits, Devices & Systems*. Institution of Engineering and Technology (IET), 2018, pp. 116–123. doi: <https://doi.org/10.1049/iet-cds.2017.0097>.

- [40] T. Kilian, H. Ahrens, D. Tille, M. Huch, and U. Schlichtmann. "A Scalable Design Flow for Performance Monitors Using Functional Path Ring Oscillators". In: *IEEE International Test Conference (ITC)*. IEEE, Oct. 2021. doi: [10.1109/itc50571.2021.00041](https://doi.org/10.1109/itc50571.2021.00041).
- [41] T. Kilian, M. Hanel, D. Tille, M. Huch, and U. Schlichtmann. "Reducing Routing Overhead by Self-Enabling Functional Path Ring Oscillators". In: *IEEE European Test Symposium (ETS)*. IEEE, May 2022. doi: [10.1109/ets54262.2022.9810382](https://doi.org/10.1109/ets54262.2022.9810382).
- [42] T. Kilian, M. Hanel, D. Tille, M. Huch, and U. Schlichtmann. "A Path Selection Flow for Functional Path Ring Oscillators using Physical Design Data". In: *IEEE International Test Conference (ITC)*. IEEE, Sept. 2022.
- [43] T. Kilian, A. Sengupta, D. Tille, M. Huch, and U. Schlichtmann. "An efficient High-Volume Production Performance Screening using On-Chip Ring Oscillators". In: *2023 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. Oct. 2023, pp. 1–6.
- [44] T. Kilian, H. Ahrens, D. Tille, M. Huch, and U. Schlichtmann. "Scalable Implementation of Functional Path Ring Oscillator for MCU Performance Screening". In: *33. GI/GMM/ITG Testmethoden und Zuverlässigkeit von Schaltungen und Systemen (TuZ 2021)*. 2021.
- [45] T. Kilian, H. Ahrens, D. Tille, M. Huch, and U. Schlichtmann. "Automatic and Scalable Implementation Flow of Performance Monitors for Automotive MCU Using Functional Path Ring Oscillators". In: *ARTe 2021 - First International Workshop on Automotive Reliability and Test in Europe*. 2021.
- [46] T. Kilian, H. Ahrens, D. Tille, M. Huch, and U. Schlichtmann. "A Layout-aware Selection Flow for Functional Path Ring Oscillators". In: *34. GI/GMM/ITG Testmethoden und Zuverlässigkeit von Schaltungen und Systemen (TuZ 2022)*. 2022.
- [47] T. Kilian, D. Tille, M. Huch, and U. Schlichtmann. "Reducing Routing Overhead using Natural Loops". In: *eARTS 2022 - European Automotive Reliability, Test and Safety workshop*. 2022.
- [48] T. Kilian, D. Tille, M. Huch, and U. Schlichtmann. "The Capabilities of Functional Path Ring Oscillators for Performance Screening". In: *36. GI/GMM/ITG Testmethoden und Zuverlässigkeit von Schaltungen und Systemen (TuZ 2024)*. 2024.
- [49] T. Kilian, D. Tille, M. Huch, M. Hanel, and U. Schlichtmann. "Performance Screening Using Functional Path Ring Oscillators". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. IEEE, 2023, pp. 711–724. doi: [10.1109/tvlsi.2023.3252471](https://doi.org/10.1109/tvlsi.2023.3252471).

- [50] N. Bellarmino, R. Cantoro, M. Huch, T. Kilian, R. Martone, U. Schlichtmann, and G. Squillero. "Exploiting Active Learning for Microcontroller Performance Prediction". In: *IEEE European Test Symposium (ETS)*. IEEE, May 2021. doi: [10.1109/ets50041.2021.9465472](https://doi.org/10.1109/ets50041.2021.9465472).
- [51] N. Bellarmino, R. Cantoro, M. Huch, T. Kilian, R. Martone, U. Schlichtmann, and G. Squillero. "A Multi-Label Active Learning Framework for Microcontroller Performance Screening". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 2023. doi: [10.1109/TCAD.2023.3245989](https://doi.org/10.1109/TCAD.2023.3245989).
- [52] T. Kilian, H. Ahrens, M. Huch, and D. Tille. *INTEGRATED CIRCUIT, TEST ASSEMBLY AND METHOD FOR TESTING AN INTEGRATED CIRCUIT*. Infineon Technologies AG. US Patent US 2023079599 A1, Mar. 2023. DE Patent DE 102021123889 B3, Feb. 2023.
- [53] T. Kilian, H. Ahrens, M. Huch, and D. Tille. *INTEGRATED CIRCUIT, TEST ASSEMBLY AND METHOD FOR TESTING AN INTEGRATED CIRCUIT*. Infineon Technologies AG. US Patent US 20230138651 A1, Mar. 2023. DE Patent DE 102021128331 B3, Mar. 2023.
- [54] V. S. Chakravarthi and S. R. Koteswar. *SoC Physical Design*. Springer International Publishing, 2022. doi: [10.1007/978-3-030-98112-9](https://doi.org/10.1007/978-3-030-98112-9).
- [55] V. S. Chakravarthi. *A Practical Approach to VLSI System on Chip (SoC) Design*. Springer International Publishing, 2020. doi: [10.1007/978-3-030-23049-4](https://doi.org/10.1007/978-3-030-23049-4).
- [56] R. J. Baker. *CMOS: Circuit Design, Layout, and Simulation*. WILEY, July 2019. ISBN: 1119481511. URL: [https://www.ebook.de/de/product/36062569/r\\_jacob\\_baker\\_cmos\\_circuit\\_design\\_layout\\_and\\_simulation.html](https://www.ebook.de/de/product/36062569/r_jacob_baker_cmos_circuit_design_layout_and_simulation.html).
- [57] M. Chanda, S. De, and A. Sarkar. *Low Power VLSI Design*. Gruyter, Walter de GmbH, Aug. 2016. ISBN: 3110455293. URL: [https://www.ebook.de/de/product/33538502/manash\\_chanda\\_swapnadip\\_de\\_angsuman\\_sarkar\\_low\\_power\\_vlsi\\_design.html](https://www.ebook.de/de/product/33538502/manash_chanda_swapnadip_de_angsuman_sarkar_low_power_vlsi_design.html).
- [58] S. Mittal. "A Survey of Architectural Techniques for Managing Process Variation". In: *ACM Computing Surveys*. Association for Computing Machinery (ACM), Feb. 2016, pp. 1–29. doi: [10.1145/2871167](https://doi.org/10.1145/2871167).
- [59] S. Borkar. "Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation". In: *IEEE Micro*. IEEE, Nov. 2005, pp. 10–16. doi: [10.1109/mm.2005.110](https://doi.org/10.1109/mm.2005.110).
- [60] V. Champac and J. G. Gervacio. *Timing Performance of Nanometer Digital Circuits Under Process Variations*. Springer International Publishing, 2018. doi: [10.1007/978-3-319-75465-9](https://doi.org/10.1007/978-3-319-75465-9).

- [61] N. Weste. *CMOS VLSI design : a circuits and systems perspective*. Boston: Addison Wesley, 2011. ISBN: 9780321547743.
- [62] K. Bernstein, K. M. Carrig, C. M. Durham, P. R. Hansen, D. Hogenmiller, E. J. Nowak, and N. J. Rohrer. *High Speed CMOS Design Styles*. Springer US, 1999. doi: [10.1007/978-1-4615-5573-5](https://doi.org/10.1007/978-1-4615-5573-5).
- [63] Q. Xu, N. Yu, and F. Essaf. "Improved Wafer Map Inspection Using Attention Mechanism and Cosine Normalization". In: *Machines*. MDPI AG, Feb. 2022, p. 146. doi: [10.3390/machines10020146](https://doi.org/10.3390/machines10020146).
- [64] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer. "Statistical Timing Analysis: From Basic Principles to State of the Art". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. IEEE, Apr. 2008, pp. 589–607. doi: [10.1109/tcad.2007.907047](https://doi.org/10.1109/tcad.2007.907047).
- [65] S. Bhunia and S. Mukhopadhyay. *Low-Power Variation-Tolerant Design in Nanometer Silicon*. Springer US, 2011. doi: [10.1007/978-1-4419-7418-1](https://doi.org/10.1007/978-1-4419-7418-1).
- [66] M. Wirnshofer. *Variation-Aware Adaptive Voltage Scaling for Digital CMOS Circuits*. Springer Netherlands, 2013. doi: [10.1007/978-94-007-6196-4](https://doi.org/10.1007/978-94-007-6196-4).
- [67] T. Sakurai and A. Newton. "Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas". In: *IEEE Journal of Solid-State Circuits*. IEEE, Apr. 1990, pp. 584–594. doi: [10.1109/4.52187](https://doi.org/10.1109/4.52187).
- [68] L.-T. Wang, C. E. Stroud, and N. A. Touba. *System-on-chip test architectures. nanometer design for testability*. Morgan Kaufmann Publishers, 2008, p. 856. ISBN: 9780123739735.
- [69] K. Bernstein, D. J. Frank, A. E. Gattiker, W. Haensch, B. L. Ji, S. R. Nassif, E. J. Nowak, D. J. Pearson, and N. J. Rohrer. "High-performance CMOS variability in the 65-nm regime and beyond". In: *IBM Journal of Research and Development*. IBM, July 2006, pp. 433–449. doi: [10.1147/rd.504.0433](https://doi.org/10.1147/rd.504.0433).
- [70] M. Cho, M. Khellah, K. Chae, K. Ahmed, J. Tschanz, and S. Mukhopadhyay. "Characterization of Inverse Temperature Dependence in logic circuits". In: *IEEE Custom Integrated Circuits Conference*. IEEE, Sept. 2012. doi: [10.1109/cicc.2012.6330659](https://doi.org/10.1109/cicc.2012.6330659).
- [71] R. C. J. Bhasker. *Static Timing Analysis for Nanometer Designs*. Springer-Verlag New York Inc., Apr. 2009. ISBN: 0387938192. URL: [https://www.ebook.de/de/product/8020071/j\\_bhasker\\_rakesh\\_chadha\\_static\\_timing\\_analysis\\_for\\_nanometer\\_designs.html](https://www.ebook.de/de/product/8020071/j_bhasker_rakesh_chadha_static_timing_analysis_for_nanometer_designs.html).
- [72] L. W. Nagel and D. Pederson. *SPICE (Simulation Program with Integrated Circuit Emphasis)*. Tech. rep. EECS Department, University of California, Berkeley, 1973.

- [73] B. Li, M. Hashimoto, and U. Schlichtmann. "From Process Variations to Reliability: A Survey of Timing of Digital Circuits in the Nanometer Era". In: *IPSJ Transactions on System LSI Design Methodology*. Information Processing Society of Japan, 2018, pp. 2–15. doi: [10.2197/ipsjtsldm.11.2](https://doi.org/10.2197/ipsjtsldm.11.2).
- [74] Q. Liu and S. S. Sapatnekar. "Confidence Scalable Post-Silicon Statistical Delay Prediction under Process Variations". In: *44th ACM/IEEE Design Automation Conference*. 2007, pp. 497–502. doi: [10.1145/1278480.1278609](https://doi.org/10.1145/1278480.1278609).
- [75] A. B. Kahng. "New game, new goal posts". In: *52nd ACM/IEEE Design Automation Conference*. ACM, June 2015. doi: [10.1145/2744769.2747937](https://doi.org/10.1145/2744769.2747937).
- [76] X. Liu, A. M. Gough, and J. Li. "Semiconductor corner lot generation robust to process variation: Modeling and analysis". In: *IISE Transactions*. Informa UK Limited, Nov. 2017, pp. 126–139. doi: [10.1080/24725854.2017.1383636](https://doi.org/10.1080/24725854.2017.1383636).
- [77] D. Lorenz, M. Barke, and U. Schlichtmann. "Efficiently analyzing the impact of aging effects on large integrated circuits". In: *Microelectronics Reliability*. Elsevier BV, Aug. 2012, pp. 1546–1552. doi: [10.1016/j.microrel.2011.12.029](https://doi.org/10.1016/j.microrel.2011.12.029).
- [78] S. S. Sapatnekar. "What happens when circuits grow old: Aging issues in CMOS design". In: *International Symposium on VLSI Technology, Systems and Application (VLSI-TSA)*. IEEE, Apr. 2013. doi: [10.1109/vlsi-tsa.2013.6545621](https://doi.org/10.1109/vlsi-tsa.2013.6545621).
- [79] S. Karapetyan and U. Schlichtmann. "Integrating aging aware timing analysis into a commercial STA tool". In: *VLSI Design, Automation and Test(VLSI-DAT)*. IEEE, Apr. 2015. doi: [10.1109/vlsi-dat.2015.7114528](https://doi.org/10.1109/vlsi-dat.2015.7114528).
- [80] J. M. Galey, R. E. Norby, and J. P. Roth. "Techniques for the diagnosis of switching circuit failures". In: *IEEE Transactions on Communication and Electronics*. IEEE, Sept. 1964, pp. 509–514. doi: [10.1109/tcome.1964.6539498](https://doi.org/10.1109/tcome.1964.6539498).
- [81] R. D. Eldred. "Test Routines Based on Symbolic Logical Statements". In: *Journal of the ACM*. Association for Computing Machinery (ACM), Jan. 1959, pp. 33–37. doi: [10.1145/320954.320957](https://doi.org/10.1145/320954.320957).
- [82] G. L. Smith. "Model for Delay Faults Based Upon Paths." In: *International Test Conference (ITC)*. 1985, pp. 342–351.
- [83] E. S. Park. "Robust and nonrobust tests for path delay faults in a combinational circuit". In: *International Test Conference (ITC)*. 1987, pp. 1027–1034.
- [84] A. Pramanick and S. Reddy. "On the detection of delay faults". In: *International Test Conference (ITC)*. IEEE. doi: [10.1109/test.1988.207872](https://doi.org/10.1109/test.1988.207872).

- [85] J. Savir and S. Patil. "Broad-side delay test". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. IEEE, 1994, pp. 1057–1064. doi: [10.1109/43.298042](https://doi.org/10.1109/43.298042).
- [86] N. Ahmed, M. Tehranipoor, C. P. Ravikumar, and K. M. Butler. "Local At-Speed Scan Enable Generation for Transition Fault Testing Using Low-Cost Testers". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. IEEE, May 2007, pp. 896–906. doi: [10.1109/tcad.2006.884405](https://doi.org/10.1109/tcad.2006.884405).
- [87] M. Bhushan and M. B. Ketchen. *CMOS Test and Evaluation*. Springer New York, 2015. doi: [10.1007/978-1-4939-1349-7](https://doi.org/10.1007/978-1-4939-1349-7).
- [88] H. Onodera and H. Terada. "Characterization of WID delay variability using RO-array test structures". In: *IEEE 8<sup>th</sup> International Conference on ASIC*. IEEE, Oct. 2009. doi: [10.1109/asicon.2009.5351332](https://doi.org/10.1109/asicon.2009.5351332).
- [89] M. Bhushan, A. Gattiker, M. Ketchen, and K. Das. "Ring Oscillators for CMOS Process Tuning and Variability Control". In: *IEEE Transactions on Semiconductor Manufacturing*. IEEE, Feb. 2006, pp. 10–18. doi: [10.1109/tsm.2005.863244](https://doi.org/10.1109/tsm.2005.863244).
- [90] K. Maragos, G. Lentaris, and D. Soudris. "In-the-Field Mitigation of Process Variability for Improved FPGA Performance". In: *IEEE Transactions on Computers*. IEEE, July 2019, pp. 1049–1063. doi: [10.1109/tc.2019.2898833](https://doi.org/10.1109/tc.2019.2898833).
- [91] A. L. Samuel. "Some Studies in Machine Learning Using the Game of Checkers". In: *IBM Journal of Research and Development*. IBM, July 1959, pp. 210–229. doi: [10.1147/rd.33.0210](https://doi.org/10.1147/rd.33.0210).
- [92] A. V. Joshi. *Machine Learning and Artificial Intelligence*. Springer International Publishing, 2020. doi: [10.1007/978-3-030-26622-6](https://doi.org/10.1007/978-3-030-26622-6).
- [93] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer New York, 2009. doi: [10.1007/978-0-387-84858-7](https://doi.org/10.1007/978-0-387-84858-7).
- [94] M. A. Syakur, B. K. Khotimah, E. M. S. Rochman, and B. D. Satoto. "Integration K-Means Clustering Method and Elbow Method For Identification of The Best Customer Profile Cluster". In: *IOP Conference Series: Materials Science and Engineering*. IOP Publishing, Apr. 2018. doi: [10.1088/1757-899x/336/1/012017](https://doi.org/10.1088/1757-899x/336/1/012017).
- [95] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *2nd International Conference on Knowledge Discovery*. 1996, pp. 226–231.
- [96] G. R. Naik, ed. *Advances in Principal Component Analysis*. Springer Singapore, 2018. doi: [10.1007/978-981-10-6704-4](https://doi.org/10.1007/978-981-10-6704-4).

- [97] H.-G. Stratigopoulos. "Machine learning applications in IC testing". In: *IEEE 23<sup>rd</sup> European Test Symposium (ETS)*. IEEE, May 2018. doi: [10.1109/ets.2018.8400701](https://doi.org/10.1109/ets.2018.8400701).
- [98] M. Rapp, H. Amrouch, Y. Lin, B. Yu, D. Z. Pan, M. Wolf, and J. Henkel. "MLCAD: A Survey of Research in Machine Learning for CAD". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. IEEE, Oct. 2022, pp. 3162–3181. doi: [10.1109/tcad.2021.3124762](https://doi.org/10.1109/tcad.2021.3124762).
- [99] ISO 26262. *Road vehicles – Functional safety*. Norm. 2018.
- [100] A. Ralf. "A Tutorial of How to Ensure High Automotive Microcontroller Quality". In: *IEEE European Test Symposium (ETS)*. IEEE, May 2021. doi: [10.1109/ets50041.2021.9465379](https://doi.org/10.1109/ets50041.2021.9465379).
- [101] U. Abelein, H. Lochner, D. Hahn, and S. Straube. "Complexity, quality and robustness - the challenges of tomorrow's automotive electronics". In: *Design, Automation and Test in Europe Conference (DATE)*. IEEE, Mar. 2012. doi: [10.1109/date.2012.6176573](https://doi.org/10.1109/date.2012.6176573).
- [102] N. Mukherjee and J. Rajski. "Digital Testing of ICs for Automotive Applications". In: *29th International Conference on VLSI*. IEEE, Jan. 2016. doi: [10.1109/vlsid.2016.134](https://doi.org/10.1109/vlsid.2016.134).
- [103] B. Smith. "Six-sigma design (quality control)". In: *IEEE Spectrum*. IEEE, 1993, pp. 43–47. doi: [10.1109/6.275174](https://doi.org/10.1109/6.275174).
- [104] N. Vivekananthamoorthy and S. Shanmuganathan. *Six Sigma*. IntechOpen, 2011. ISBN: 9533073705. URL: [https://www.ebook.de/de/product/37154332/six\\_sigma.html](https://www.ebook.de/de/product/37154332/six_sigma.html).
- [105] J. Remmers, D. Lee, and R. Fisette. "Hierarchical DFT with enhancements for AC scan, test scheduling and on-chip compression - a case study". In: *IEEE International Conference on Test*, 2005. 2005. doi: [10.1109/TEST.2005.1584034](https://doi.org/10.1109/TEST.2005.1584034).
- [106] R. S. Shelar and M. Patyra. "Impact of Local Interconnects on Timing and Power in a High Performance Microprocessor". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. IEEE, Oct. 2013, pp. 1623–1627. doi: [10.1109/tcad.2013.2266404](https://doi.org/10.1109/tcad.2013.2266404).
- [107] I. B. Mohamad and D. Usman. "Standardization and Its Effects on K-Means Clustering Algorithm". In: *Research Journal of Applied Sciences, Engineering and Technology*. Maxwell Scientific Publication Corp., Sept. 2013, pp. 3299–3303. doi: [10.19026/rjaset.6.3638](https://doi.org/10.19026/rjaset.6.3638).
- [108] ANSYS Inc. <https://www.ansys.com/de-de/products/semiconductors/ansys-redhawk-sc>. Accessed: 2022-03-09.
- [109] MunEDA GmbH. <https://www.muneda.com/circuit-analysis-and-verification-tools/>. Accessed: 2022-03-09.

- [110] H. Dornelas, A. Schmidt, G. Strube, and E. Fabris. "New Technology Migration Methodology for Analog IC Design using MunEDA tools". In: 2015. doi: [10.13140/RG.2.2.32187.41767](https://doi.org/10.13140/RG.2.2.32187.41767).
- [111] R. B. Bendel, S. S. Higgins, J. E. Teberg, and D. A. Pyke. "Comparison of skewness coefficient, coefficient of variation, and Gini coefficient as inequality measures within populations". In: *Oecologia*. Springer Science and Business Media LLC, 1989, pp. 394–400. doi: [10.1007/bf00379115](https://doi.org/10.1007/bf00379115).
- [112] X. Lin. "Power Supply Droop and Its Impacts on Structural At-Speed Testing". In: *IEEE 21st Asian Test Symposium*. IEEE, Nov. 2012. doi: [10.1109/ats.2012.63](https://doi.org/10.1109/ats.2012.63).
- [113] Z.-H. Zhou. *Machine Learning*. Springer Singapore, 2021. doi: [10.1007/978-981-15-1967-3](https://doi.org/10.1007/978-981-15-1967-3).
- [114] T. M. Mitchell. *Machine Learning*. McGraw Hill, 1997.
- [115] L. Breiman. "Random forests". In: *Machine Learning*. Springer Science and Business Media LLC, 2001, pp. 5–32. doi: [10.1023/a:1010933404324](https://doi.org/10.1023/a:1010933404324).
- [116] L. Breiman. "Bagging predictors". In: *Machine Learning*. Springer Science and Business Media LLC, Aug. 1996, pp. 123–140. doi: [10.1007/bf00058655](https://doi.org/10.1007/bf00058655).
- [117] D. Zhang and X. Wang. "An on-chip binning sensor for low-cost and accurate speed binning". In: *2<sup>nd</sup> IEEE International Conference on Integrated Circuits and Microsystems (ICICM)*. IEEE, Nov. 2017. doi: [10.1109/icam.2017.8242158](https://doi.org/10.1109/icam.2017.8242158).
- [118] D. Chicco and G. Jurman. "The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation". In: *BMC Genomics*. Springer Science and Business Media LLC, Jan. 2020. doi: [10.1186/s12864-019-6413-7](https://doi.org/10.1186/s12864-019-6413-7).
- [119] G. C. McDonald. "Ridge regression". In: *Wiley Interdisciplinary Reviews: Computational Statistics*. Wiley, July 2009, pp. 93–100. doi: [10.1002/wics.14](https://doi.org/10.1002/wics.14).
- [120] J. M. Sutter, J. H. Kalivas, and P. M. Lang. "Which principal components to utilize for principal component regression". In: *Journal of Chemometrics*. Wiley, July 1992, pp. 217–225. doi: [10.1002/cem.1180060406](https://doi.org/10.1002/cem.1180060406).
- [121] D. Chicco, M. J. Warrens, and G. Jurman. "The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation". In: *PeerJ Computer Science*. PeerJ, July 2021. doi: [10.7717/peerj-cs.623](https://doi.org/10.7717/peerj-cs.623).
- [122] Y. Liu, N. Mukherjee, J. Rajska, S. M. Reddy, and J. Tyszer. "Deterministic Stellar BIST for In-System Automotive Test". In: *IEEE International Test Conference (ITC)*. IEEE, Oct. 2018. doi: [10.1109/test.2018.8624872](https://doi.org/10.1109/test.2018.8624872).